|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки 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 Системы обработки информации и управления

**РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

***К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ***

***НА ТЕМУ:***

***«Модуль формирования метаграфа на основе текстового описания»***

Студент \_\_ИУ-5 84\_\_ **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_А.А.Фадеев\_\_\_**

(Группа) (Подпись, дата) (И.О.Фамилия)

Руководитель ВКР **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_В.Б.Тимофеев\_\_**

(Подпись, дата) (И.О.Фамилия)

Консультант **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_Ю.Е.Гапанюк\_\_**

(Подпись, дата) (И.О.Фамилия)

Нормоконтролер **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_Ю.Н.Кротов\_\_\_**

(Подпись, дата) (И.О.Фамилия)

2020 г.

РЕФЕРАТ

Расчётно-пояснительная записка квалификационной работы бакалавра содержит 77 страниц. С приложениями объем составляет 103 страниц. Работа включает в себя 31 таблицу и 26 иллюстраций. В процессе выполнения было использовано 16 источников.

Объектом разработки является программный модуль, формирующий метаграф на основе результатов синтаксического анализа русскоязычного текста.

После синтаксического анализа исходный текст преобразуется в метаграф. Он отражает синтаксическую структуру входной последовательности и хорошо подходит для дальнейшей обработки.

В процессе выполнения квалификационной работы бакалавра были изучены различные аспекты работы с естественным языком (ЕЯ) и её автоматизации.

Расчётно-пояснительная записка состоит из двух частей.

Постановка задач разработки рассматривает данную предметную область, процессы, подлежащие автоматизации при создании синтаксического анализатора и метаграфа, различные подходы и алгоритмы, которые для этого используются.

Конструкторско-технологическая часть включает выбор языка программирования для разработки программного изделия, проектирование, правил, базы данных, алгоритмов синтаксического анализа, формирования метаграфа и интерфейса пользователя.

В расчётно-пояснительную записку включены 3 приложения. Она составлена с учетом рекомендаций [10], [11] и [12].

**СОДЕРЖАНИЕ**

[РЕФЕРАТ 2](#_Toc43147522)

[ВВЕДЕНИЕ 5](#_Toc43147524)

[1. ПОСТАНОВКА ЗАДАЧ РАЗРАБОТКИ 6](#_Toc43147525)

[1.1. Общетехническое обоснование разработки 6](#_Toc43147526)

[1.1.1. Постановка цели работы 6](#_Toc43147527)

[1.1.2. Постановка задач 6](#_Toc43147528)

[1.1.3. Описание предметной области 7](#_Toc43147529)

[1.1.4. Перечень процессов, подлежащих автоматизации 9](#_Toc43147530)

[1.1.5. Выбор критериев качества 10](#_Toc43147531)

[1.1.6. Анализ аналогов и прототипов 12](#_Toc43147532)

[1.2. Анализ существующих алгоритмов и внедрение отличительных особенностей 14](#_Toc43147533)

[2. КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 21](#_Toc43147534)

[2.1. Выбор программных средств 21](#_Toc43147535)

[2.2. Выбор аппаратных средств 22](#_Toc43147536)

[2.3. Выбор языка разработки 23](#_Toc43147537)

[2.4. Структура программного изделия 24](#_Toc43147538)

[2.5. Проектирование структуры правил 26](#_Toc43147539)

[2.6. Проектирование модели базы данных 40](#_Toc43147540)

[2.7. Разработка программного продукта 58](#_Toc43147541)

[2.7.1. Разработка интерфейса пользователя 58](#_Toc43147542)

[2.7.2. Разработка порядка работы алгоритмов синтаксического анализа 60](#_Toc43147543)

[2.7.3. Разработка алгоритма синтаксического анализа 62](#_Toc43147544)

[2.7.4. Разработка структуры классов модуля построения метаграфа 63](#_Toc43147545)

[2.7.5. Разработка алгоритма преобразования результатов синтаксического анализа в метаграф 65](#_Toc43147546)

[2.8. Результаты опытной эксплуатации 67](#_Toc43147547)

[ЗАКЛЮЧЕНИЕ 73](#_Toc43147548)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 75](#_Toc43147549)

[ПРИЛОЖЕНИЕ А ГРАФИЧЕСКИЕ МАТЕРИАЛЫ 77](#_Toc43147550)

[ПРИЛОЖЕНИЕ В ТЕХНИЧЕСКОЕ ЗАДАНИЕ 83](#_Toc43147551)

[ПРИЛОЖЕНИЕ С ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ 95](#_Toc43147552)

# ВВЕДЕНИЕ

Квалификационная работа на тему «Формирование метаграфа на основе текстового описания» посвящена разработке модуля для построения метаграфа по результатам синтаксического анализа исходного русскоязычного текста.

В данной области множество научных работ и различных продуктов, дошедших до простого пользователя: электронные переводчики, распознаватели текста, поисковые системы, вопросно-ответные системы. Но почти все доступные сторонним пользователям системы не понимают слова так, как это делает человек, а лишь обрабатывают цепочки символов, используя статистику.

При работе с ЕЯ основными задачами являются разбор и анализ входного текста или синтез выходных предложений. Анализа текстов на ЕЯ состоит из 4-х этапов: морфологический, синтаксический, семантический и прагматический.

В данной квалификационной работе подробно рассматривается синтаксический этап разбора входного текста. Морфологический этап разбора проводится за счёт сторонних программных средств, о которых говорится в пункте 2.1 данной работы.

Морфологический анализ включает определение морфологических характеристик слова, таких как часть речи, род, число, падеж и т.д.

Синтаксический анализ, который содержит определение синтаксических ролей слов и словосочетаний в предложении. Результатом анализа является метаграф синтаксических зависимостей слов, который отражает структуру их взаимодействия в рамках одного предложения.

1. ПОСТАНОВКА ЗАДАЧ РАЗРАБОТКИ
   1. Общетехническое обоснование разработки
      1. Постановка цели работы

В настоящее время существует множество систем, каким-либо образом анализирующих содержание текста. Однако почти все открытые для пользователей системы не анализируют текст как таковой, а лишь исследуют «важность» тех или иных слов внутри него, проводя максимально примитивный семантический анализ. Синтаксический анализ предваряет семантический, поэтому его роль в анализе текстов на ЕЯ не менее важна.

В связи с этим цель работы формулируется следующим образом: предоставление возможности синтаксического анализа текста на русском языке и построения структуры, которая отражает синтаксическую структуру входной последовательности, без сколь угодно простого семантического анализа. А также предоставлении возможности консольной визуализации метаграфа-результата синтаксического разбора предложений текста.

* + 1. Постановка задач

В соответствии с целью данной работы, подразумевающей под собой создание синтаксического анализатора без каких-либо функций семантического анализатора, задачами проектирования становятся:

* Создание алгоритма фрагментации сложных предложений;
* Создание алгоритма поиска грамматической основы предложения;
* Создание структуры правил выделения синтаксических конструкций;
* Создание алгоритма применения правил для построения синтаксических групп (СГ) на фрагментах простых предложений в составе сложных;
* Формирование структуры синтаксического метаграфа;
* Создание алгоритма построения метаграфа;
* Разработка модуля консольной визуализации метаграфа.
  + 1. Описание предметной области

Текст – довольно однородная структура, а слова, являющиеся его элементами, более-менее равноправны, поскольку в каком-либо тексте мы, люди без помощи компьютера, не можем придать одному слову значение всего текста.

В настоящее время количество и, следовательно, объём электронных документов увеличивается с гигантской скоростью. В связи с этим каждый день возрастает потребность в обработке неструктурированной текстовой информации, повышению качества и эффективности имеющихся методов и разработке универсальных технологий обработки текста. В настоящее время актуально фактически любое направление, связанное с облегчением этой глобальной задачи: поиск информации, фильтрация, кластеризация документов, поиск ответов на вопросы, автоматическое аннотирование документа и группы документов, поиск похожих документов и дубликатов, сегментирование документов, формирование баз знаний на их основе и многое другое.

Сейчас такая обработка ведётся в основном статистическими методами, подкреплёнными, например, технологией нейронных сетей. Уже несколько десятков лет ведётся научная работа по поиску оптимальных алгоритмов семантической обработки текста (семантических анализаторов), которая позволит компьютеру понять текст.

Если обобщить различные определения семантического анализатора, то данное понятие становится очень простым. Семантический анализатор – это программное средство, которое, благодаря своей алгоритмической мощи в общем случае позволяет понимать, анализировать, находить истину и ложь, недосказанности и противоречия, аннотировать и редактировать тексты, находить в них смысл и знания, организовать диалог с пользователями или отвечать на их вопросы по каким-либо текстам и т.п. Однако первоочередной задачей семантических анализаторов всегда есть и будет поиск знаний в различных текстовых источниках информации, представленных в форме естественного языка.

Чтобы выполнить эту задачу, необходимо произвести синтаксическую обработку текста. Существует несколько подходов к реализации синтаксических анализаторов. Выбор использованного метода подробно описан в пункте 1.3 данной работы. Метод предполагает явное разделение процесса анализа текста на следующие модули: морфологический анализ, синтаксический анализ, семантический анализ и прагматический анализ.

В данной работе рассматривается процесс синтаксического анализа с использованием готовых решений морфологического анализа.

В процессе исследования и разработки, следуя источнику [13], были выделены следующие понятия, необходимые для реализации выбранного метода синтаксического анализа:

* Синтаксический анализатор (СА) – программа, которая производит синтаксический анализ текста без его сколь угодно простого семантического анализа;
* Граммема – компонент грамматической категории, представляющий собой по своему значению видовое понятие по отношению к значению грамматической категории как понятие родовому;
* Часть предложения – простое предложение с одной грамматической основой в составе сложного, исходного предложения текста;
* Синтаксическая группа (СГ) – группа синтаксически связанных слов, среди которых выделено главное, которое определяет роль группы;
* Входная СГ – СГ, рассматриваемая как один двух из входов правила;
* Правило – принцип обработки двух входных синтаксических групп, условий согласования и преобразования двух СГ в результирующую СГ;
* Тип входной СГ – константа, которая определяет синтаксическую роль (подчиненная или главная) двух входящих в правило СГ во время его применения.
* Вид СГ – характеристика каждой СГ, которая определяет набор правил, применимых к данной СГ.
  + 1. Перечень процессов, подлежащих автоматизации

При разработке модуля морфологического анализа текста следующие процессы подлежат автоматизации:

* Фрагментация текста на абзацы, абзацев – на предложения, а предложений – на лексемы (слова и знаки препинания);
* Формирование словаря, сохраняющего информацию о строении текста и морфологическую информацию о словах, то есть информацию о вариантах морфологического разбора каждой лексемы предложения.

При разработке модуля синтаксического анализа текста следующие процессы подлежат автоматизации:

* Формирование списка союзов каждого предложения;
* Фрагментация сложного предложения на простые с учётом информации о союзах и знаках препинания;
* Поиск грамматической основы каждой части предложения;
* Поиск вариантов цепочек словоформ следуя результатам морфологического разбора для лексем каждой части предложения;
* Применение правил к очередному варианту части предложения;
* Запоминание успешных попыток анализа части предложения;

При разработке модуля формирования метаграфа, как результата синтаксического разбора предложения:

* Формирование списка метаграфов на основе информации об абзацах, и положении предложений текста в абзацах, а абзацев в тексте.
  + 1. **Выбор критериев качества**

Проблема понимания компьютером естественного языка широко распространена, многие прикладные системы используют в процессе работы синтаксические анализаторы, принципы работы которых разобраны более-менее хорошо. Этим обусловлено наличие аналогов. Для сравнения выбраны системы Solarix [15] и AOT [16]. Варианты синтаксических анализаторов представлены в таблице 1.

**Таблица 1. Варианты синтаксических анализаторов**

| **№ п/п** | **Название варианта** | **Обозначение** |
| --- | --- | --- |
| 1. | Solarix | В1 |
| 2. | AOT | В2 |
| 4. | Разрабатываемый модуль | В3 |

Так как мы уже выбрали алгоритмы разрабатываемого СА, мы можем сравнить его с другими, готовыми СА. Для разрабатываемого программного изделия приоритетными являются критерии качества, приведенные в таблице 2.

**Таблица 2. Критерии сравнения аналогов и прототипов**

| **Критерий** | **Описание** | **Обозначение** |
| --- | --- | --- |
| Наличие графического интерфейса | Критерий показывает, насколько пользователю будет удобно рассматривать и анализировать результаты анализа по картинке на экране | К1 |
| Стоимость | Стоимость программного модуля для его использования в собственной программе | К2 |
| Платформа | Платформа для которой предназначена система | К3 |
| Сложность эксплуатации | Сложность использования модуля в рамках собственной программы | К4 |
| Форматы выходных данных | Качество формата данных, генерируемых системой | К5 |
| Открытый исходный код | Доступность исходного кода программы, предоставляющая возможность собственной модификации | К6 |

Показатели различных вариантов по данным критериям представлены в таблице 3.

**Таблица 3. Показатели различных вариантов по данным критериям**

| **Варианты** | **К1** | **К2** | **К3** | **К4** | **К5** | **К6** |
| --- | --- | --- | --- | --- | --- | --- |
| В1 | 3 | 1 | 5 | 4 | 3 | 4 |
| В2 | 2 | 1 | 3 | 4 | 3 | 5 |
| В3 | 4 | 5 | 5 | 3 | 5 | 5 |

* + 1. Анализ аналогов и прототипов

Для анализа выбран метод базового критерия. В связи с этим методом обозначенным выше критериям назначены весовые коэффициенты. Результаты приведены ниже в таблице 4.

**Таблица 4. Проранжированные критерии качества**

| **№**  **п/п** | **Название критерия** | **Весовой коэффициент** |
| --- | --- | --- |
| 1. | Наличие графического интерфейса | 2 α |
| 2. | Стоимость | 5 α |
| 3. | Платформа | 6 α |
| 4. | Сложность эксплуатации | 3 α |
| 5. | Форматы выходных данных | 4 α |
| 6. | Открытый исходный код | 5 α |

Определим величину α, рассчитав значения весовых коэффициентов по формуле 1, используемую благодаря источнику [8].

, (1)

где *п* – общее количество весовых коэффициентов; *ai*  – весовые коэффициенты.

Подставив значения в формулу 1 получаем:

В таблице 5 отображены результата расчёта весов методом базового критерия.

**Таблица 5. Расчет весов методом базового критерия**

| **Критерий** | **Коэффициент важности (ki)** | **Вес (αi)** |
| --- | --- | --- |
| К1 | 2 | 0.08 |
| К2 | 5 | 0,2 |
| К3 | 6 | 0,24 |
| К4 | 3 | 0,12 |
| К5 | 4 | 0,16 |
| К6 | 5 | 0,2 |

Для вычисления локальных критериев необходимо произвести нормировку значений критериев.

Нормированные значения приведены в таблице 6.

**Таблица 6. Нормированные значения локальных критериев**

| **Критерий** | **Вес (αi)** | **В1** | **В2** | **В3** |
| --- | --- | --- | --- | --- |
| К1 | 0,08 | 0,75 | 0,5 | 1 |
| К2 | 0,2 | 0,2 | 0,2 | 1 |
| К3 | 0,24 | 1 | 0,6 | 1 |
| К4 | 0,12 | 1 | 1 | 0,75 |
| К5 | 0,16 | 0,6 | 0,6 | 1 |
| К6 | 0,2 | 0,8 | 1 | 1 |
| Итог | | 0,71225 | 0,64 | 0,97 |

При этом итоговые значения рассчитаны по формуле 2.

, (2)

Лучший вариант выберем по формуле 3.

, (3)

где *m* – общее количество вариантов.

Таким образом получили, что разрабатываемый модуль имеет предпочтение над имеющимися аналогами. Расчет показывает, что есть основания для разработки данного программного модуля.

* 1. Анализ существующих алгоритмов и внедрение отличительных особенностей

Для построения архитектуры синтаксического анализатора, которая отвечает всем требованиям, указанным в индивидуальном задании, необходимо проанализировать все решения, которые используются в разработке синтаксических анализаторов.

Детальный анализ приведён в трудах [1] и [2], а также работе [9]. В них говорится о различных стратегиях при анализе синтаксических конструкций с разной степенью детализации.

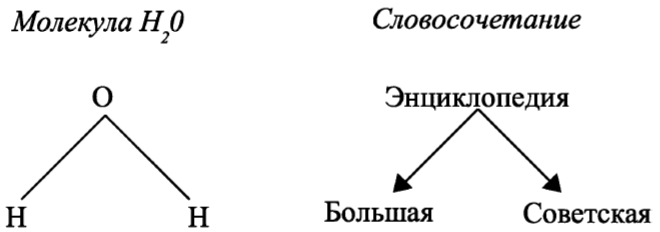
Наиболее четкие границы есть между тремя подходами к описанию синтаксиса:

1) Грамматики зависимостей;

2) Грамматики непосредственных составляющих;

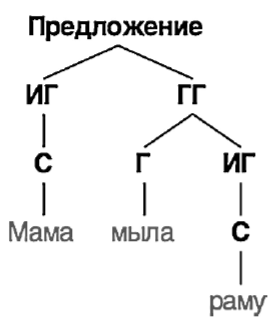
3) Комбинированные теории, например, теория синтаксических групп.

Основной принцип подхода грамматики зависимостей заключается в том, что каждое слово предложения составляет иерархию. Каждое слово является своего рода атомом в молекуле. Это означает что само предложение в результате анализа представляется в виде дерева зависимостей или, другими словами, графа зависимостей. Таким образом синтаксические связи устанавливают между словами отношения зависимости. Одно из двух слов является главным, а другое зависимым. Например, в словосочетании «Большая Советская Энциклопедия» имеется две связи, изображённые на рисунке ниже (см. рис.1). Эти связи образуют отношения зависимости: в обеих связях главной является словоформа энциклопедия. Словоформы «Большая» и «Советская» оказываются зависимыми.

Рис. 1. Строение взаимосвязей в «грамматике зависимостей»

Как мы видим, узким местом этого подхода является невозможность построения какой-либо другой связи, кроме подчинительной.

Непосредственные составляющие (НС, фразы) – на рисунке ниже (см. рис.2) приведён пример структуры предложения "Мама мыла раму". У этого предложения имеются НС: "мама" и "мыла раму". В свою очередь, "мыла раму" имеет составляющие "мыла" и "раму", то есть эти НС не входят непосредственно в предложение, а являются компонентами его более-менее крупной части. По этой схеме ни одна словоформа не будет являться НС целого предложения, но будет являться НС по отношению к тем или иным его составляющим.

  
Рис. 2. Грамматика непосредственных составляющих.

По данной схеме предложение состоит из одной или нескольких НС. Каким образом данные НС-грамматика оперирует такими понятиями, как классы НС, например, предложения, именная группа или глагольное группа. Однако названия этих классов никак не обосновываются частями речи тех словоформ, которые входят в его состав.

Именно в подходе непосредственных составляющих появляется такое понятие как восходящие алгоритмы или нисходящие алгоритмы. Подробное описание восходящих и нисходящих алгоритмов описано в работе [2]. Для данной работы «восходящий» способ был выбран именно потому, что он отличается относительной простотой в плане построения правил нахождения взаимосвязи между различными НС. Основной принцип такого вида алгоритма заключается в том, что в предложении ищется наименее значимая с синтаксической точки зрения словоформа, и она соединяется с более значимой по специальным правилам, например, "частица+глагол" или "предлог+существительное". Таким образом мы получаем возможность отсечения неверных вариантов на том этапе разбора предложения, когда мы не обработали все словоформы и не порождаем множество заведомо неверных вариантов. Таким образом мы можем экономить время работы и ресурсы компьютера. Одним из основных минусов такого вида алгоритмов является порождение так называемых эллипсисов или рекурсивных грамматик. Эллипсис – возникновение пропусков некоторых слов в предложении. В таком случае мы должны восстанавливать потерянные слова из контекста. Примеры предложений с эллипсисом: "Он нравится Маше, а она – ему".

