МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

КУРСОВАЯ РАБОТА

по дисциплине «Введение в нереляционные базы данных»

Тема: Анализатор студ отчетов по упоминаемости слов и фраз

Студент гр. 2303 _____ Волков И.С.

Студент гр. 2303 _____ Мышкин Н.В.

Студент гр. 2300 _____ Рогожин К.Д.

Студент гр. 2300 _____ Пахомов С.Д.

Студент гр. 2300 _____ Локосов Д.Д.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2025

ЗАДАНИЕ
НА КУРСОВУЮ РАБОТУ

Студент Волков И.С. 2303

Студент Мышкин Н.В. 2303

Студент Рогожин К.Д. 2300

Студент Пахомов С.Д. 2300

Студент Локосов Д.Д. 2300

Тема работы: Анализатор студ отчетов по упоминаемости слов и фраз

Исходные данные:
Сделать сервис, который принимает на вход дипломные работы (docx), парсит их содержимое и разбивает текст по разделам / подразделам, считает статистики (когда и какие слова употребляются, где много шумовых слов и тд), анализирует тексты на схожесть, водность, степень раскрытия отдельных тем / задач. Используемая БД – neo4j.

Содержание пояснительной записки:
«Содержание», «Введение», «Сценарии использования», «Модель данных», «Разработанное приложение», «Заключение», «Список использованных источников»

Предполагаемый объем пояснительной записки:
Не менее 15 страниц.

Дата выдачи задания: 05.02.2025

Дата сдачи реферата: 29.05.2025

Дата защиты реферата: 29.05.2025

Студент гр. 2303	_____	Волков И.С.
Студент гр. 2303	_____	Мышкин Н.В.
Студент гр. 2300	_____	Рогожин К.Д.
Студент гр. 2300	_____	Пахомов С.Д.
Студент гр. 2300	_____	Локосов Д.Д.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках работы было разработано приложение для анализа студенческих дипломов на предмет различных метрик. Дипломы хранятся в виде графов в БД neo4j. Данный сервис нацелен на парсинг дипломов в виде docx с визуальным представление в виде иерархической структуры, где можно посмотреть статистику как по диплому в целом, так и по каждому разделу/подразделу.

SUMMARY

As part of the work, an application was developed to analyze student diplomas for various metrics. Diplomas are stored as graphs in the neo4j database. This service is aimed at parsing diplomas in the form of docx with a visual representation in the form of a hierarchical structure, where you can view statistics on both the diploma as a whole and for each section/subsection.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Решаемая проблема актуальна, поскольку парсинг и анализ дипломов, их сравнение с другими дипломами, могут помочь как студентам, так и преподавателям для скорого анализа содержимого дипломов.

Задача: реализовать сервис, принимающий от пользователей дипломы в виде .docx файлов, а затем предоставляющий краткую статистику по диплому в целом и по каждому разделу/подразделу. Также сервис должен позволять проводить поиск по дипломам и разделам по нескольким фильтрам. Реализовать возможность импорта/экспорта содержимого БД.

Решение: использовать Neo4j для графового представления дипломов в БД для возможности легкой работы с деревьями, а также Flask в связке с NLTK для реализации сервисной части приложения, в том числе морфологических анализаторов.

Качественные требования к решению включают: цепочка действий «экспорт-импорт» должна инициализировать то же содержимое, что было до действий; статистика диплома должна представляться в виде вложенного аккордеона; поиск выводит содержимое в виде таблиц с краткой информацией о сущности с возможностью перехода на соответствующий диплом.

1. СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

1.1. Макет UI

Макет приложения содержит страницу для загрузки диплома, вывода его статистики, а также страницы просмотра содержимого БД с возможностью сортировки и фильтрации, страницу импорта/экспорта БД и страницу анализа содержимого БД. Полученный макет см. на рис. 1.