Комбинированные алгоритмы нивелируют минусы друг друга за счёт того, что в них может реализовываться разный подход к использованию восходящего алгоритма и нисходящего алгоритма. Например, сначала может быть разобранной восходящем алгоритмом именная группа подлежащего в предложении, а нисходящем – группа сказуемого. За счет использования принципов обоих подходов комбинированные алгоритмы находятся по середине между восходящими и нисходящими алгоритмами в плане производительности и скорости. Однако они впереди нисходящих и восходящих алгоритмов в плане точности разбора предложений, поскольку они могут реализовывать разбор эллипсисов, рекурсивных грамматик или других сложных случаев предложений, которые недоступны для разбора обычным восходящим или нисходящим алгоритмам.

На рисунке 3 изображены общие схемы работы трех описанных выше алгоритмов. Число рядом с НС показывает порядок нахождения зависимостей.

  
Рисунок 3. Порядок работы нисходящих, восходящих и комбинированных алгоритмов

Еще одним минусом восходящих и нисходящих алгоритмов НС является их неспособность анализировать те НС, которые разделены другими НС. Для устранения недостатка подобных алгоритмов А. В. Гладким была разработана теория синтаксических групп (ТСГ), которая реализуется в разрабатываемой системе.

Этот подход является представителем комбинированных теорий анализа предложения [2]. Он предполагает использование восходящего алгоритма, но уже применительно не к непосредственным составляющим, а к синтаксическим группам (СГ). ТСГ допускает включение в структуру предложения тех групп, которые участвуют в отношении зависимости «целиком», а не посредством одной словоформы. Таким образом этот подход позволяет легко обрабатывать разрывные составляющие.

Независимо от выбранного подхода главная проблема синтаксического анализа связана с неоднозначностью языковых единиц. Помимо очевидной многозначности слов (лексической неоднозначности), есть и менее очевидная многозначность грамматических форм (морфологическая неоднозначность): ср., например, например, числи тельное три и форму повелительного наклонения единственного числа глагола тереть, или форму именительного падежа слова стекло и форму прошедшего времени, единственного числа, среднего рода глагола стекать. Морфологическая неоднозначность может показаться редким явлением, но только потому, что она редко бывает очевидной.

Больше всего проблем вызывает, однако, не морфологическая, а синтаксическая неоднозначность. Из-за морфологической неоднозначности одна и та же фраза может иметь несколько разных пониманий, которые соответствуют разным структурам. Более того, одна и та же фраза может быть понята по-разному даже при однозначности всех её словоформ («Вася встретил Петю в коридоре» и «Вася встретил Петю в костюме»). В действительности, обе фразы имеют, по крайней мере, три возможные синтаксических структуры. Им соответствует три разные ситуации. Для первой фразы они таковы:

* Вася встретил Петю, и это произошло в коридоре;
* Вася встретил Петю, одетого в коридор;
* Вася, одетый в коридор, встретил Петю.

Чтобы найти верный вариант разбора целого предложения, можно попытаться перебрать все комбинации вариантов разбора его частей. Проблема в том, что при компьютерном анализе это приводит к так называемому комбинаторному взрыву.

Как уже видно из примера даже у самого короткого предложения может быть несколько интерпретаций на уровне синтаксиса и лексики. Поэтому, если стоит цель выдать единственно верный вариант синтаксического разбора, необходимо решить задачу, по сложности сопоставимую с задачей коммивояжера. Однако поиск верного решения среди нескольких вариантов –задача семантического характера, поэтому ее решение идет в разрез с целью данного проекта.

Выбранный алгоритм СА предполагает, что каждая СГ части предложения равноправна любой другой. Исключения составляют слова грамматической основы. Если слово – часть грамматической основы, то оно не может быть подчинено, а слова, подчиняемые словам грамматической основы, присоединяются с ней на предпоследнем шаге. Таким образом любое слово, кроме грамматической основы в теории может быть связано с любым другим словом предложения. Само предложение считается проанализированным, когда каждое слово, кроме грамматической основы стало подчиненным. Тогда грамматическая основа связывается в предложение одним из трёх уникальных правил: только «Подлежащее», только «Сказуемое» и «Подлежащее+Сказуемое или Сказуемое+Подлежащее».

Подробное описание правил представлено в пункте 2.5 работы. Подробное описание структуры программного изделия и конкретных алгоритмов представлено в пункте 2.4 и 2.7 работы.

1. КОНСТРУКТОРСКО-ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ
   1. Выбор программных средств

Модуль кроссплатформенный, поэтому для использующих его ПК нет ограничений по предустановленным на ним ОС.

Программа состоит из нескольких функций, работающих над разными задачами, порядок выполнения которых определяется конкретным подходом, оба которых описаны выше.

Для использования любого из этих двух принципов нужно понимать, какие задачи ставит перед собой проект. Но независимо от этих задач предварительным этапом работы любого синтаксического анализатора (СА) является токенизация текста на предложения, предложений – на лексемы: слова и знаки препинания. Для этого необходимо использовать наиболее точные библиотечные программные средства, написанные на языке Python 3:

* Разделение на предложения выполняются с помощью библиотеки ru\_sent\_tokenize. Она предназначена и разработана для разделения русского текста на предложения с учётом специфики русского языка.
* Для разделения предложений на слова используется библиотека NLTK для русского языка с предварительной загрузкой соответствующих модулей и словарей русского языка.
* Для реализации морфологического разбора лексем предложения используется библиотека pymorphy2 [7]. Она написана для морфологического анализа русского языка с использованием открытых материалов корпуса словарей русского языка OpenCorpora, обогащением которого может заняться любой желающий.

NLTK – библиотека, удачно подходящая для разбора английского языка. Предложения русского языка она разделяет с точностью 95,5 процента. ru\_sent\_tokenize справляется с этой же задачей в 99 и более процентах случаев, а также скорость выполнения задачи более, чем в полтора раза выше. Именно поэтому ru\_sent\_tokenize был выбран для разделения текста на предложения. NLTK, настроенный на морфологический анализ русского языка отлично справляется с токенизацией по словам.

В данном проекте база данных отвечает исключительно за хранение данных о правилах, и необходима лишь для того, чтобы хранить информацию о разобранных словах предложения. То есть каждый успешный результат анализа предложения записывается в Базу Данных.

Выбрана СУБД MySQL. Она оптимизирована под использование небольших приложений. В будущем с расширением возможностей СА не возрастет нагрузка на БД, нет нужды выбирать более быстрые, надежные базы данных, поскольку в ней не хранятся данные приложения или пользователей.

* 1. Выбор аппаратных средств

Модуль планируется для использования в других системах в качестве компонента.

Минимальные необходимые характеристики:

* процессор Intel Core i3 или более мощный (или совместимый с ним);
* рекомендуемый объем оперативной памяти 2048 мегабайт (МБ);
* минимальный объем 1024 МБ;
* 100 MB свободного места на жестком диске для системы и от 500 МБ до 1 ГБ для приложения, базы данных и демонстрационных файлов;
* монитор;
* клавиатура;
* мышь или совместимое указывающее устройство.
  1. Выбор языка разработки

Алгоритмическим задачам, стоящим перед проектом, удовлетворяет язык программирования Python 3. Выбран он был по следующим причинам:

* Этот язык прост;
* Реализация алгоритмов не требует множества строк кода, то есть лишнего времени на написание тонких моментов проекта;
* Онлегок и прост в использовании, интеграции в многомодульный проект;
* Язык ориентирован на решение задач общего направления, то есть не ограничивает возможности реализации СА;
* Широкая распространенность и большое количество стандартных и сторонних библиотек, которые могут оптимально упростить и ускорить разработку;
* Все перечисленные в пункте 2.1 средства используют именно его;
* Структура хранения данных идеально походит под временное хранение информации об анализе текста в течение работы программы.

Этот ЯП, однако, сложно использовать для параллельных вычислений. Куда лучше для анализа нескольких предложений сразу подходит, например, функциональный ЯП Golang, однако используемые сторонние средства не поддерживают этот ЯП. В случае его использования как основного языка разработки потребуется написать дополнительный модуль, который обеспечит взаимодействие кода на Python 3 и на Golang.

* 1. Структура программного изделия

Данный программный продукт состоит из трех логически разделенных модулей:

* Морфологический анализатор;
* Синтаксический анализатор;
* Формирование метаграфа.

Морфологический анализатор включает в себя фрагментацию текста, основная задача которого состоит в выделении абзацев в составе текста, предложений в составе абзацев, слов и знаков (лексем) препинания в составе предложения, а также в морфологическом анализе полученных лексем. На выходе данного модуля получаются фрагменты, которые готовы для синтаксического анализа.

В связи с выбранной целью было выбран подход, игнорирующий так называемый «комбинаторный взрыв». Данная проблема стоит перед всеми синтаксическими анализаторами.

Каждое слово имеет несколько допустимых правилами форм, называемых словоформами. Каждая словоформа имеет одно и несколько вариантов набора морфологических характеристик. Например, слово «её» имеет само по себе более 20 вариантов разбора. Поэтому разбор любого предложения, даже абсолютно правильный с точки зрения правил русского языка, влечёт за собой размножение вариантов разбора предложения (предложение из 7 слов может быть разобрано, например, 10-ю и более разными способами).

Чтобы решить эту проблему существующие системы пытаются использовать внедрение элементов семантического анализатора на синтаксический. Это влечёт за собой многократное возрастание сложности и временной дороговизны создания анализатора, нивелирование разницы между этими двумя этапами разбора. В данном проекте используется другая схема: морфология, синтаксис, семантика и прагматика текста полностью разделяется.

Это позволяет не только абстрагировать конкретный уровень от проблем другого, но и сосредоточиться на тонкостях проблем конкретного уровня. Например, все мы знаем, что предлоги не имеют морфологических характеристик, которые позволяют использовать их в сочетании с другими словами. Однако, мы все знаем, что, например, предлог «по» употребляется только со словами в дательном падеже. Таким образом, «обработка предлогов» позволит сократить количество вариантов морфологического разбора слов в 2-3 раза, а для неизменяемых слов (например, «пальто») ровно в 6 раз. Этот этап может предшествовать синтаксическому этапу, этапу поиска союзов предложения и грамматической основы.

После получения морфологической информации (с помощью библиотек ru\_sent\_tokenize, NLTK и pymorphy2) о составе предложения начинается этап синтаксического анализа, который проходит в два этапа (подробное описание в пункте 2.7.2):

1. Производится разделение сложных предложений на простые;

2) Анализ каждой части предложения с учётом всех возможных вариантов грамматической основы данной части, с учётом приоритета правил и т.д.;

Результат анализа предложения сохраняется в объекте метаграфа, исходя из следующих представлений (подробное описание в пункте 2.7.3):

* Одно предложение – один метаграф;
* Одна часть – одна метавершина;
* Одно слово – одна вершина;
* Одна связь простых предложений в составе сложного – одно ребро метаграфа;
* Одна связь между 2-мя словами – одно ребро метавершины.
  1. Проектирование структуры правил

В связи с выбранным методом синтаксического анализа, правила должны определять взаимодействие двух синтаксических групп без участия семантики текста. Таким образом, предложенные в работах [4] и [5] системы правил не подходят разрабатываемой системе. Так как алгоритм анализа можно охарактеризовать как восходящий, каждое правило имеет приоритет применения. Правила хранятся в БД, схема которой описывается в пункте 2.6.

Для работы правил необходима информация о двух СГ, составляющих объекты применения правила, о том, какая из двух СГ – главная, а какая – подчиненная. Также необходима информация о самом приоритете правила и о флагах согласования двух СГ по роду, числу и падежу.

Основной задачей в процессе изучения предметной области является выявления зависимостей и закономерностей формирования синтаксических конструкций.

В основе синтаксического анализа выбранного подхода лежит построение синтаксических групп на одном морфологическом варианте разбора части предложения. Частью называется простое предложение, которое содержит только одну грамматическую основу. Перед непосредственно анализом одного фрагмента производится фрагментация сложного предложения на простые.

Существует два типа синтаксического анализа:

* Нисходящий синтаксический анализ. При нисходящем анализе делается предположение о строении предложения и проверяется путем сравнения с исходным. Таким образом, движение идет от корня к листьям.
* Восходящий синтаксический анализ. При восходящем синтаксическом анализе с помощью набора правил единицы предложения объединяются в группы, пока в предложении не останется одна группа, которая будет соответствовать корню – грамматической основе.

Для того, чтобы унифицировать применение различных правил необходимо заранее разработать структуру, которая будет учитывать все возможные варианты и разработать алгоритм применения правил, соответствующих данной структуре. Слово рассматривается исключительно как СГ, и не важно, сколько слов она подчинила и подчинила ли. Каждое словосочетание (СГ) выполняет определенную синтаксическую функцию, которая определяется главным словом в данном словосочетании. Словосочетания могут объединятся в более крупные, сохраняя роль главного или приобретая новую.

Синтаксическая группа (СГ) может обладать синтаксической ролью, также в ней определено главное слово, которое определяет роль словосочетания. В таком случае и слова, и словосочетания будут является синтаксическими группами.

На вход алгоритма подаются две синтаксические группы, по которым делается запрос в таблицу БД «Правило». Алгоритм определяет уникальное правило сочетания именно этих двух СГ. Далее по приоритету выбранного правила определяется, необходимо ли применять данное правило (подробное описание алгоритма анализа с применением правил приводится в пункте 2.7.2). Затем правило применяется, и из двух СГ остается доступной для анализа только одна из них: главная.

Всего у входных СГ может быть два типа: главная и подчиненная.

В русском языке есть правила сочетания слов, как правило, не закрепленные документально. В русском языке нет правил порядка следования членов предложения в связи с чем моментально и многократно возрастает сложность нахождения отношений между словами. Однако с этим ничего не поделать, поэтому необходимо просто обработать в два-три раза больше вариантов сочетаний слов, чем для других языков.

Отношения СГ, по сути, отношения зависимости с некоторыми оговорками, поскольку возможно нахождение однородных членов и даже однородных простых предложений в составе сложного. Однако правила устанавливают именно отношения зависимости, выстраивая иерархию слов предложения.

Само правило хранит свой приоритет для определения порядка применения правил. Также правило содержит информации о необходимом согласовании входных групп, которые к нему относятся. Это позволяет обеспечивать требования согласования в различных правилах по роду, числу и падежу.

Все правила однозначно определяют итоговый член предложения, который реализует подчиненная СГ данного словосочетания.

Как говорилось ранее у каждого правила есть приоритет. Этот приоритет соответствует порядку построения групп: от меньших к большим. Например, сначала надо построить группы нареч-ПРИЛ, а потом прил-СУЩ, чтобы построить структуру на отрезке "очень красивый человек": прил-СУЩ (нареч-ПРИЛ (очень красивый), человек) Каждое правило применяется к каждому слову входного отрезка слева направо.

Ниже представлены примеры правил по каждому из 9 приоритетов и их описания в рамках входных синтаксических групп. Также здесь приведено правило, объединяющее грамматическую основу в предложение.

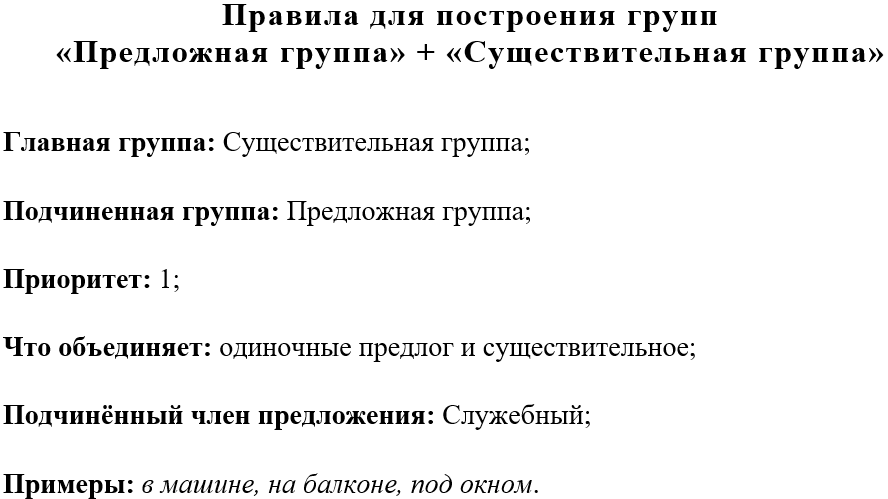
Чем больше номер приоритета, тем больше вероятность нарушения связей предложения при простановке для этого правила меньшего номера. Например, есть словосочетание «за чем-то очень приятные слова». Это сочетание «Предлог + Местоимение + Наречие + Прилагательное + Существительное». Мы понимаем, что возможны сочетания «Местоимение + Наречие», однако это менее вероятное сочетание, чем присутствующее в данном примере «Местоимение + Прилагательное» и «Наречие + Прилагательное», поэтому его приоритет становится больше, чем приоритет того же «Наречие + Прилагательное».

Также мы видим, что Предлог стоит через три слова от своего главного – Существительного, поэтому сочетания предлога и наречия менее приоритетны, предлога и местоимения или прилагательного более приоритетны, а предлога и существительного – самые приоритетные и имеют приоритет, равный 1.

Так как разбор идет слева направо, не возникнет конфликтных ситуаций внутри цепочки СГ, где главная группа следует за подчиненной. Во всех остальных ситуациях конфликт может возникнуть и эти потенциальные конфликты решаются именно ведением согласования и приоритетов.

Правила построены так, чтобы любое слово, даже грамматическая основа, сохранилось в БД в таблице «Подчиненная СГ». Таким образом предложение станет восстановимым в своем изначальном виде благодаря лишь данным в одной таблице БД.

На рисунке 4 приведена структура правила «предл+СУЩ».

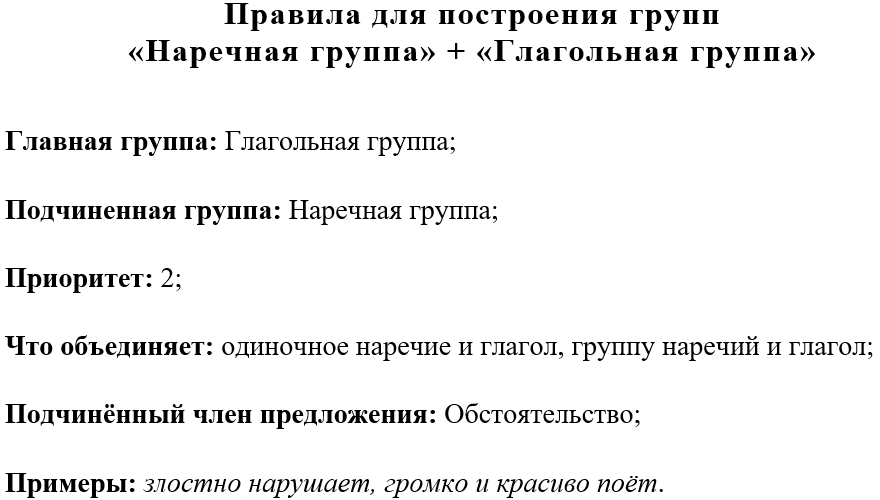
  
Рисунок 4. Структура правила «предл+СУЩ»

Согласование по падежу, числу и роду не нужно, поскольку Предлог не имеет этих характеристик.

Это правило имеет первый приоритет. Это означает, что оно обработается первым среди всех прочих.

Существительная группа – главная, поэтому после применения правила она останется в предложении, а группа предлога станет считаться обработанной и в дальнейшем поиске двух СГ для применения правила использоваться не будет. Поэтому после применения правила данная связка слов станет СГ вида Существительная группа, а предлог станет Служебным членом предложения.

На рисунке 5 приведена структура правила «нареч+ГЛ».

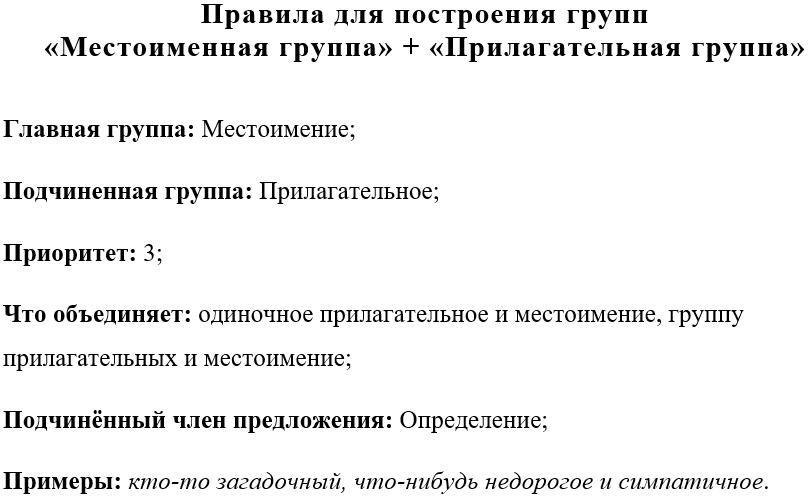
  
Рисунок 5. Структура правила «нареч+ГЛ»

Согласование по падежу, числу и роду не нужно, поскольку наречие не обладает этими характеристиками.

В принципе, данному конкретному правилу можно поставить и первый приоритет, но так как первый приоритет предназначен для подчинения служебных частей речи и элементов составного сказуемого, имеет смысл логически отделить от них все остальные правила.

Глагольная группа – главная, поэтому после применения правила она останется в предложении, а группа наречия станет считаться обработанной и в дальнейшем поиске двух СГ для применения правила использоваться не будет. Поэтому после применения правила данная связка слов станет СГ вида Глагольная группа, а наречие станет Обстоятельством.

На рисунке 6 приведена структура правила «МЕСТ+прил».

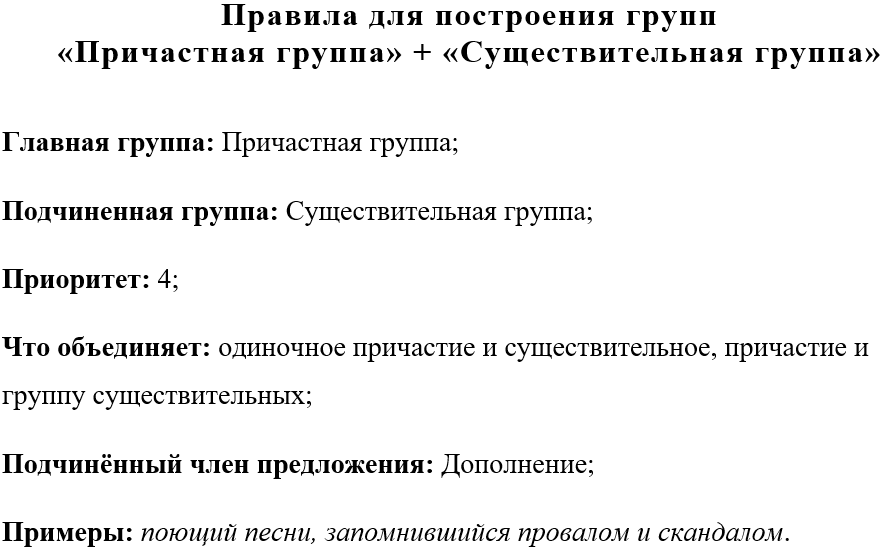
  
Рисунок 6. Структура правила «МЕСТ+прил»

Согласование по падежу, числу и роду нужно, поскольку прилагательное – описание какого-либо объекта, выраженного местоимением.

У данного правила 3-й приоритет, поскольку такое словосочетание должно обрабатываться после «мест+ПРИЛ», например, в сочетании «чем-то приятные слова».

Местоименна группа – главная, поэтому после применения правила она останется в предложении, а группа прилагательного станет считаться обработанной и в дальнейшем поиске двух СГ для применения правила использоваться не будет. Поэтому после применения правила данная связка слов станет СГ вида Местоименная группа, а прилагательное станет Определением.

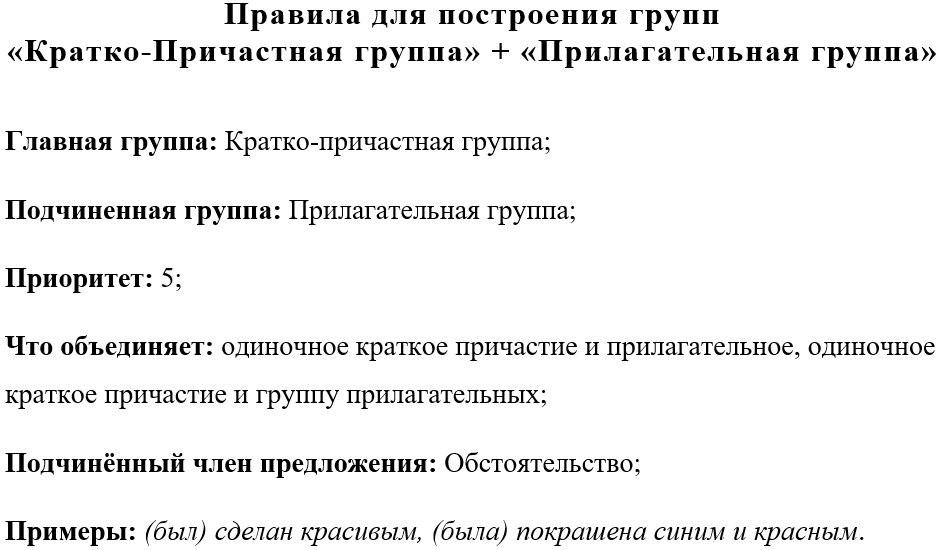
На рисунке 7 приведена структура правила «ПРИЧ+сущ».

  
Рисунок 7. Структура правила «ПРИЧ+сущ»

Согласование по падежу, числу и роду не нужно, поскольку существительное может являться дополнением, которое по определению может иметь любой падеж, число и род.

Причастная группа – главная, поэтому после применения правила она останется в предложении, а группа существительного станет считаться обработанной и в дальнейшем поиске двух СГ для применения правила использоваться не будет. Поэтому после применения правила данная связка слов станет СГ вида Причастная группа, а наречие станет Дополнением.

На рисунке 8 приведена структура правила «КР.ПРИЧ+прил».

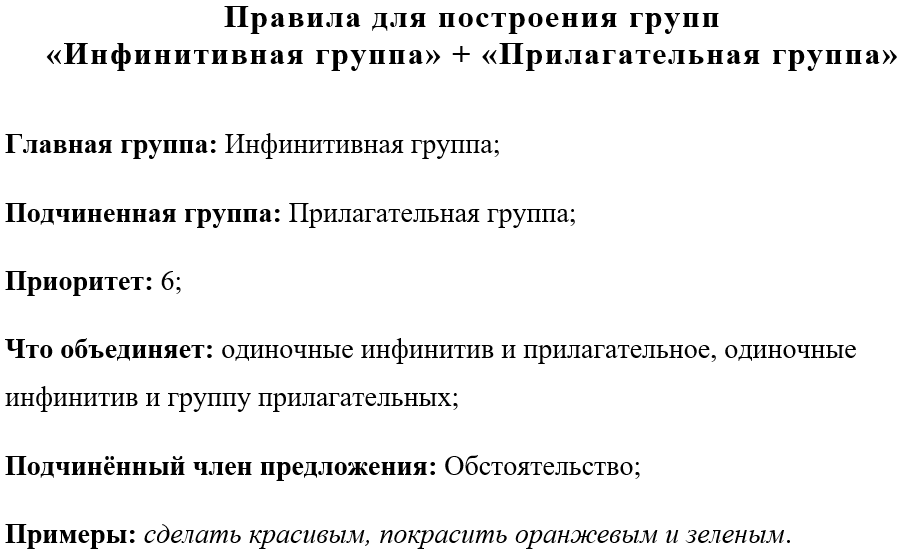
  
Рисунок 8. Структура правила «КР.ПРИЧ+прил»

Необходимо согласование по числу и роду, поскольку наречие не обладает этими характеристиками.

Данному правилу поставлен 4-й приоритет, потому что его применение куда менее вероятно, чем сочетание «прил+СГ», но оно имеет место быть обстоятельством для составного глагола или даже когда сказуемое выражено простым кратким причастием.

Группа краткого причастия – главная, поэтому после применения правила она останется в предложении, а группа прилагательного станет считаться обработанной и в дальнейшем поиске двух СГ для применения правила использоваться не будет. Поэтому после применения правила данная связка слов станет СГ вида Кратко-причастная группа, а прилагательное станет Обстоятельством.

На рисунке 9 приведена структура правила «ИНФ+прил».

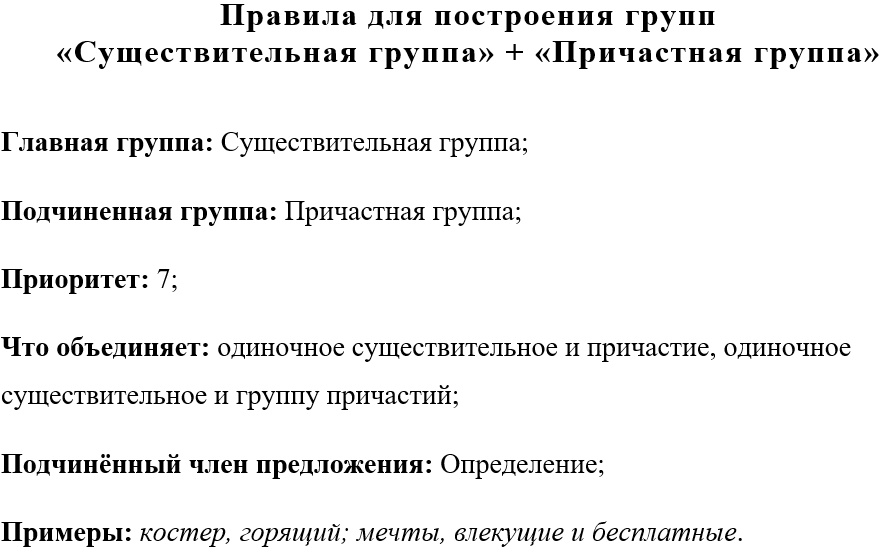
  
Рисунок 9. Структура правила «ИНФ+прил»

Согласование по падежу, числу и роду не нужно, поскольку инфинитив не обладает этими характеристиками.

Инфинитив в предложении является частью грамматической основы, поэтому его разбор обычно происходит одним из последних. Прилагательное обычно подчиняется различным Именным группам, поэтому сначала мы должны рассмотреть именно эти сочетания, а потом уже рассматривать данное.

Инфинитивная группа – главная, поэтому после применения правила она останется в предложении, а группа прилагательного станет считаться обработанной и в дальнейшем поиске двух СГ для применения правила использоваться не будет. Поэтому после применения правила данная связка слов станет СГ вида Инфинитивная группа, а прилагательное станет Обстоятельством.

На рисунке 10 приведена структура правила «СУЩ+прич».

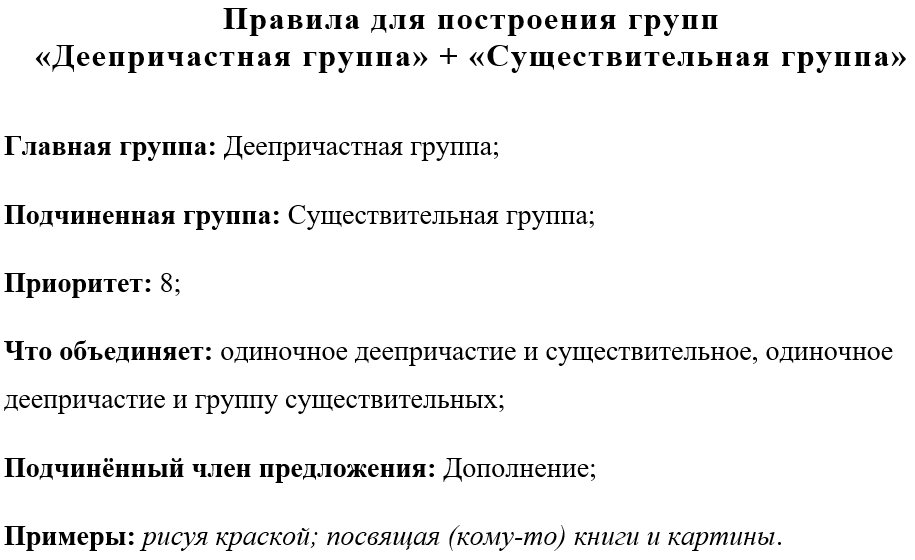
  
Рисунок 10. Структура правила «СУЩ+прич»

Согласование по падежу, числу и роду нужно, поскольку причастный оборот всегда согласуется с управляющим словом.

Сначала необходимо обработка всех возможных зависимых слов для причастия, поскольку оно участвует лишь в причастном обороте.

Существительная группа – главная, поэтому после применения правила она останется в предложении, а группа причастия станет считаться обработанной и в дальнейшем поиске двух СГ для применения правила использоваться не будет. Поэтому после применения правила данная связка слов станет СГ вида Существительная группа, а причастие станет Определением.

На рисунке 11 приведена структура правила «ДЕЕПР+сущ».

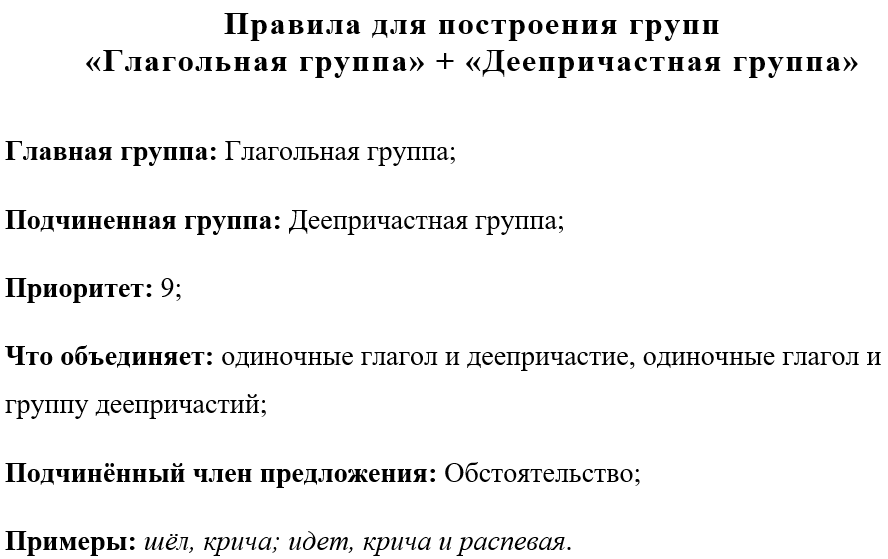
  
Рисунок 11. Структура правила «ДЕЕПР+сущ»

Согласование по падежу, числу и роду не нужно, поскольку деепричастие не обладает этими характеристиками.

Деепричастный оборот – явление не однозначное. Единственное, что можно сказать точно, так это то, что он всегда обособляется запятыми, что отделяет его разбор от остальной части предложения. Это сочетание имеет поздний приоритет, поскольку до того, как подчинять какое-либо существительное к деепричастию, нужно подчинить ему другие слова по правилам более низкого приоритета.

Деепричастная группа – главная, поэтому после применения правила она останется в предложении, а группа существительного станет считаться обработанной и в дальнейшем поиске двух СГ для применения правила использоваться не будет. Поэтому после применения правила данная связка слов станет СГ вида Деепричастная группа, а существительное станет дополнением.

На рисунке 12 приведена структура правила «ГЛ+деепр».

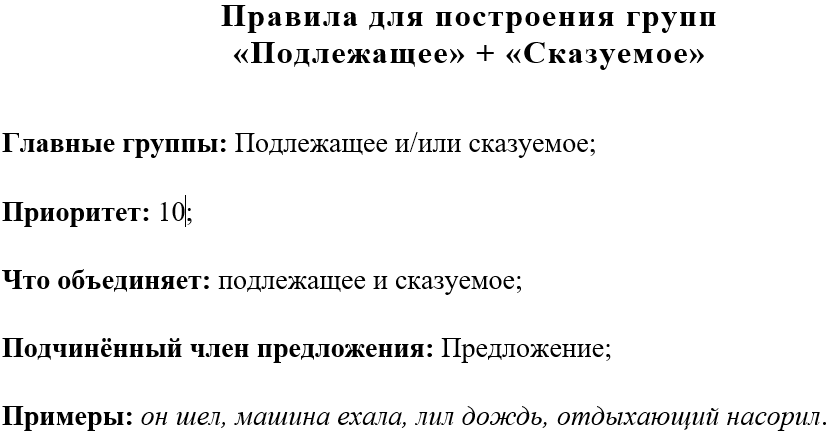
  
Рисунок 12. Структура правила «ГЛ+деепр»

Согласование по падежу, числу и роду не нужно, поскольку деепричастие не обладает этими характеристиками.

Подчинение деепричастного оборота глаголу (сказуемому), от которого оно зависит по правилам русского языка.

Глагольная группа – главная, поэтому после применения правила она останется в предложении, а группа деепричастия станет считаться обработанной и в дальнейшем поиске двух СГ для применения правила использоваться не будет. Поэтому после применения правила данная связка слов станет СГ вида Глагольная группа, а деепричастие станет Обстоятельством.

На рисунке 13 приведена структура правила «ПОДЛ+СКАЗ/СКАЗ+ПОДЛ».

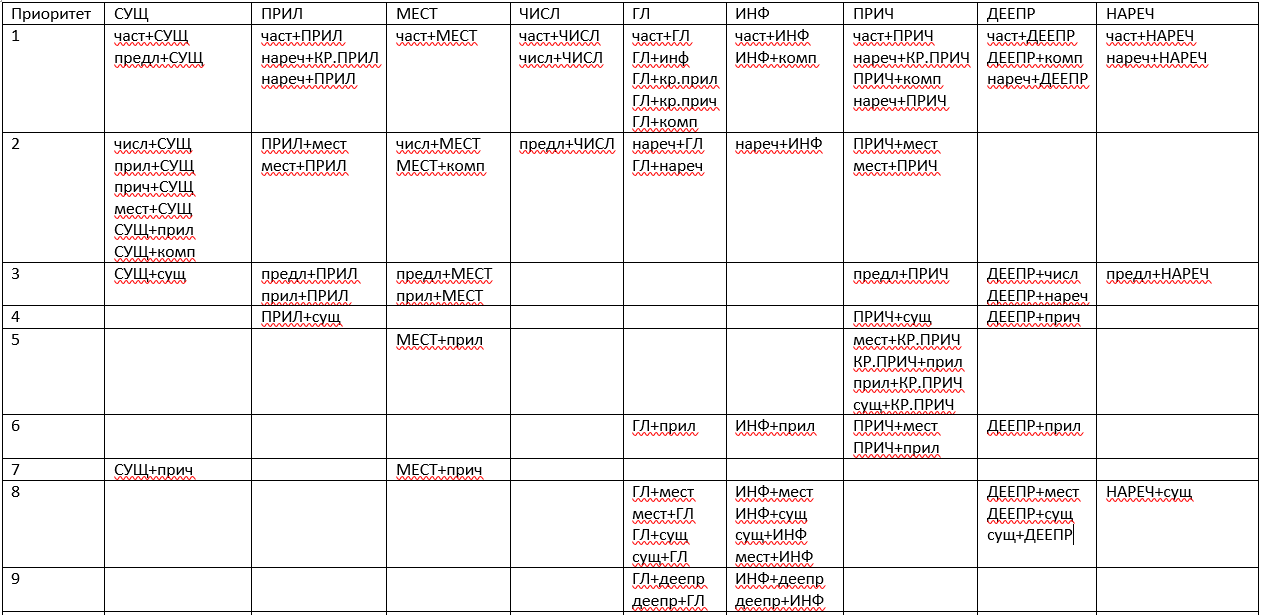
  
Рисунок 13. Структура правила «ПОДЛ+СКАЗ/СКАЗ+ПОДЛ»

Согласование по падежу, числу и роду не нужно, поскольку эту сочетаемость уже проверили на предыдущей стадии работы алгоритма.

У данного правила последний приоритет, поскольку оно должно применяться только после того, как подчинены все остальные слова, иначе есть риск того, что какое-то из слов основы подчиниться слову, не входящему в основу и предложение останется неразобранным.

Главной группы тут нет, поскольку обе группы: и Подлежащее, и Сказуемое – самые главные слова предложения. Это правило необходимо для сохранения в БД самой грамматической основы, поскольку именно этих слов может просто не оказаться в числе Главных СГ предложения. Таким образом грамматическая основа подчиняется абстрактному предложению и любое слово предложение сохраняется в БД в качестве подчиненной СГ и предложение становится полностью восстановимым из данных БД.

Приоритеты для каждой пары СГ создавались вручную, но опытные испытание показали абсолютную точность самых популярных сочетаний слов. Ниже, в рисунке 13, представлены все используемые сочетания. Первая колонка – номер приоритета. Слева направо в заглавии колонок стоит обозначение той СГ, которая в сочетаниях данного столбца является главной.

  
Рисунок 13. Используемые словосочетания.

где СУЩ – существительное, ПРИЛ – прилагательное, ГЛ – глагол, НАРЕЧ – наречие, ПРИЧ – причастие, МЕСТ – местоимение, ЧИСЛ – числительное, ДЕЕПР – деепричастие, ЧАСТ – частица, ПРЕДЛ – предлог, КР.ПРИЧ – краткое причастие, КР.ПРИЛ – краткое прилагательное, КОМП – компаратив (прилагательное в сравнительной степени), ИНФ – инфинитив.

* 1. Проектирование модели базы данных

Для создания оптимальной базы данных необходимо пройти следующие три этапа:

* Концептуальное проектирование БД, т.е. построение инфологической модели БД;
* Логическое проектирование БД, т.е. построение даталогической модели БД;
* Физическое проектирование БД, т.е. построение самой БД средствами выбранной СУБД.

Основной задачей на этапе концептуального проектирования БД является выделение объектов данных, их свойств и связей между ними. Построение оптимальной инфологической модели позволит минимизировать возможность создания лишних, или отсутствия необходимых объектов и связей, замедляющих процесс обработки данных при работе разрабатываемого ПО. Основными понятиями на данном этапе являются: объекты, сущности, параметры, атрибуты.

Объект – элемент действительности, обладающий набором параметров, т.е. свойств. Отображение какого-либо объекта называется сущностью. Параметры объекта хранятся при сущности в виде её атрибутов.

В базе данных сущность становится таблицей. В таблице, как всегда, есть строки и столбцы. Одна строка называется записью. Запись – конкретный экземпляр данной сущности, обладающий тем же набором атрибутов, что и другие записи в таблице. Поля записи содержат значения атрибутов. Таким образом столбец – множество значений одного атрибута у всех записей из таблицы. Среди атрибутов выделяется один или несколько, которые называются первичным ключом, если для них выполняется условие уникальности, позволяющее однозначно определить каждую конкретную запись.

Связи отражают отношения между сущностями, и реализуются с помощью внешних ключей, т.е. полей первичных ключей, перешедших из других таблиц.

На этапе концептуального проектирования строится модель «Сущность-Связь», или ER-модель, или Инфологическая модель. Используются следующие графические средства: сущность, атрибут, связь.

По кардинальности связи классифицируют на три вида: «один-к-одному», «один-ко-многим», «многие-ко-многим». Связь «один к одному» строится тогда, когда для двух сущностей выполняется взаимное условие: один экземпляр одной сущности может быть связан с не более чем одним экземпляром другой сущности. Связь «один-ко-многим» строится, когда для двух сущностей выполняется условие: один экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, а один экземпляр второй сущности может быть связан с не более чем одним экземпляром первой сущности. Связь «многие-ко-многим» строится, когда для двух сущностей выполняется взаимное условие: один экземпляр одной сущности может быть связан с несколькими экземплярами другой сущности.

Каждой связи присваивается имя, которое отражает семантику взаимодействия экземпляров сущностей.

Прежде чем строить Инфологическую модель, необходимо построить таблицы сущностей будущей БД с указанием типов атрибутов и указанием ключевых атрибутов.

Ключевая сущность БД: «Правило». Она необходима для сохранения информации обо всех применяемых правилах, касающихся выделения синтаксических конструкций простого предложения. Именно эти правила сочетаемости словоформ применяются на этапе синтаксического анализа.

Сущность «Правило» представлена в таблице 1.

**Таблица 7. Сущность «Правило»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID правила | Целочисленный | Первичный (PK) |
| Название | Текстовый |  |
| Приоритет | Числовой |  |
| Флаг согласования рода | Логический |  |
| Флаг согласования числа | Логический |  |
| Флаг согласования падежа | Логический |  |

Ниже в таблицах 8-9 представлены две сущности, которые отвечают за сохранение информации о примененных правилах и тех словах, к которым эти правила были применены. В связи с выбранным алгоритмом синтаксического анализа, необходимо сохранить информацию о главной и подчиненной СГ.

Сущность «Главная синтаксическая группа» представлена в таблице 8.

**Таблица 8. Сущность «Главная синтаксическая группа»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID главной группы | Целочисленный | Первичный (PK) |
| Содержимое группы | Текстовый |  |
| Приоритет | Числовой |  |
| Номер предложения | Числовой |  |
| Номер части | Числовой |  |
| Номер варианта | Числовой |  |
| Номер в части | Числовой |  |

Сущность «Подчиненная СГ» представлена в таблице 9.

**Таблица 9. Сущность «Подчиненная синтаксическая группа»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID подчиненной группы | Целочисленный | Первичный (PK) |
| Содержимое группы | Текстовый |  |
| Приоритет | Числовой |  |
| Номер предложения | Числовой |  |
| Номер части | Числовой |  |
| Номер варианта | Числовой |  |
| Номер в части | Числовой |  |

Сущность «Тип входной синтаксической группы» представлена в таблице 10. Она описывает два возможных типа: «подчиненный» и «главный», в соответствии с которыми идёт управление двумя СГ, которые связываются по какому-либо правилу. Таким образом типы 1-й и 2-й СГ определяется правилом.

**Таблица 10. Сущность «Тип входной синтаксической группы»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID типа входной синтаксической группы | Целочисленный | Первичый (PK) |
| Название типа входной синтаксической группы | Текстовый |  |

Сущность «Вид синтаксической группы» представлена в таблице 11. Данная сущность унифицирует синтаксические группы по видовому признаку, а именно их главные слова, благодаря чему делается возможным применение правила к двум СГ.

**Таблица 11. Сущность «Вид синтаксической группы»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID вида синтаксической группы | Целочисленный | Первичный (PK) |
| Название вида синтаксической группы | Текстовый |  |

Сущность «Часть речи» представлена в таблице 12. Каждое слово – представитель определенной части речи русского языка, поэтому эта характеристика – обязательная связь между русским языком и принципом СГ.

**Таблица 12. Сущность «Часть речи»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID части речи | Целочисленный | Первичный (PK) |
| Название части речи | Текстовый |  |

Сущность «Граммема» представлена в таблице 13.

**Таблица 14. Сущность «Граммема»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID граммемы | Целочисленный | Первичный (PK) |
| Содержание граммемы | Текстовый |  |

Сущность «Класс синтаксической группы» представлена в таблице 15.

**Таблица 15. Сущность «**Класс синтаксической группы**»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID класса | Целочисленный | Первичный (PK) |
| Название класса | Текстовый |  |

Сущность «Член предложения» представлена в таблице 16. Данная таблица дает информацию о том, какой член предложения реализуется той или иной СГ.

**Таблица 16. Сущность «Член предложения»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID члена предложения | Целочисленный | Первичный (PK) |
| Название члена предложения | Текстовый |  |

Ниже представлены три сущности, соответствующие трём видам граммем русского языка: род, число и падеж, в соответствии с которыми определяется лексически правильное взаимодействие слов.

Сущность «Род» представлена в таблице 17.

**Таблица 17. Сущность «Род»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID рода | Целочисленный | Первичный (PK) |
| Название рода | Текстовый |  |

Сущность «Число» представлена в таблице 18.

**Таблица 18. Сущность «Число»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID числа | Целочисленный | Первичный (PK) |
| Название числа | Текстовый |  |

Сущность «Падеж» представлена в таблице 19.

**Таблица 19. Сущность «Падеж»**

| **Логическое имя атрибута** | **Тип данных** | **Ключ** |
| --- | --- | --- |
| ID падежа | Целочисленный | Первичный (PK) |
| Название падежа | Текстовый |  |

Ключевые атрибуты, показанные в таблицах 7-19, на Инфологической модели БД выделяются подчёркиванием, но прежде, чем строить эту модель, необходимо определить связи между сущностями БД. А на основе ключевых атрибутов в дальнейшем будут строится первичные ключи в таблицах Даталогической модели.

Преобразование ER-модели в реляционную модель БД происходит с учётом типа связи:

* Связь «1:1». Для реализации данного типа связи требуется от одной до двух таблиц в зависимости от того, обязательно ли поле.
* Связь «1:М». Связь может быть реализована двумя или тремя таблицами.
* Связь «М:1». Также необходимо две или три таблицы.
* Связь «М:N». Такая связь требует обязательного построения третей таблицы – одну для каждой сущности и одну для связи.

Первичный ключ должен удовлетворять ряду требований:

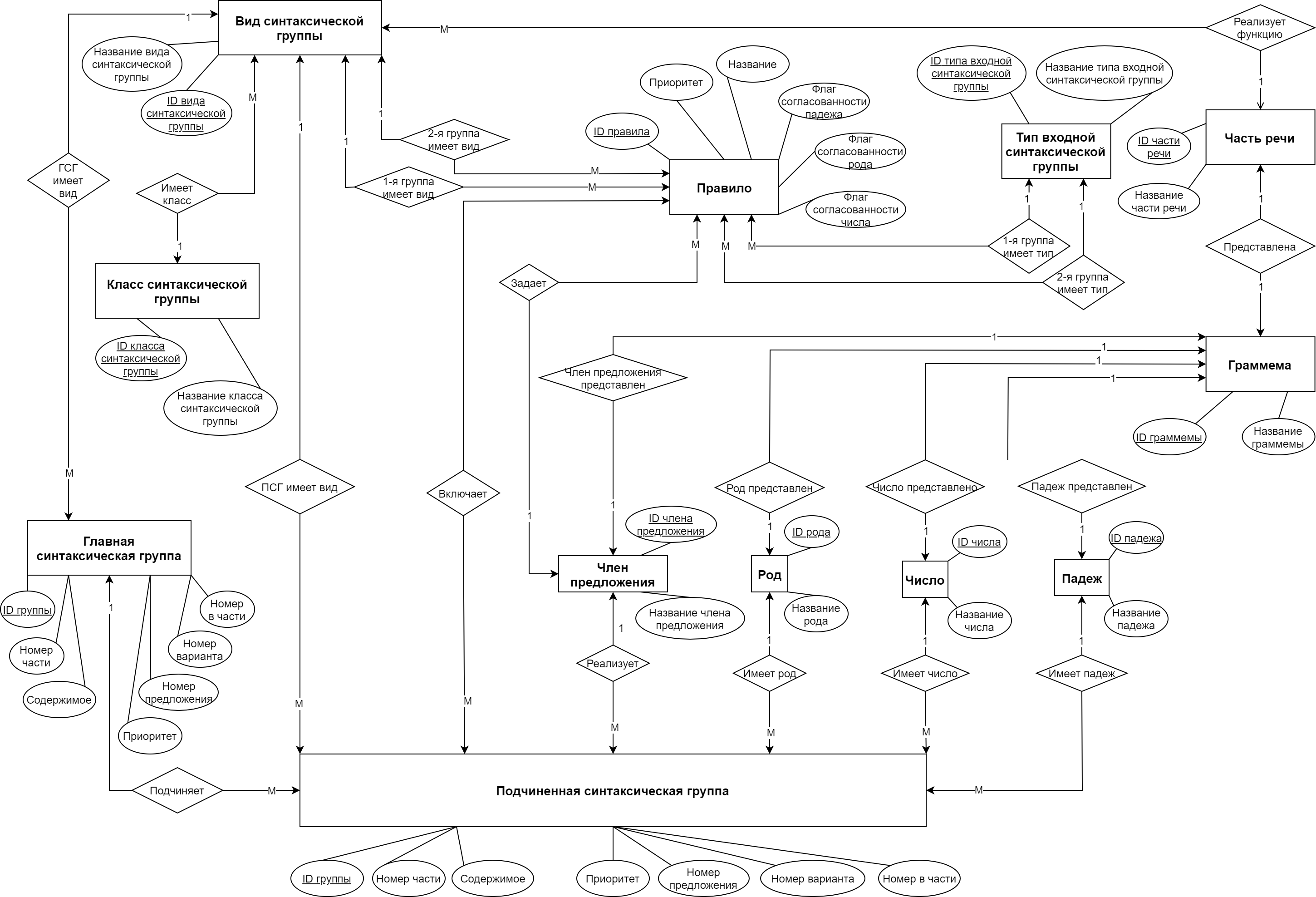
* Уникальность. Два экземпляра не должны иметь одинаковых значений ключа.
* Минимум атрибутов. Составной первичный ключ (группа атрибутов) не должен содержать ни одного атрибута, удаление которого не приводило бы к утрате уникальности.

Внешний ключ (Foreign Key) появляется в процессе установления связей между сущностями. Это описательный атрибут, который назначается в дополнение к идентифицирующему атрибуту и служит не для идентификации этого уникального экземпляра, а для выделения набора интересующих тех объектов по каким-либо внешним признакам, привязанным к этим объектам из других таблиц.

С учётом описания сущностей формируются следующие связи:

* «Род представлен»: связь между сущностями «Граммема» и «Род» типа «один-к-одному»
* «Число представлено»: связь между сущностями «Граммема» и «Число» типа «один-к-одному»
* «Падеж представлен»: связь между сущностями «Граммема» и «Падеж» типа «один-к-одному»
* «Часть речи представлена»: связь между сущностями «Граммема» и «Часть речи» типа «один-к-одному»
* «Член предложения представлен»: связь между сущностями «Граммема» и «Член предложения» типа «один-к-одному»
* «Реализует функцию»: связь между сущностями «Часть речи» и «Вид синтаксической группы» типа «один-ко-многим»
* «1-я группа имеет тип»: связь между сущностями «Тип входной синтаксической группы» и «Правило» типа «один-ко-многим»
* «2-я группа имеет тип»: связь между сущностями «Тип входной синтаксической группы» и «Правило» типа «один-ко-многим»
* «1-я группа имеет вид»: связь между сущностями «Вид синтаксической группы» и «Правило» типа «один-ко-многим»
* «2-я группа имеет вид»: связь между сущностями «Вид синтаксической группы» и «Правило» типа «один-ко-многим»
* «Имеет класс»: связь между сущностями «Класс синтаксической группы» и «Вид синтаксической группы» типа «один-ко-многим»
* «ГСГ имеет вид»: связь между сущностями «Вид синтаксической группы» и «Главная синтаксическая группа» типа «один-ко-многим» (ГСГ – Главная Синтаксическая Группа)
* «ПСГ имеет вид»: связь между сущностями «Вид синтаксической группы» и «Подчиненная синтаксическая группа» типа «один-ко-многим» (ПСГ – Подчиненная Синтаксическая Группа)
* «Определена»: связь между сущностями «Правило» и «Подчиненная синтаксическая группа» типа «один-ко-многим»
* «Подчиняется»: связь между сущностями «Вид синтаксической группы» и «Подчиненная синтаксическая группа» типа «один-ко-многим»
* «Реализует»: связь между сущностями «Член предложения» и «Подчиненная синтаксическая группа» типа «один-ко-многим»
* «Задает»: связь между сущностями «Член предложения» и «Правило» типа «один-ко-многим»
* «Имеет род»: связь между сущностями «Род» и «Подчиненная синтаксическая группа» типа «один-ко-многим»
* «Имеет число»: связь между сущностями «Число» и «Подчиненная синтаксическая группа» типа «один-ко-многим»
* «Имеет падеж»: связь между сущностями «Падеж» и «Подчиненная синтаксическая группа» типа «один-ко-многим».

На рисунке 14 представлена Инфологическая модель БД.

Рисунок 14. Схема Инфологической модели.

Под даталогической понимается модель, отражающая логические взаимосвязи между элементами данных безотносительно их содержания и физической организации. Даталогическое проектирование включает построение БД в виде таблиц. Особенности выбранной СУБД также влияют на проеткирование даталогической модели, например, при описании названий типов данных, их размера. Результатом данной стадии проектирования БД является построение датологической модели в виде набора таблиц, отражающих основные свойства и взаимодействия сущностей предметной области.

Таблица содержит название и набор полей, которые характеризуются типом данных. Поле может содержать информативные данных стандартных типов данных (целый, с плавающей точкой, строковый и т.д.) или быть ссылкой на другую таблицу – внешним ключом.

При добавлении информации в БД в таблицы вставляются записи, для которых каждое поле принимает значение заданного типа.

Проектированию базы данных уделяется особое внимание по причине того, что внесение изменений в структуру при наличии данных влечет дополнительные сложности, объясняющиеся необходимостью специальными методами перезаписывать и изменять все имеющиеся данные.

Данная предметная область облегчает построение Базы данных, поскольку ее данные используют лишь алгоритмы синтаксического анализатора. Она не будет содержать конфиденциальных или сколь угодно уникальных данных, поэтому БД проекта не нуждается в дополнительных средствах защиты данных, не предусмотренных базовыми настройками выбранной СУБД.

Перевод названий элементов структуры БД проводился с использованием источника [14].

Модель «Rule» представлена в таблице 20.

**Таблица 20. Модель «Rule»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID правила | ID | INT(11) | Auto Incremental | Primary Key |
| Название правила | Name | VARCHAR(45) | NULL |  |
| Приоритет | Priority | INT(11) | NULL |  |
| Флаг согласования рода | Gender consistency | TINYINT(4) | NULL |  |
| Флаг согласования числа | Number consistency | TINYINT(4) | NULL |  |
| Флаг согласования падежа | Case consistency | TINYINT(4) | NULL |  |
| Вид первой СГ | 1G Syntax kind ID | INT(11) | NULL | Foreign Key |
| Вид второй СГ | 1G Syntax kind ID | INT(11) | NULL | Foreign Key |
| Тип первой СГ | 1G Syntax kind ID | INT(11) | NULL | Foreign Key |
| Тип второй СГ | 1G Syntax kind ID | INT(11) | NULL | Foreign Key |
| Член предложения подчиненной группы | 1G SG attribute ID | INT(11) | NULL | Foreign Key |

Модель «Syntax kind» представлена в таблице 21.

**Таблица 21. Модель «Syntax kind»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID вида СГ | ID | INT(11) | Auto Incremental | Primary Key |
| Название вида СГ | Name | VARCHAR(45) | NULL |  |
| Части речи | Part of speech ID | INT(11) | NULL | Foreign Key |
| Класс СГ | Syntax class ID | INT(11) | NULL | Foreign Key |

Модель «Input syntax type» представлена в таблице 22.

**Таблица 22. Модель «Input syntax type»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID правила | ID | INT(11) | Auto Incremental | Primary Key |
| Название правила | Name | VARCHAR(45) | NULL |  |

Модель «Attribute» представлена в таблице 23.

**Таблица 23. Модель «Attribute»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID члена предложения | ID | INT(11) | Auto Incremental | Primary Key |
| Название члена предложения | Name | VARCHAR(45) | NULL |  |

Модель «Syntax class» представлена в таблице 24.

**Таблица 24. Модель «Syntax class»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID класса СГ | ID | INT(11) | Auto Incremental | Primary Key |
| Название класса СГ | Name | VARCHAR(45) | NULL |  |

Модель «Grammeme» представлена в таблице 25.

**Таблица 25. Модель «Grammeme»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID граммемы | ID | INT(11) | Auto Incremental | Primary Key |
| Название граммемы | Name | VARCHAR(45) | NULL |  |

Модель «Gender» представлена в таблице 26.

**Таблица 26. Модель «Gender»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID рода | ID | INT(11) | Auto Incremental | Primary Key |
| Название рода | Name | VARCHAR(45) | NULL |  |
| Граммема | Grammeme ID | INT(11) | NULL | Foreign Key |

Модель «Case» представлена в таблице 27.

**Таблица 27. Модель «Case»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID падежа | ID | INT(11) | Auto Incremental | Primary Key |
| Название падежа | Name | VARCHAR(45) | NULL |  |
| Граммема | Grammeme ID | INT(11) | NULL | Foreign Key |

Модель «Number» представлена в таблице 28.

**Таблица 28. Модель «Number»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID числа | ID | INT(11) | Auto Incremental | Primary Key |
| Название числа | Name | VARCHAR(45) | NULL |  |
| Граммема | Grammeme ID | INT(11) | NULL | Foreign Key |

Модель «Part of speech» представлена в таблице 29.

**Таблица 29. Модель «Part of speech»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID часть речи | ID | INT(11) | Auto Incremental | Primary Key |
| Название части речи | Name | VARCHAR(45) | NULL |  |
| Граммема | Grammeme ID | INT(11) | NULL | Foreign Key |

Модель «Main syntax group» представлена в таблице 30.

**Таблица 30. Модель «Main syntax group»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID группы | ID | INT(11) | Auto Incremental | Primary Key |
| Содержимое | Content | VARCHAR(45) | NULL |  |
| Приоритет | Priority | INT(11) | NULL |  |
| Номер предложения | Sentence number | INT(11) | NULL |  |
| Номер части | Part number | INT(11) | NULL |  |
| Номер варианта | Variant number | INT(11) | NULL |  |
| Номер в части | Number in part | INT(11) | NULL |  |
| Вид СГ | Syntax kind ID | INT(11) | NULL | Foreign Key |

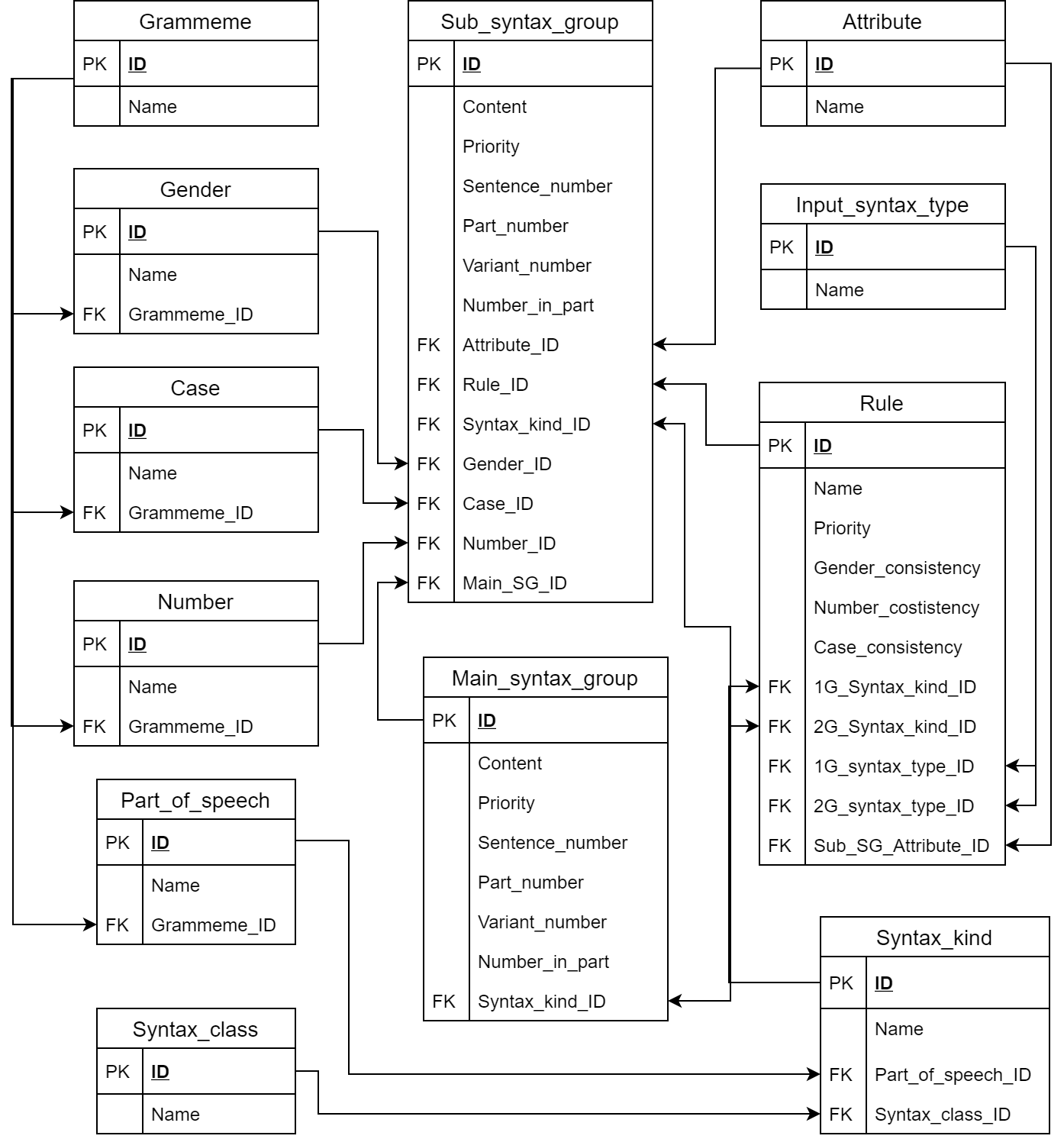
Модель «Sub syntax group» представлена в таблице 31.

**Таблица 31. Модель «Sub syntax group»**

| **Логическое имя атрибута** | **Поле** | **Тип данных** | **Default** | **Ключ** |
| --- | --- | --- | --- | --- |
| ID группы | ID | INT(11) | Auto Incremental | Primary Key |
| Содержимое | Content | VARCHAR(45) | NULL |  |
| Приоритет | Priority | INT(11) | NULL |  |
| Номер предложения | Sentence number | INT(11) | NULL |  |
| Номер части | Part number | INT(11) | NULL |  |
| Номер варианта | Variant number | INT(11) | NULL |  |
| Номер в части | Number in part | INT(11) | NULL |  |
| Член предложения | Attribute ID | INT(11) | NULL | Foreign Key |
| Правило | Rule ID | INT(11) | NULL | Foreign Key |
| Вид СГ | Syntax kind ID | INT(11) | NULL | Foreign Key |
| Род | Gender ID | INT(11) | NULL | Foreign Key |
| Число | Number ID | INT(11) | NULL | Foreign Key |
| Падеж | Case ID | INT(11) | NULL | Foreign Key |
| Главная СГ | Main SG ID | INT(11) | NULL | Foreign Key |

Связи даталогической модели формируются на основе предыдущих этапов проектирования. На рисунке 15 представлена итоговая схема Даталогической модели.

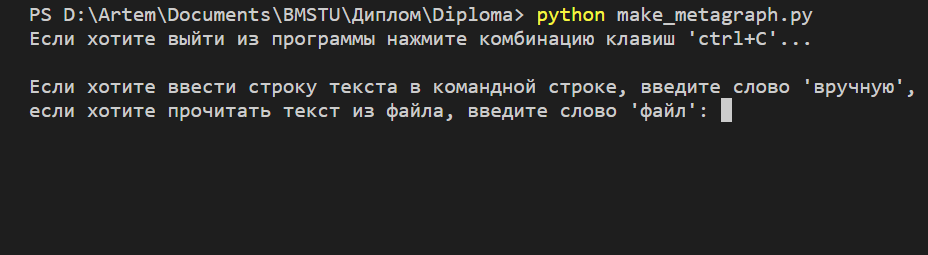
По результатам данного этапа проектирования строится сама БД средствами СУБД MySQL.

Рисунок 15. Схема даталогической модели

* 1. Разработка программного продукта
     1. Разработка интерфейса пользователя

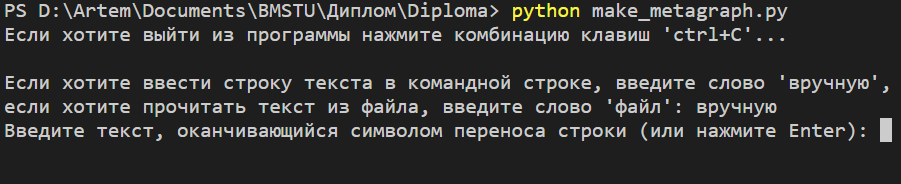
Продукт предназначается для специалистов-разработчиков, поэтому специального интерфейса нет. В рамках дипломного проекта разработана простая система взаимодействия с пользователем в рамках его работы в командной строке.

На рисунке 16 изображен интерфейс пользователя после запуска им программы, а точнее, главного файла проекта «make\_metagraph.py».

  
Рисунок 16. Интерфейс при запуске программы.

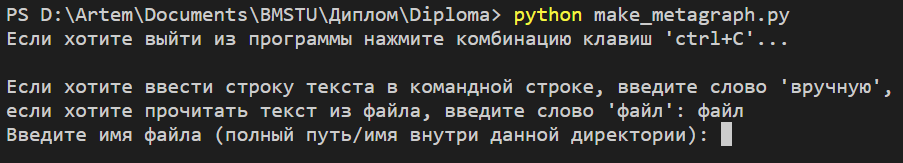
Пользователю предлагается ввести одно из двух ключевых слов для дальнейшего взаимодействия с программой: «вручную» или «файл».

Если пользователь введёт «вручную», перед ним появится следующая картинка (см. рисунок 17):

  
Рисунок 17. Интерфейс при вводе «вручную».

Пользователю предлагается ввести текст с помощью клавиатуры. Нажатие клавиши Enter окончит ввод. Таким образом пользователь может ввести только один абзац текста в данном режиме.

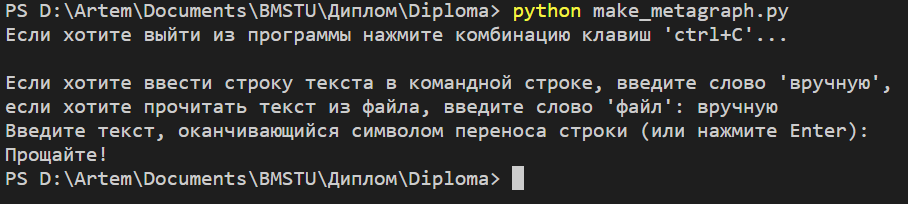
Если пользователь введёт «файл», перед ним появится следующая картинка (см. рисунок 18):

  
Рисунок 18. Интерфейс при вводе «файл».

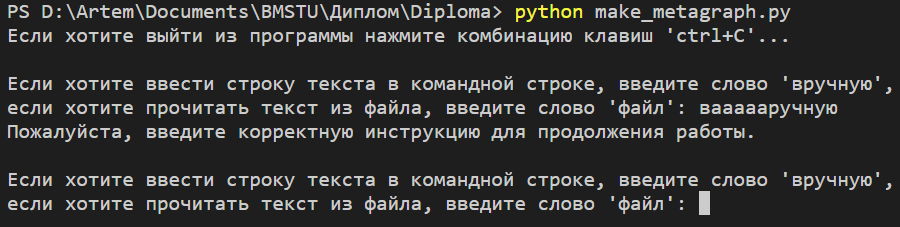
Пользователю предлагается ввести имя файла. Если он хочет проанализировать файл с текстовой информацией, который находится в директории проекта, то может ввести только имя файла. Если этот файл находится в другом месте на его рабочем компьютере, то придется вводить полный путь до файла.

После ввода в режиме «вручную» или в режиме «файл», программа проанализирует текст и выдаст результат работы программы, которым является метаграфовое описание синтаксического разбора текста. Подробнее результаты работы программы описаны в пункте 2.8 данной работы.

Если пользователь ошибся при выборе режима, передумал работать с программой или захотел выйти из программы досрочно, он может задействовать комбинацию клавиш «ctrl+C» клавиатуры. На рисунке 19 приведен результат такого завершения работы программы.

  
Рисунок 19. Прощание с пользователем.

Если пользователь некорректно ввел слова «файл» или «вручную», программа попросит пользователя ввести корректную команду (см. рисунок 20):

  
Рисунок 20. Попытка ввести неправильную команду.

Если пользователь ввел корректно команду, имя файла или текст, что он хочет проанализировать, то программа проведет анализ и завершит работу.

* + 1. Разработка порядка работы алгоритмов синтаксического анализа

После получения морфологической информации (с помощью библиотек ru\_sent\_tokenize, NLTK и pymorphy2) о составе предложения начинается этап синтаксического анализа, который проходит в два этапа:

1. Производится разделение сложных предложений на простые:

* Формирование списка союзов, которые могут разделять простые предложения в составе сложного (сложно-подчиненные и сложно-сочиненные). Прежде чем приступать к следующей стадии, происходит фильтрование ложных вариантов союзов: например, были найдены варианты союза (между элементами которого нет других союзов) «если …, то»: «если …, то» и «если …, то …, то» – в этом случае будет выбран второй вариант, поскольку отдельного союза «то» не существует, а сочетание «, то», где «то» – не часть союза, не является корректной для русского языка;
* С помощью оптимального списка союзов-разделителей (в котором, например, составные союзы не разделяются другими союзами) происходит выделение всех возможных пар подряд идущих простых предложений в составе сложного;
* Для каждой пары производится проверка наличия вариантов грамматических основ двух возможных частей предложения, которые разделяет какой-то союз-разделитель. Две части в составе сложного предложения формируются только в том случае, если в обеих этих частях есть свои варианты грамматической основы. Если хотя бы в одной из этих двух предполагаемых частей грамматическая основа не может быть сформирована, то эти две части не разделяются.

2) Анализ каждой части предложения с учётом всех возможных вариантов грамматической основы данной части:

* Так как каждая лексема части может иметь несколько вариантов морфологического разбора, с помощью рекурсивного алгоритма формируется необходимое количество (произведение морфологических вариантов всех лексем части) уникальных цепочек из вариантов лексем;
* Каждый такой вариант цепочки рассматривается вслед за предыдущим на следующей стадии;
* Если этот вариант сохранил хотя бы один из вариантов грамматической основы, которую изначально могла иметь данная часть после шага 1), производится синтаксический анализ с использованием правил (подробно схема применения правил описана в пункте 2.5);
* Вариант считается успешно разобранным, если после синтаксического анализа не осталось неразобранных слов, после этого он сохраняется в списке вариантов разбора части данного предложения;
  + 1. Разработка алгоритма синтаксического анализа

Входные данные:

* Входное предложение;
* Список союзов-разделителей;
* Список индексов частей предложения;
* Номер предложения в абзаце (начинается с 0);
* Номер абзаца в тексте (начинается с 0);
* Список вариантов грамматической основы;

Заводится список флагов на два типа результатов анализа: существование вариантов для разбора конкретной части предложения (True для всей части, если есть грамматическая основа), существование реальных результатов анализа части предложения (True для варианта, если все слова варианта были разобраны).

Далее для каждой из частей предложения проводится поиск всех вариантов цепочек входящих в предложение лексем, основываясь на результатах их морфологического анализа. Данный поиск проводится с помощью рекурсивного обхода в глубину.

После того, как вариант для цепочки найден, проверяется, входит ли в него грамматическая основа части (хотя бы один ее вариант). Если грамматическая основа есть - проводится анализ с помощью правил.

Разбор предложения идет слева направо. Знаки препинания не игнорируются: если какая-то часть предложения обособлена запятыми, то сначала будет проведен анализ словосочетаний внутри этой обособленной части, а уже потом главное слово этой части будет взаимодействовать с остальным предложением. Если обособленная часть предложения стала разобранной, то в тексте автоматически появляются 2 идущих подряд необработанных знака препинания. В таком случае происходит простановка флага обработки первого из двух знаков препинания. Обычно это запятая, тире или двоеточие, поскольку остальные знаки пунктуации являются разделителями предложений.

Таким образом делается возможным обработка однородных членов предложения, однородных простых предложений в составе сложного.

Если в тексте появилось неизвестное слово, или слово, которое не может по правилам быть обработанным, его флаг обработки остается ложным. Поэтому после окончания разбора, то есть после неудачной попытки провести сочетание СГ по всем возможным приоритетам по всей части предложения данный вариант разбора части становится неактуальным.

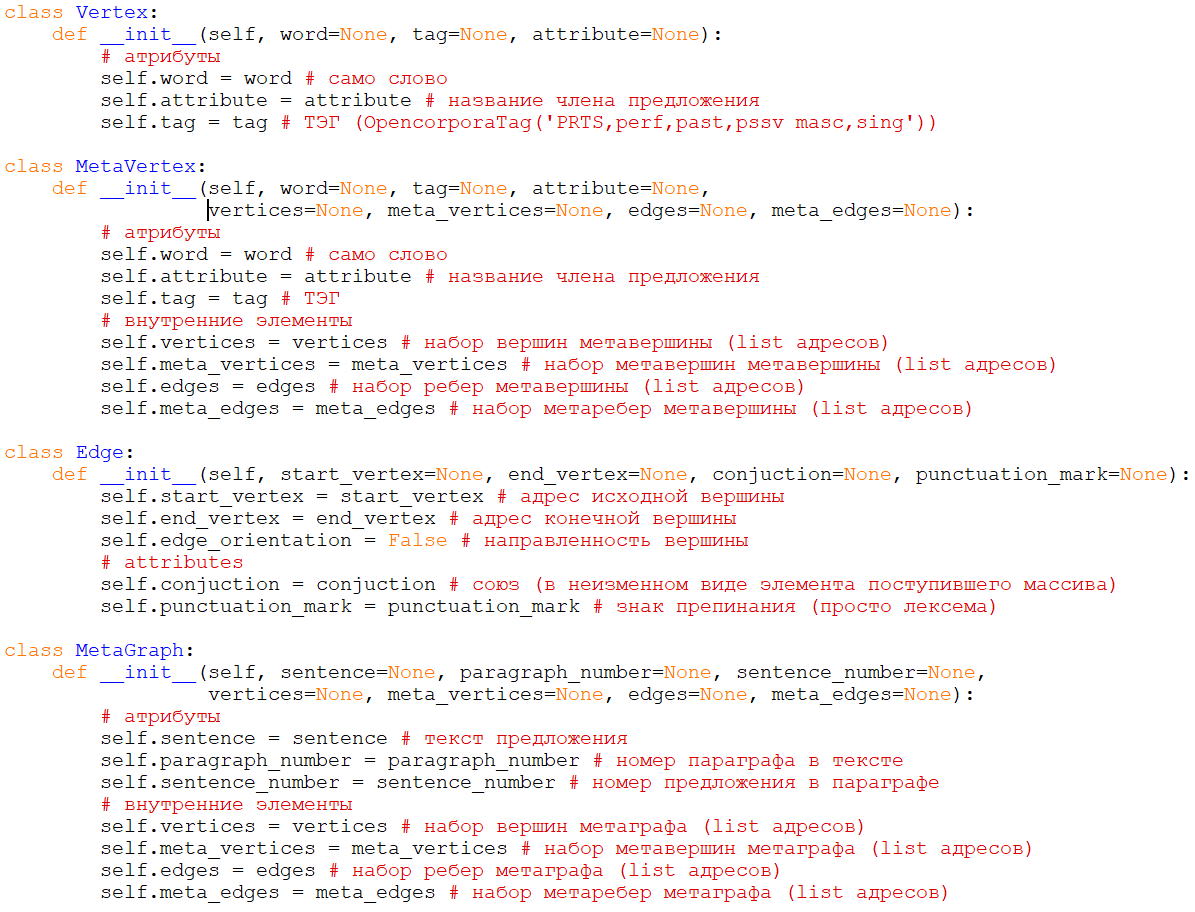
* + 1. Разработка структуры классов модуля построения метаграфа

В своей работе я использую метаграфовую модель в соответствии с источником [3]:

где MG – метаграф, V – множество вершин метаграфа; MV – множество метавершин метаграфа; E – множество ребер метаграфа; ME – множество метаребер метаграфа.

Единственным исключением в применении данной модели на этапе синтаксического анализатора является то, что не используются метаребра. Функционала Вершин, Рёбер и Метавершин достаточно для реализации записей о словах и об их взаимодействии между собой.

На рисунке 21 представлен код Классов Вершина, Метавершина, Ребро и Метаграф соответственно.

Рисунок 21. Классы метаграфа и его элементов

У вершины-слова атрибутами являются: само слово, член предложения, который реализует слово, а также тэг слова. Тэг – набор морфологических характеристик, которые имеет данное слово по правилам русского языка, и взятых из словаря OpenCorpora. Данная информация о конкретном слове, а также значение поля attribute являются необходимой и достаточной информации о лексической роли слова в предложении. Полный список тэгов вы можете посмотреть в источнике [6], являющимся страницей официального сайта OpenCorpora.

У метавершины-части предложения атрибутами являются те же поля. Значением атрибута word в нем будет являться грамматическая основа предложения в формате «подлежащее\_сказуемое». Также метавершина имеет набор вершин, ребер, метавершин, которые могут входить в её состав.

У ребра атрибуты: исходная и конечная вершины, а также ориентация вершины, союз и знак пунктуации. Последние два атрибута необходимы, чтобы сохранить информацию о связующих лексемах предложения: союзах и знаках препинания: запятая, двоеточие, тире. Знаки препинания несут определенную семантическую нагрузку, о которой мы не можем делать выводы на данном синтаксическом этапе разбора предложения.

Метаграф – это хранилище информации о предложении, поэтому одним из его атрибутов является само предложение. Также атрибутами являются номер параграфа, в котором находится данное предложение и номер предложения в данном абзаце. Объект метаграфа хранит список вершин, метавершин, ребер и метарёбер данного метаграфа.

* + 1. Разработка алгоритма преобразования результатов синтаксического анализа в метаграф

Так как предложения это, по сути, набор слов, то помимо самого слова, метавершина должна обладать информацией о том, какой номер имеет данное слово в данной части предложения. Таким образом слова внутри метавершины следует хранить не как список, а как словарь. Элементом словаря будет являться само слово, а ключом к данному элементу – его номер в предложении.

Таким же образом хранится информация о рёбрах, находящихся в составе метавершины. Ключом к элементу ребра является кортеж из двух значений: номера главного и подчинённого слов в части предложения.

Информация о рёбрах, которые связывают метавершины метаграфа, тоже хранится в словаре. Ключом к элементу данного словаря является кортеж из 4 элементов: номера главной и подчиненной частей предложения, номера вариантов разбора главной и подчиненной частей предложения. Таким образом, какую бы главную и подчиненную вершины, мы бы не связали, какой бы вариант разбора части предложения мы не выбрали, мы получим уникальный ключ для каждого ребра. Данный ключ позволит получить доступ к уникальным метавершинам и вершинам, информация о которых не будет дублироваться, благодаря структурам данных языка Python 3.

У каждой части предложения возможно несколько вариантов разбора, которые никак не связаны между собой. Поэтому каждый такой вариант разбора будет являться элементом соответствующего списка метавершин. Каждый элемент данного списка является в принципе равноправным по сравнению с другими элементами после осуществления синтаксического разбора. Тот или иной вариант разбора части получает больший номер в связи с результатами морфологического и синтаксического разборов, которые помимо всего прочего показывают вероятность актуальности того или иного варианта: чем больше вероятность, тем больше номер варианта в списке. Таким образом, метаграф хранит словарь метавершин, элементами которых являются списки вариантов разбора каждой части предложения.

Таким образом мы храним информацию без излишеств и в максимально удобной и формализованной форме без хранения в памяти лишних индексов и обеспечением максимально быстрого доступа к этим данным. В будущем возможна разработка модуля, который позволит записывать информацию о разборе метаграфа в специальный файл, возможно, уникального формата.

* 1. Результаты опытной эксплуатации

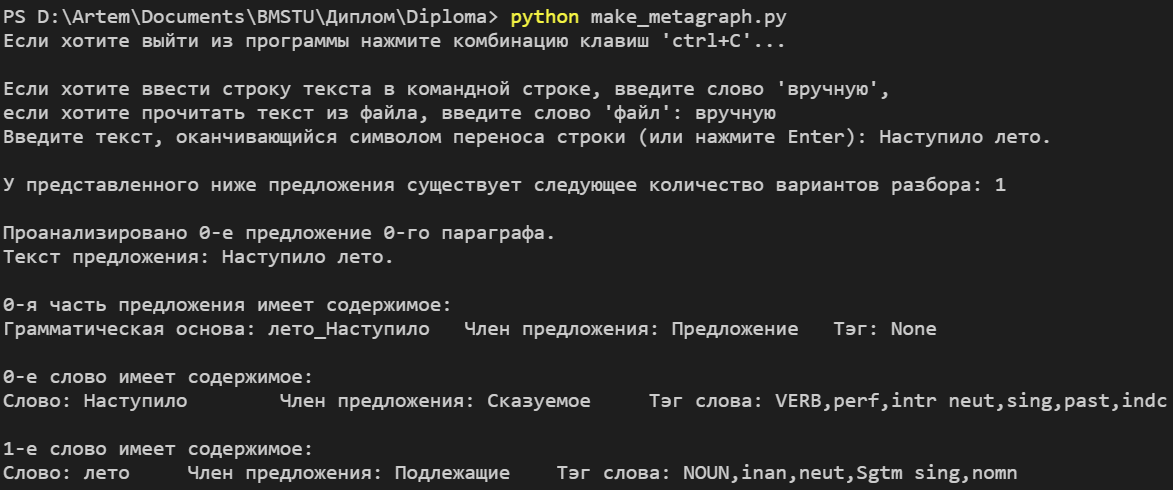
Ниже приведены примеры разборов 4 предложений различного строения, показывающие правильность работы системы:

* Наступило лето.
* Сегодня шёл сильный дождь.
* Отдыхающий насорил на пляже, поэтому был оштрафован.
* Косой косой косил траву косой косой.

Разбор в режиме «вручную» аналогичен разбору в режиме «файл», поэтому для демонстрации используется исключительно первый режим. Каждый раз будет вводиться одно предложение, однако это не отменяет того, что программой данной предложение воспринимается как текст.

Результатом разбора для пользователя будет консольное представление метаграфа, как хранилища результатов синтаксического разбора. Для экономии места на экране из демонстрации убраны сведения рёбер метаграфа, с сохранением информации о вершинах и метавершинах.

На рисунке 22 представлен результат разбора предложения «Наступило лето».

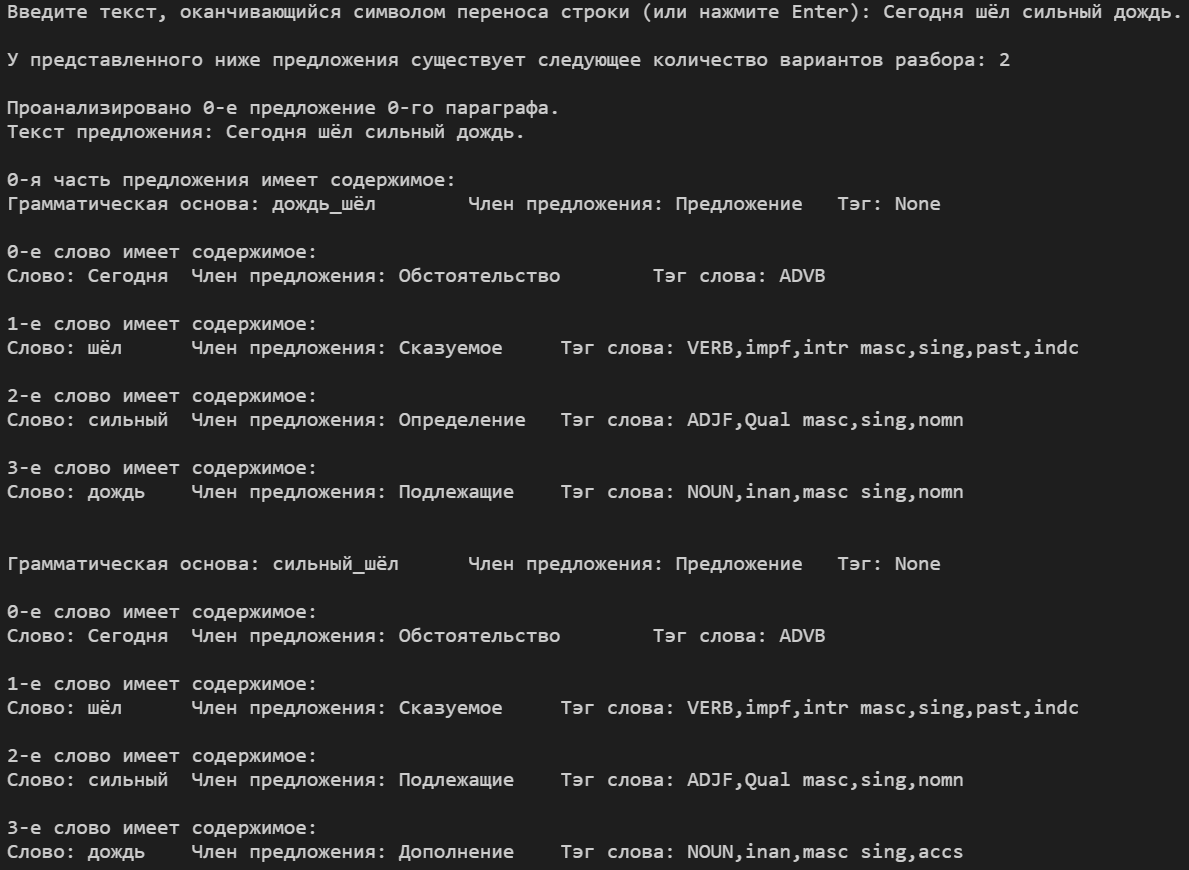
  
Рисунок 22. Результат разбора предложения «Наступило лето».

Это предложение состоит лишь из слов грамматической основы, поэтому единственные два слова в его составе идентифицируются однозначно как подлежащее и сказуемое.

В результате разбора за каждым словом сохраняется его содержимое, тэг и реализуемый им член предложения.

На рисунке 23 представлен результат разбора предложения «Сегодня шёл сильный дождь».

У этого предложения два варианта разбора, поскольку возможна ситуация, когда подлежащее выражено прилагательным в именительном падеже, у которого есть зависимое слово в другом падеже. В данном случае используются слова «сильный» и «дождь», форма которых в именительном и родительном падеже одинакова, что вызывает удивление при получении результата. Однако синтаксический анализ не может понять, что перед ним за слово, имеет оно изменяемые форму или нет, поскольку смотрит только на граммемы каждого из слов и делает выводы на основе общих правил.

  
Рисунок 23. Результат разбора предложения «Сегодня шёл сильный дождь».

В результате разбора за каждым словом сохраняется его содержимое, тэг и реализуемый им член предложения.

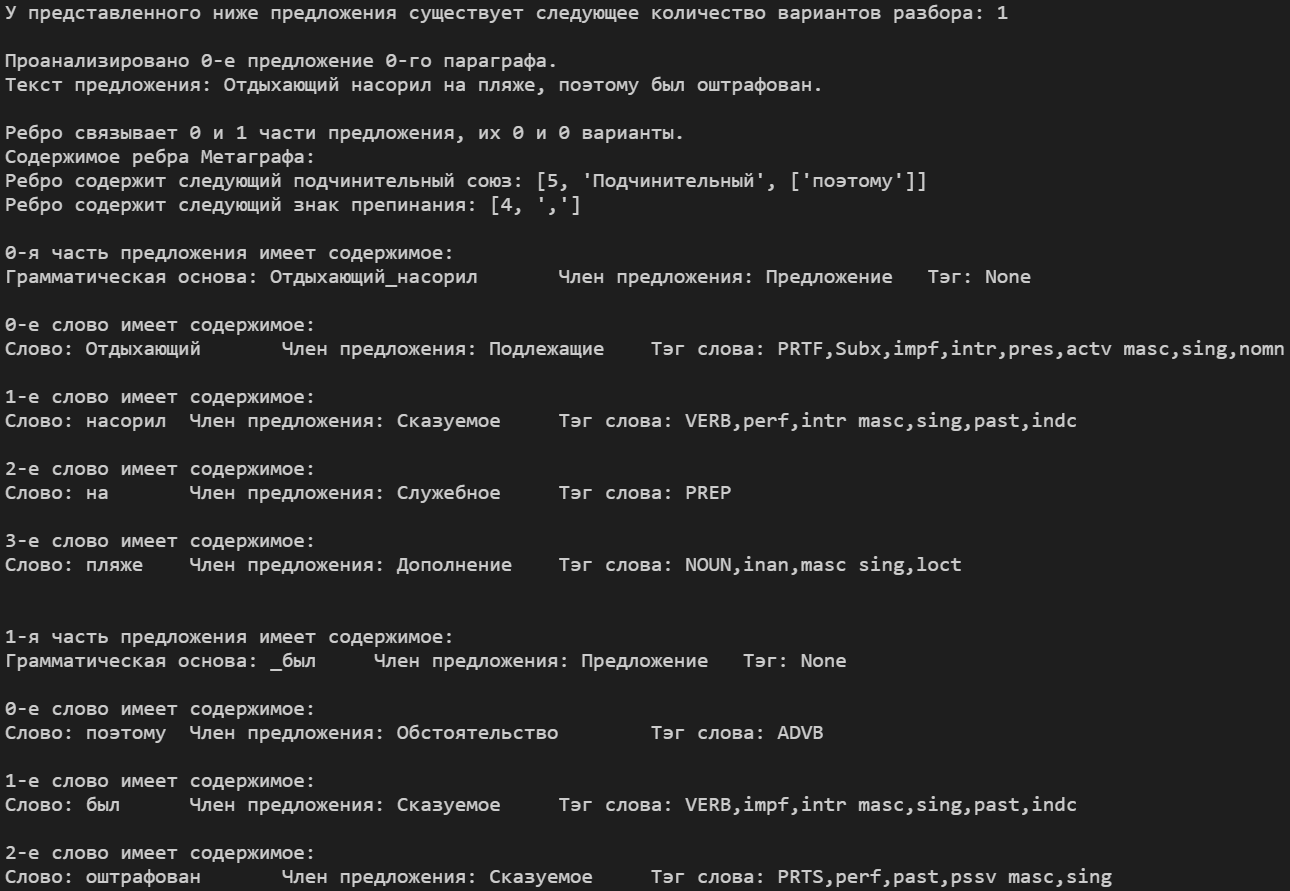
Мы видим правильный результат разбора. Он наиболее вероятный, а потому выводится первым.

Дождь – подлежащее. Дождь (какой?) сильный – определение дождя.

Шёл – сказуемое. Шёл (как? когда?) сегодня – обстоятельство действия.

Если заменить слово «шёл» на слово «лил», то мы выясним, что морфологический словарь имеет в своем составе некое существительное «Лила». Это существительное – имя, которое во множественном числе винительного и родительного падежа даёт словоформу «лил».

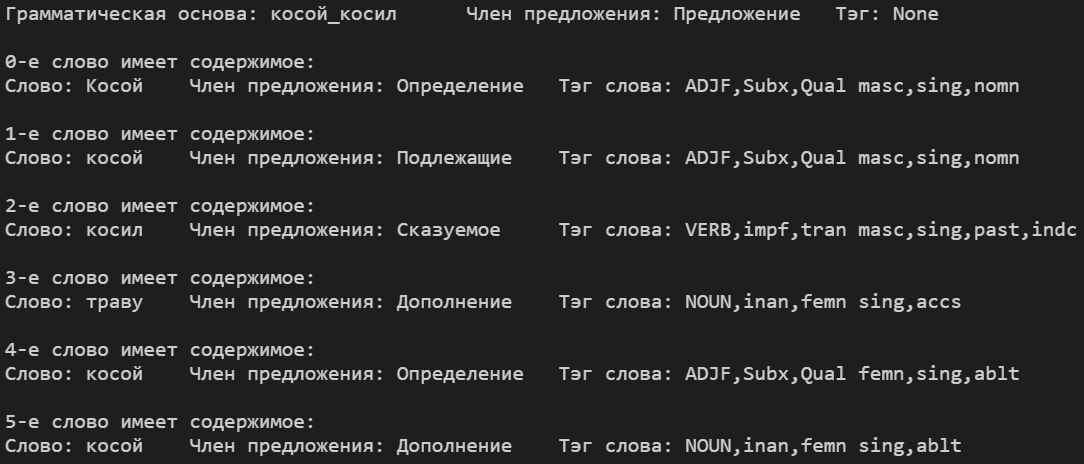
На рисунке 24 представлен результат разбора предложения «Отдыхающий насорил на пляже, поэтому был оштрафован».

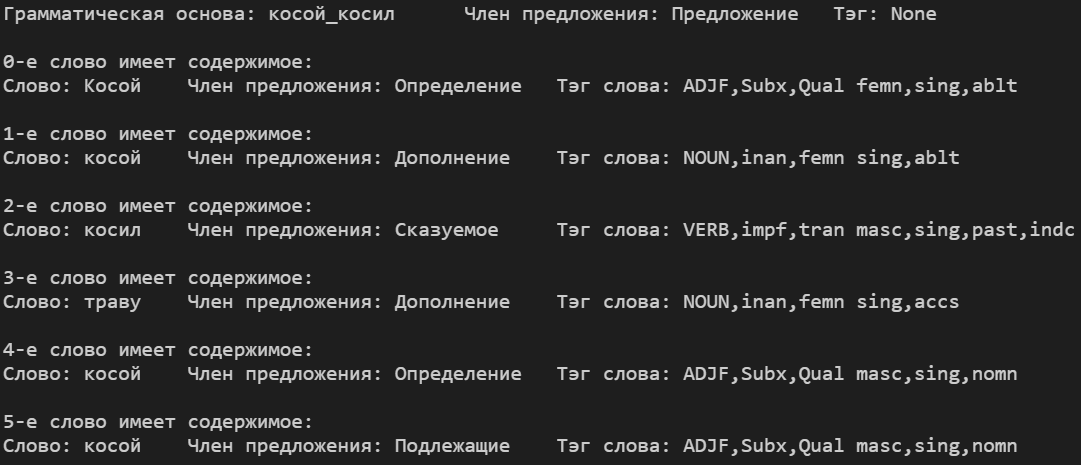
  
Рисунок 24. Результат разбора предложения «Отдыхающий насорил на пляже, поэтому был оштрафован».

Это предложение состоит лишь из слов грамматической основы, поэтому единственные два слова в его составе идентифицируются однозначно как подлежащее и сказуемое.

В результате разбора за каждым словом сохраняется его содержимое, тэг и реализуемый им член предложения.

На рисунках 25 и 26 представлен результат разбора предложения «Косой косой косил траву косой косой».

  
Рисунок 25. Первый вариант разбора предложения «Косой косой косил траву косой косой».

  
Рисунок 26. Второй вариант разбора предложения «Косой косой косил траву косой косой».

В результате разбора за каждым словом сохраняется его содержимое, тэг и реализуемый им член предложения.

Это предложение состоит из 6 слов, 4 из которых абсолютно одинаковые. Естесственно, это увеличивает количество вариантов разбора. Однозначным сказуемым является «косил», а дополнением к нему – «траву».

Однако даже человек видит, что подлежащее с определением может стоять и в начале предложения, и в конце. Поэтому у данного предложения минимум 2 абсолютно верных варианта разбора. Причем они равновероятны настолько, что от итогового выбора одного из них контекст не изменится совершенно.

Помимо этого, есть еще несколько результатов, которые определяют два подряд «косой» как дополнение к «косил» (косил (чем?) косой), но не как определение к «косе», поскольку такие дополнения могут быть рядами однородных членов. Поэтому, чтобы отсеять такие неверные варианты, необходимо провести анализ результатов с помощью семантического анализа, который на данном этапе и в данной работе не рассматривается.

ЗАКЛЮЧЕНИЕ

При разработке информационной системы планирования и учета личных тренировок были получены следующие результаты:

* Была изучена предметная область, связанная с текстовыми анализаторами, тонкости различных подходов к разработке алгоритмов.
* Были сформулированы требования к разрабатываемой системе, в том числе к правилам, алгоритмам их применения и БД.
* Спроектирована База Данных, пригодная для хранения правил обработки словосочетаний, структура алгоритма подготовки к синтаксическому анализу и его осуществления, структура классов, преобразующих результаты анализа в метаграф.
* Разработана система правил и их применения, алгоритмы их применения и преобразования результатов анализа в метаграф.

Полученное информационно-программное изделие имеет возможности расширения за счет улучшения системы правил и алгоритмов их применения. Это позволит системе привлечь новых пользователей и оставаться конкурентно способной среди аналогов.

В результате система исправно выдает результат анализа предложений довольно сложной структуры. В будущем можно будет расширить базу правил, углубиться в тонкости русского языка, чтобы применение правил всегда было корректным и не тратило время на заведомо неверные с лексической точки зрения варианты.

Так как в результате разбора предложения присутствуют все возможные верные варианты, можно считать, что система выполняет свою задачу точного анализа.

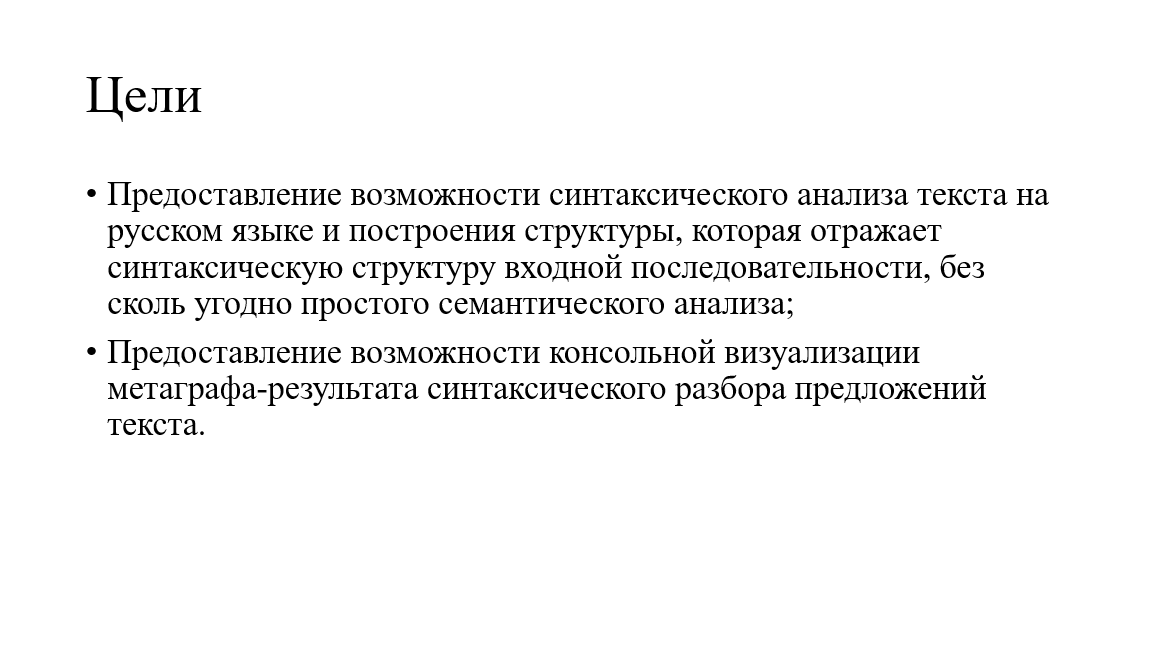
Так как результирующий метаграф анализа не хранит лишнюю информацию, оставляет возможным восстановить изначальное предложения, и обеспечивает быстрый доступ к данным, можно считать, что система выполняет свою задачу преобразования результатов анализа в метаграф.

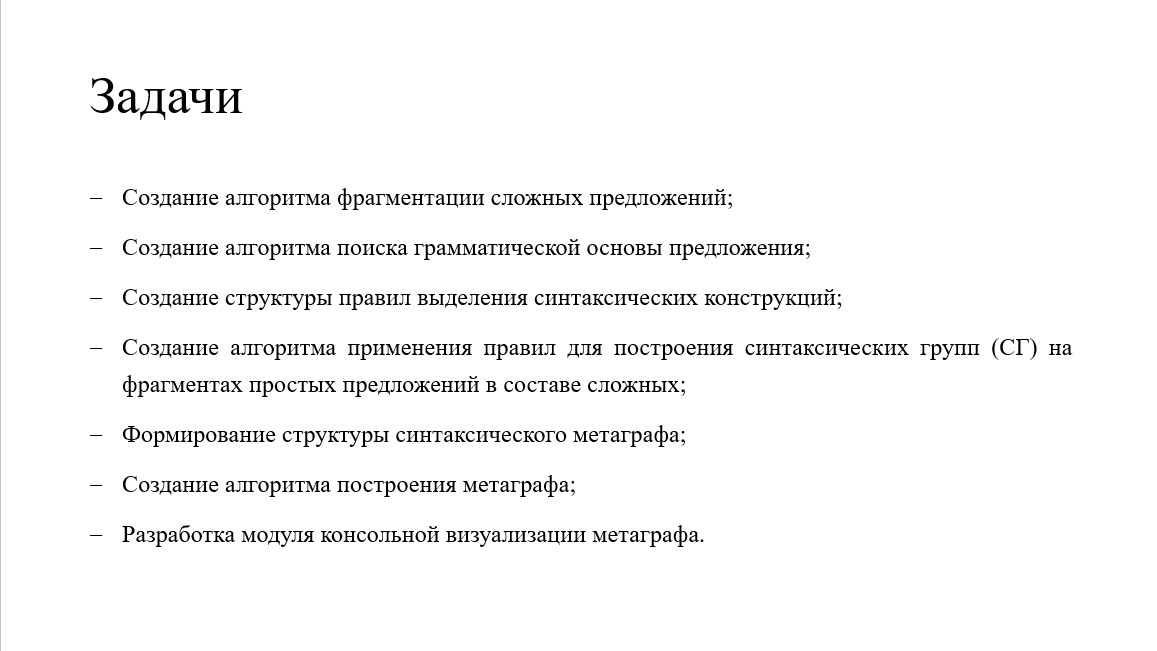
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

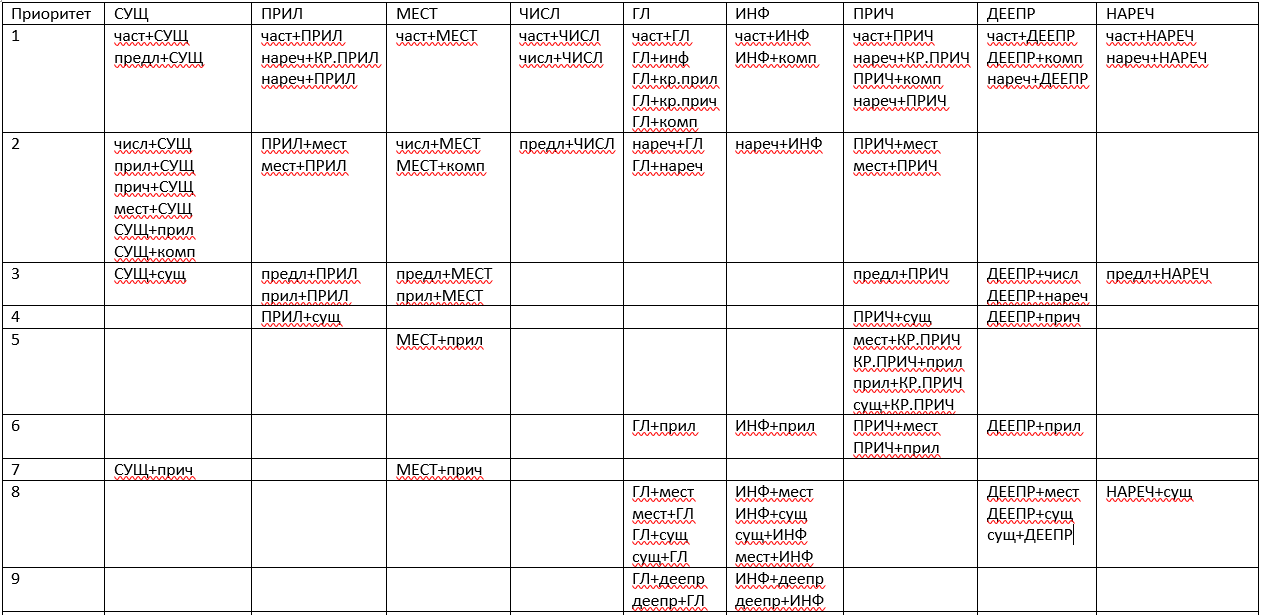
1. Волкова, И. А. Введение в компьютерную лингвистику. Практические аспекты создания лингвистических процессоров: Учебное пособие для студентов ВМиК МГУ / И. А. Волкова – Москва: Издательский отдел факультета ВМиК МГУ, 2006. – 43 с.
2. Прикладная и компьютерная лингвистика / ред. Николаев И. С., Митренина О. В., Ландо Т. М. – 2-е изд., – Москва: Издательская группа URSS, 2017. – 320 с.
3. Гапанюк, Ю. Е. Конспект лекций по спецкурсу «Гибридные интеллектуальные информационные системы на основе метаграфового подхода» / Ю. Е. Гапанюк – Москва: Издательство «Спутник +», 2018. – 56 с.
4. Тузов, В. А. Компьютерная семантика русского языка / В. А. Тузов – СПб.: Изд-во СПбГУ, 2003. – 391 с.
5. Мартынов, В. В. Основы семантического кодирования. Опыт Представления и преобразования знаний / В. В. Мартынов. – Минск: ЕГУ, 2001 г. – 140 с.
6. Проект «Открытый корпус» (OpenCorpora) [Электронный ресурс] – 2009-2020 г. – Режим доступа: http://opencorpora.org/dict.php?act=gram, свободный.
7. Морфологический анализатор pymorphy2 [Электронный ресурс] / М. Коробов – 2015-2020 г. – Режим доступа: https://pymorphy2.readthedocs.io/en/latest/index.html, свободный.
8. Постников, В. М. Основы эксплуатации АСОИиУ Том 1. Техническое обслуживание АСОИиУ. 2-е издание, переработанное и дополненное, учеб. пособие / В. М. Постников. – М.: Изд-во МГТУ им. Н. Э. Баумана, 2015. – 214 [2] c.
9. Леонтьева, Н. Н. Автоматическое понимание текстов: системы, модели, ресурсы: учеб. пособие для студ. лингв. фак. вузов / Нина Николаевна Леонтьева. – М.: Издательский центр «Академия», 2006. – 304 с.
10. ГОСТ 19.201-78. ЕСПД. Техническое задание. Требования к содержанию и оформлению.
11. ГОСТ 19.404-79. ЕСПД. Пояснительная записка. Требования к содержанию и оформлению.
12. ГОСТ 19.301-79. ЕСПД. Программа и методика испытаний. Требования к содержанию и оформлению.
13. Зализняк, А. А. Грамматический словарь русского языка: Словоизменение. 3-е изд. / М.: Изд-во «Русский язык», 1987. 880 с.
14. Хворостин, Д. В. Англо-Русский словарь лингвистических терминов / Д. В. Хворостин – Челябинск, 2007 г. – 113 с.
15. Парсер Solarix – морфологический и синтаксический анализатор русскоязычных текстов [Электронный ресурс] / Илья Козиев – 2019 г. – Режим доступа: http://www.solarix.ru, свободный.
16. Автоматическая обработка текста AOT [Электронный ресурс] – Режим доступа: http://aot.ru/docs/synan.html/, свободный.

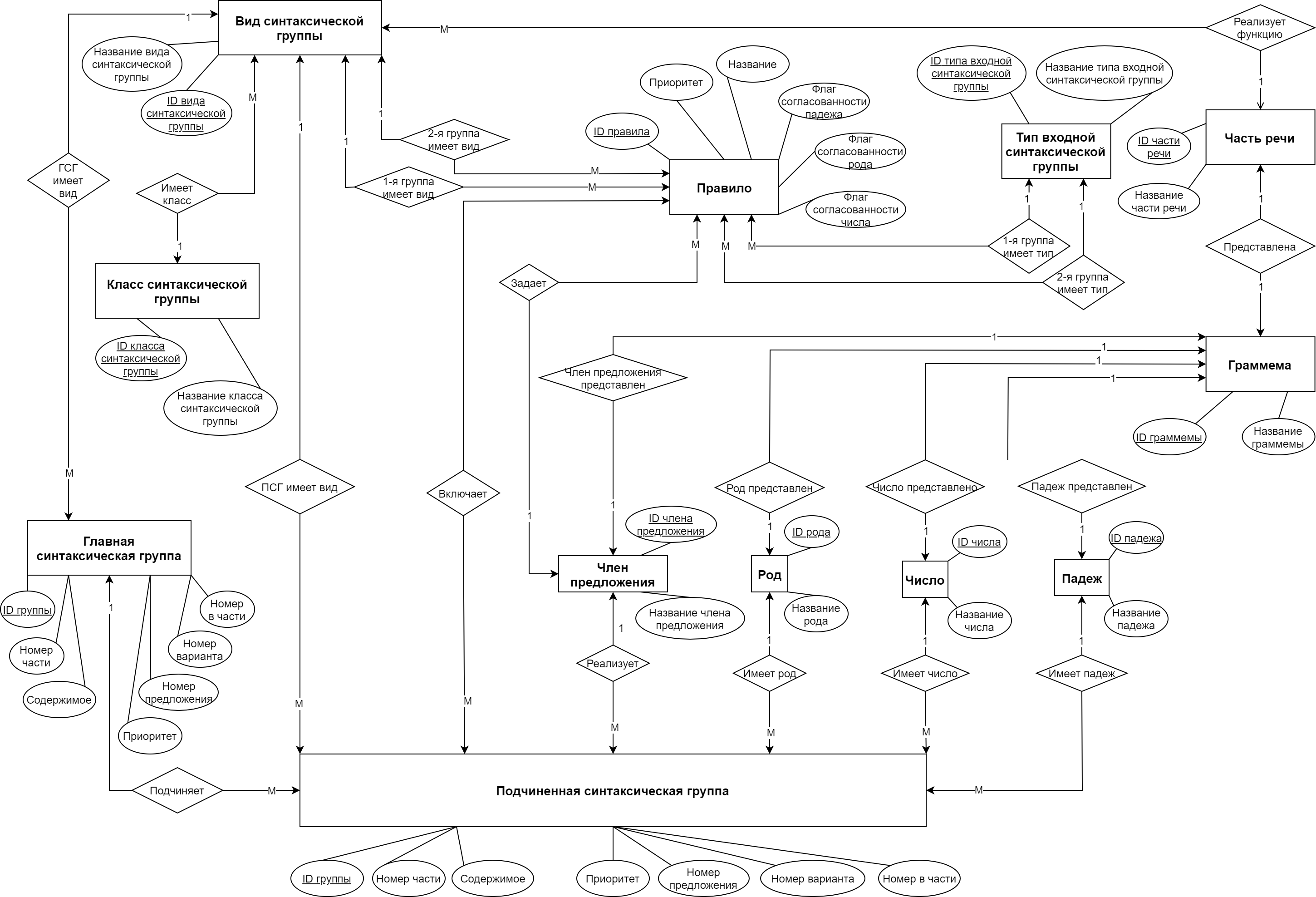
ПРИЛОЖЕНИЕ А   
ГРАФИЧЕСКИЕ МАТЕРИАЛЫ

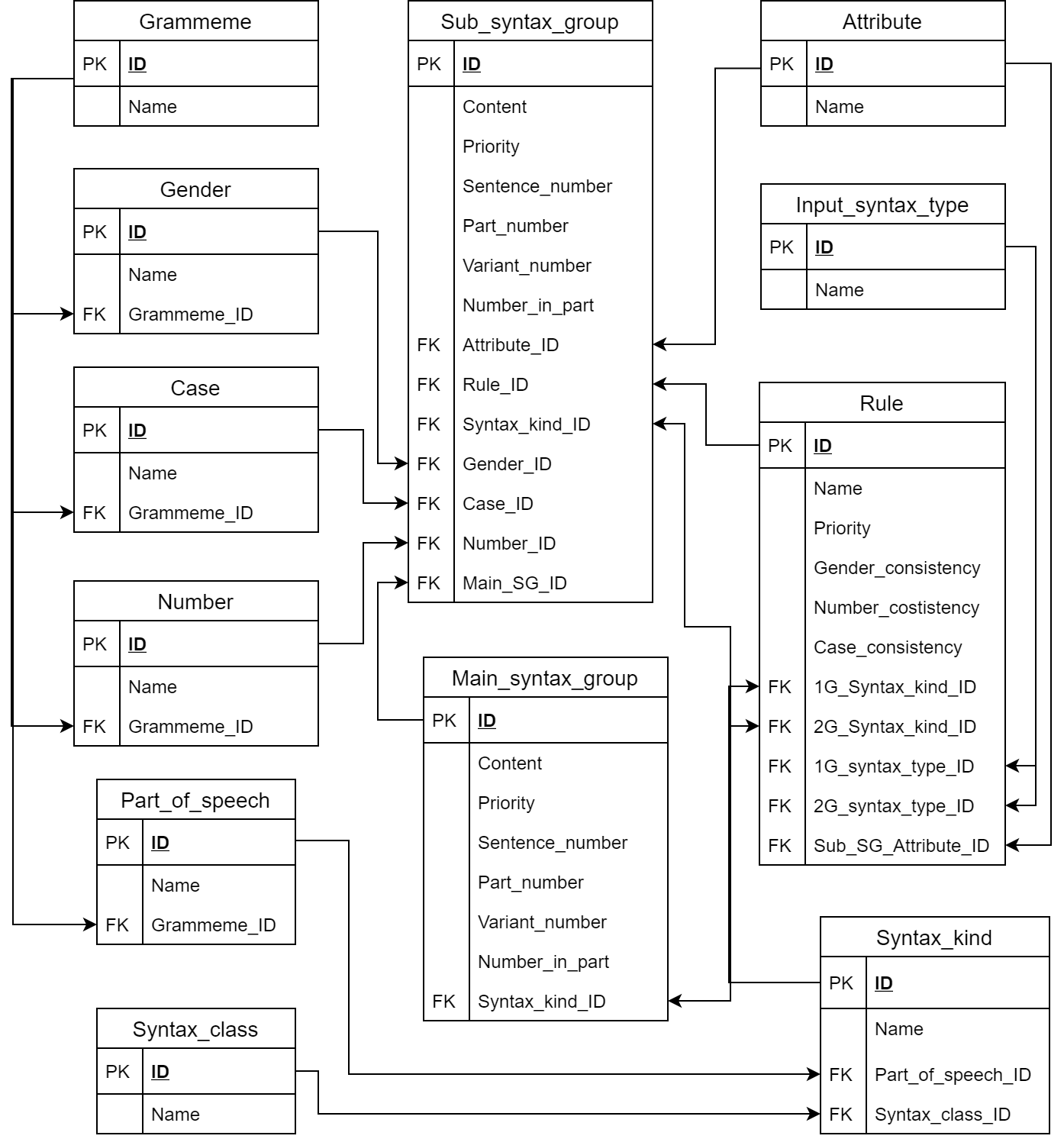
1. Цели
2. Задачи
3. Используемые словосочетания
4. Инфологическая модель
5. Даталогическая модель











ПРИЛОЖЕНИЕ В   
ТЕХНИЧЕСКОЕ ЗАДАНИЕ

**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**«Московский государственный технический университет имени Н. Э. Баумана»**

**(МГТУ им. Н. Э. Баумана)**

Утверждаю

Научный руководитель

«\_\_\_\_» 2020 г.

**Модуль формирования метаграфа на основе текстового описания**

Техническое задание

(вид документа)

Листы А4

(вид носителя)

11

(количество листов)

Исполнитель:  
студент группы ИУ5-84Б

Фадеев А.А.

« » 2020 г.

Москва, 2020 г.

**СОДЕРЖАНИЕ**

[1. Наименование 3](#_Toc40452742)

[2. Основание для разработки 3](#_Toc40452743)

[3. Исполнитель 3](#_Toc40452744)

[4. Назначение и цель разработки 3](#_Toc40452745)

[5. Содержание работы 4](#_Toc40452746)

[5.1. Задачи, подлежащие решению 4](#_Toc40452747)

[5.2. Требования к функциональным характеристикам 5](#_Toc40452748)

[5.3. Требования к составу программных компонентов 5](#_Toc40452749)

[5.4. Требования к программным компонентам 6](#_Toc40452750)

[5.5. Требования к архитектуре программного изделия 7](#_Toc40452751)

[5.6. Требования к БД 7](#_Toc40452752)

[5.7. Требования к входным и выходным данным 8](#_Toc40452753)

[5.8. Требования к надежности 8](#_Toc40452754)

[5.9. Требования к составу технических средств 9](#_Toc40452755)

[5.9.1. Требования к аппаратному обеспечению 9](#_Toc40452756)

[5.9.2. Требования к программному обеспечению 9](#_Toc40452757)

[6. Этапы разработки 10](#_Toc40452758)

[6.1. Сроки выполнения отдельных этапов работ 10](#_Toc40452759)

[7. Техническая документация, предъявляемая по окончании работы 1](#_Toc40452760)1

[8. Порядок приёма работы 11](#_Toc40452761)

[9. Дополнительные условия 11](#_Toc40452762)

1. **Наименование**

Модуль формирования метаграфа на основе текстового описания.

Шифр: Модуль «Синтаксический анализатор».

1. **Основание для разработки**

Основанием для разработки является задание на выпускную работу, подписанное руководителем выпускной работы и утверждённое заведующим кафедрой. Задание утверждено кафедрой ИУ-5 «Системы Обработки Информации и Управления» МГТУ им. Н.Э. Баумана.

1. **Исполнитель**

Студент четвёртого курса, группы ИУ5-84Б Фадеев Артём Александрович.

1. **Назначение и цель разработки**

Данный программный модуль предназначен для разработчиков программного обеспечения (ПО) для семантического анализа текста. Модуль предоставляет возможности синтаксического анализа текста на русском языке и построения его графовидной структуры. Данная структура отражает синтаксическое строение текста и хорошо подходит для его дальнейшей обработки. Таким образом, на выходе программного модуля мы имеем результат синтаксического анализа текста, представленного в виде специальной графовидной структуры.

Целью разработки является создание программного продукта, написанного на языке Python3. Данный продукт должен предоставлять разработчику функционал синтаксического анализа текста на русском языке и конвертации результатов данного анализа в объект(ы) метаграфа. В ходе анализа семантика текста не должна быть задействована. В результате анализа возможно получение нескольких метаграфов в том случае, когда по тексту нельзя сформировать единственно верный с лексической точки зрения вариант синтаксического разбора, не прибегая при этом к его сколь угодно простому семантическому анализу.

1. **Содержание работы**
   1. **Задачи, подлежащие решению**

* Исследование литературных источников;
* Определение функциональных требований;
* Сравнение аналогов и прототипов;
* Создание алгоритма фрагментации сложных предложений;
* Создание алгоритма поиска грамматической основы предложения;
* Создание структуры правил выделения синтаксических конструкций;
* Создание алгоритма применения правил для построения синтаксических групп на фрагментах простых предложений в составе сложных;
* Создание алгоритма построения метаграфа;
* Разработка структуры базы данных (БД);
* Программная реализация;
* Консольная визуализация метаграфа;
* Тестирование программы;
* Разработка конструкторской и эксплуатационной документации к программе.
  1. **Требования к функциональным характеристикам**

Разрабатываемый модуль должен выполнять следующие функции:

* + 1. Синтаксический анализ текста;
    2. Распознавание структуры предложения, нахождение зависимостей;
    3. Преобразование исходного текста в метаграф;
    4. Предоставление разработчику библиотечных функций для преобразования текста в метаграф.
  1. **Требования к составу программных компонентов**

Разрабатываемый модуль включает в себя следующие модули нижнего уровня:

* Модуль разделения текста на предложения, а предложений на слова и знаки препинания - модуль токенизации;
* Модуль синтаксического анализа;
* Модуль консольной визуализации метаграфов.
  1. **Требования к программным компонентам**

Модуль токенизации предоставляет разработчику библиотечные функции формирования нескольких блоков служебной информации, необходимой для непосредственно синтаксического анализа: разделение текста на абзацы, предложения, лексемы; поиск грамматической основы, союзов и предлогов, обработка которых позволяет отсеять множество заведомо неверных вариантов морфологического разбора слов, состоящих в предложении. Продукт не предусматривает обработки орфографических или пунктуационных ошибок в тексте.

Модуль синтаксического анализа.

По исходному предложению с учётом результатов работы модуля токенизации и данных, хранящихся в БД, формирует структуру правил для анализа. Средствами СУБД может осуществляться настройка и добавление правил для более точного и однозначного распознавания структуры предложения. Для работы модуля необходимо, чтобы на компьютере был установлен определённый перечень программных библиотек языка Python, которые не входят в базовый набор при установке на компьютер самого Python версии 3.x и выше.

Модуль консольной визуализации.

Модуль обеспечивает пользователю возможность вводить исходные данные с клавиатуры или из файла и анализировать результат, отображаемый на экране и записываемый в отдельный файл.

* 1. **Требования к архитектуре программного изделия**

В данном проекте используется следующая архитектура:

* программный модуль, обеспечивающий выполнение алгоритмов синтаксического анализа, построения метаграфа;
* база данных, обеспечивающая хранение данных, необходимых для обработки текста.

Приложение используется в автономном режиме, т.е. без связи с интернетом или с какой-либо другой локальной или глобальной сетью.

* 1. **Требования к БД**

База данных должна хранить информацию о правилах согласования различных синтаксических групп, служебную информацию, позволяющую классифицировать эти правила, части речи и члены предложения, к которым относятся данные правила. Также БД должна работать под управлением реляционной СУБД. Для работы с программой пользователю нужно иметь на своём компьютере предустановленную СУБД, указанную в пункте 5.9.2 этого документа, а также файлы данной БД. Таким образом, пользователь работает на своем компьютере с локальной копией данной БД.

* 1. **Требования к входным и выходным данным**

В режиме использования модуля:

Входными данными являются:

* Текст из файла или текст, вводимый оператором вручную в командной строке во время работы программы;
* ответы БД на SQL-запросы оператора;

Выходными данными являются:

* SQL-запросы к БД;
* структура входного предложения в формате метаграфа.
  1. **Требования к надежности**

Программа должна обрабатывать ошибки попыток анализа, например, когда введено пустое предложение, когда по введенному предложению или случайному набору слов невозможно провести цельный синтаксический анализ. О подобных «ошибочных» случаях пользователь уведомляется в процессе работы программы. Программа должна обеспечить устойчивое функционирование при попытках проанализировать один или несколько разных текстов. Программа должна выдавать один и тот же результат при вводе одного и того же исходного текста. Так как взаимодействие пользователя с БД напрямую не предполагается, то при перезапуске программы или БД не требуется специальных действий для восстановления работы программы. При невозможности получить доступ к БД программа останавливает работу и уведомляет пользователя о необходимости восстановить файлы БД.

* 1. **Требования к составу технических средств**
     1. **Требования к аппаратному обеспечению**
* процессор Intel Core i3 или более мощный (или совместимый с ним);
* рекомендуемый объем оперативной памяти 2048 мегабайт (МБ);
* минимальный объем 1024 МБ;
* 100 MB свободного места на жестком диске для системы и от 500 МБ до 1 ГБ для приложения, базы данных и демонстрационных файлов;
* монитор;
* клавиатура;
* мышь или совместимое указывающее устройство.
  + 1. **Требования к программному обеспечению**

На компьютере должно быть установлено следующее ПО:

* одна из следующих ОС: Windows, Linux, или Mac OS X;
* интерпретатор Python 3;
* СУБД MySQL версии 5.7 и выше.

1. **Этапы разработки**
   1. **Сроки выполнения отдельных этапов работ**

График приведен в соответствии с учебными неделями МГТУ им. Баумана 4 курса 8 семестра.

| **№№** | **Наименование этапа** | **Содержание работы** | **Дата исполнения** |
| --- | --- | --- | --- |
| 1. | Анализ ПО | Анализ задач, подлежащих реализации | 15 марта 2020г. |
| 2. | Составление ТЗ | Формулировка требований, составление технического задания | 19 марта 2020г. |
| 3. | Разработка моделей данных | Разработка инфологической модели, выбор СУБД, разработка даталогической модели | 30 марта 2020г. |
| 4. | Создание программного обеспечения | Разработка алгоритмов работы программы и пользовательских интерфейсов | 23 мая 2020г. |
| 5. | Оформление документации | Подготовка технического задания и остальной документации, включая графическую часть | 23 мая 2020г. |
| 6. | Предзащита | Показ работающей программы | 24 мая 2020г. |
| 7. | Защита работы | Защита | ~13 июня 2020г. |

1. **Техническая документация, предъявляемая по окончании работы**

* Техническое задание;
* Руководство оператора;
* Программа и методика испытаний;
* Текст программы;
* Расчетно-пояснительная записка.

1. **Порядок приёма работы**

Приём и контроль программного изделия осуществляется в соответствие с программой и методикой испытаний (см. документ Программа и методика испытаний).

1. **Дополнительные условия**

Данное техническое задание может уточняться в установленном порядке.

ПРИЛОЖЕНИЕ С   
ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**«Московский государственный технический университет имени Н. Э. Баумана»**

**(МГТУ им. Н. Э. Баумана)**

Утверждаю

Научный руководитель

«\_\_\_\_» 2020 г.

**Модуль формирования метаграфа на основе текстового описания**

Программа и методика испытаний

(вид документа)

Листы А4

(вид носителя)

8

(количество листов)

Исполнитель:  
студент группы ИУ5-84Б

Фадеев А.А.

« » 2020 г.

Москва, 2020 г.

**АННОТАЦИЯ**

В данном документе описываются последова­тельность и методы проведения испытаний при тес­тировании программного изделия, состав и струк­тура технических и программных средств, необхо­димых для проведения испытаний, а также приво­дятся требования к предъявляемой документации, характеристикам программы применительно к условиям эксплуатации и требования к информационной и программной совместимости. Описываются тестовые примеры и реакция системы на них.

**СОДЕРЖАНИЕ**

[АННОТАЦИЯ 2](#_Toc43145312)

[1. Объект испытаний 4](#_Toc43145313)

[2. Цель испытаний 4](#_Toc43145314)

[3. Состав предъявляемой документации 4](#_Toc43145315)

[4. Технические требования 5](#_Toc43145316)

[4.1. Требования к программной документации 5](#_Toc43145317)

[4.2. Требования к техническим характеристикам 5](#_Toc43145318)

[5. Порядок проведения испытаний 6](#_Toc43145319)

[5.1. Требования к составу аппаратного обеспечения 6](#_Toc43145320)

[5.2. Требования к составу программного обеспечения](#_Toc43145321) 7

[6. Методы испытаний 7](#_Toc43145322)

1. **Объект испытаний**

Модуль формирования метаграфа на основе текстового описания.

Шифр: Модуль «Синтаксический анализатор».

1. **Цель испытаний**

Цель испытания – проверка функционирования всех указанных в техническом задании функций программы.

1. **Состав предъявляемой документации**

На испытании программного продукта предъявляются следующие документы:

* Техническое задание
* Программа и методика испытаний

1. **Технические требования**
   1. **Требования к программной документации**

Комплектность программной документации должна удовлетворять разделу данного документа "Состав предъявляемой документации".

* 1. **Требования к техническим характеристикам**
     1. **Требования к составу аппаратного обеспечения**
* процессор Intel Core i3 или более мощный (или совместимый с ним);
* рекомендуемый объем оперативной памяти 2048 мегабайт (МБ);   
  минимальный объем 1024 МБ;
* 100 MB свободного места на жестком диске для системы и от 500 МБ до 1ГБ для приложения, базы данных и демонстрационных файлов;
* монитор;
* клавиатура;
* мышь или совместимое указывающее устройство.
  + 1. **Требование к составу программного обеспечения**

На компьютере должно быть установлено следующее ПО:

* одна из следующих ОС: Windows, Linux, или Mac OS X;
* интерпретатор Python 3;
* СУБД MySQL версии 5.7 и выше.

1. **Порядок проведения испытаний**

Испытания данного программного продукта будут проводиться в следующем порядке:

1. Запуск системы. Открываем папку с проектом и запускаем его во встроенном терминале.
2. Тестирование базовых операций системы (в соотв. с таблицей п.6)
   1. **Требования к составу аппаратного обеспечения**

Требования к составу аппаратного обеспечения учитываются согласно документу программа и методика испытаний, пункт 4.2.1.

* 1. **Требования к составу программного обеспечения**

Требования к составу программного обеспечения учитываются согласно документу программа и методика испытаний, пункт 4.2.2.

1. **Методы испытаний**

|  |  |  |  |
| --- | --- | --- | --- |
| **N** | **пункт ТЗ** | **Выполняемые действия** | **Результат** |
| 1 | 5.2.1. Синтаксический анализ текста;  5.2.2. Распознавание структуры предложения, нахождение зависимостей;  5.2.3. Преобразование исходного текста в метаграф;  5.2.4. Предоставление библиотечных функций для преобразования текста в метаграф. | Открыть терминал (командную строку) и перейти в директорию проекта. | Текущая директория будет совпадать с директорией проекта, в которой находиться исполняемый файл «make\_metagraph.py» |
| Выполнить команду:  python make\_metagraph.py | На экране появится сообщение о просьбе ввести исходный текст ручным вводом с клавиатуры или из файла |
| Ввести «вручную», а затем простое нераспространённое предложение («Наступило лето») в поле ввода, нажать на кнопку «Ввод» на клавиатуре | Через некоторое время в директории появится файл result.txt. На экране отобразится терминальное отображение полученного метаграфа, где «лето» будет метавершиной группы подлежащего, а «наступило» - сказуемого. |
| Ввести простое распространённое предложение («Сегодня лил сильный дождь») в поле ввода, нажать на кнопку «Ввод» на клавиатуре | Через некоторое время в директории появится файл result.txt. На экране отобразится терминальное отображение полученного метаграфа, где «сегодня» будет вершиной внутри метавершины группы сказуемого, а «сильный» - подлежащего. |
| Ввести сложное предложение с двумя грамматическими основами («Отдыхающий насорил на пляже, поэтому был оштрафован») в поле ввода, нажать на кнопку «Ввод» на клавиатуре | Через некоторое время в директории появится файл result.txt. На экране отобразится терминальное отображение полученного метаграфа, где будет два метаграфа: «Отдыхающий насорил на пляже» и «был оштрафован». Союз «поэтому» будет являться ребром, связывающим два простых предложения в составе сложного. |