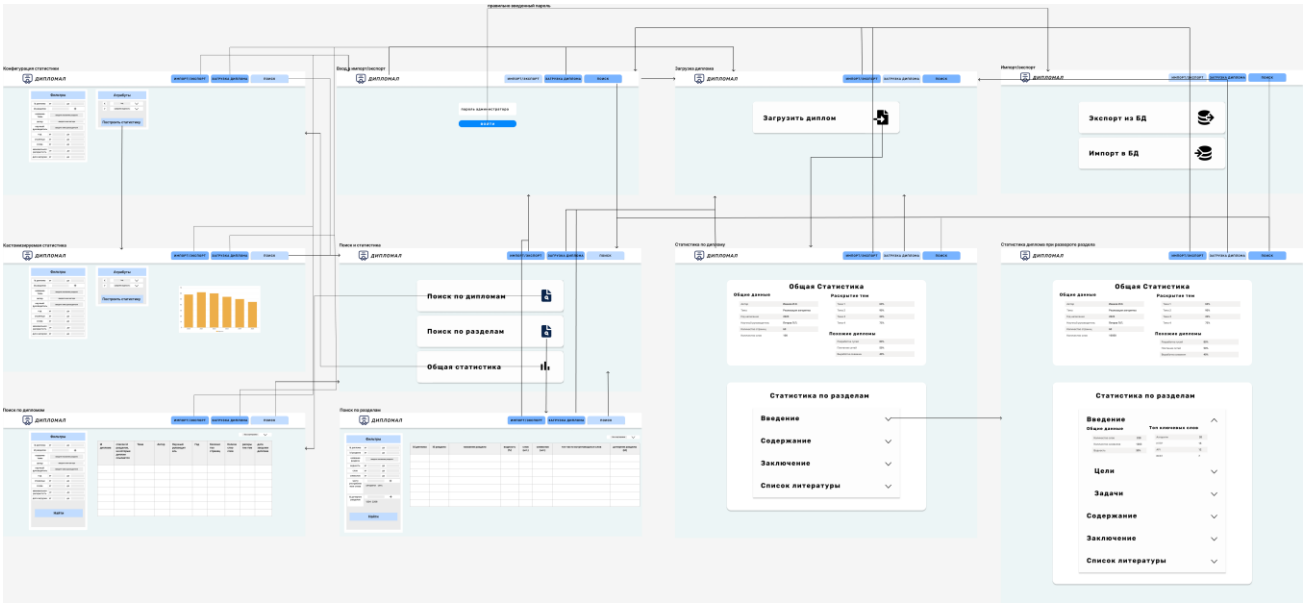

Рисунок 1 – Макет UI

1.2. Сценарии использования для задачи

- Пользователь заходит на основную страницу приложения, загружает диплом в формате .docx. После чего его перебрасывает на страницу с статистикой исследования диплома, в которой приведены основные сведения о дипломе в целом:
    - Тема
    - Автор
    - Год написания
    - Научный руководитель
    - Количество страниц
    - Количество слов
    - Раскрытие основных поставленных тем

- Самые схожие дипломы

Также дана информация по каждому смысловому разделу:

- Количество слов
- Количество символов
- Водность
- Самые частые ключевые слова

При этом в каждом разделе есть поля для раскрытия информации о его подразделах.

- Пользователь заходит в приложение, переходит на страницу "ИМПОРТ/ЭКСПОРТ". На ней необходимо ввести пароль администратора для входа на соответствующую страницу, где можно по нажатию соответствующей кнопки либо сохранить содержимое БД в одном JSON файле, либо заполнить БД уже полученным прежде файлом;
- Пользователь заходит в приложение, переходит на страницу "ПОИСК". И выбирает, по какой сущности будет осуществляться поиск.
    - Поиск дипломов выводит информацию о всех дипломах в виде таблицы, включая их идентификаторы и идентификаторы дочерних разделов. Можно осуществить фильтрацию по каждому из полей, причем для фильтрации разделов можно добавить идентификаторы разделов списком (будут оставлены те дипломы, которые ссылаются хотя бы на один раздел из списка), а также фильтровать по минимальной раскрытости тем. Также возможна сортировка по id диплома, году, количеству страниц/слов, дате загрузки;
    - Поиск разделов схож с поиском дипломов, но он так же включает идентификатор родительского диплома, также можно фильтровать по списку часто употребляемых слов. Возможна фильтрация по id раздела, id диплома, водности, количеству слов/символов.
- Пользователь заходит в приложение, переходит на страницу "ПОИСК", а с неё на страницу кастомизируемой статистики. Пользователь

может отфильтровать данные подобно фильтру поиска дипломов, а также выбрать атрибуты по оси X и Y.

- X: год написания, научный руководитель;
- Y: год написания, научный руководитель, средняя водность, средняя раскрытость тем, среднее количество страниц/слов.

В итоге можно сделать вывод, что преобладают операции чтения, поскольку дипломы загружаются довольно редко, причем для каждого загруженного диплома требуется чтение БД для извлечения информации о схожести. Также Операции чтения дипломов и разделов требуют извлечения дочерних разделов.

2. МОДЕЛЬ ДАННЫХ

2.1. Нереляционная модель

Графическое представление

Графическое представление модели см. на рис. 2.

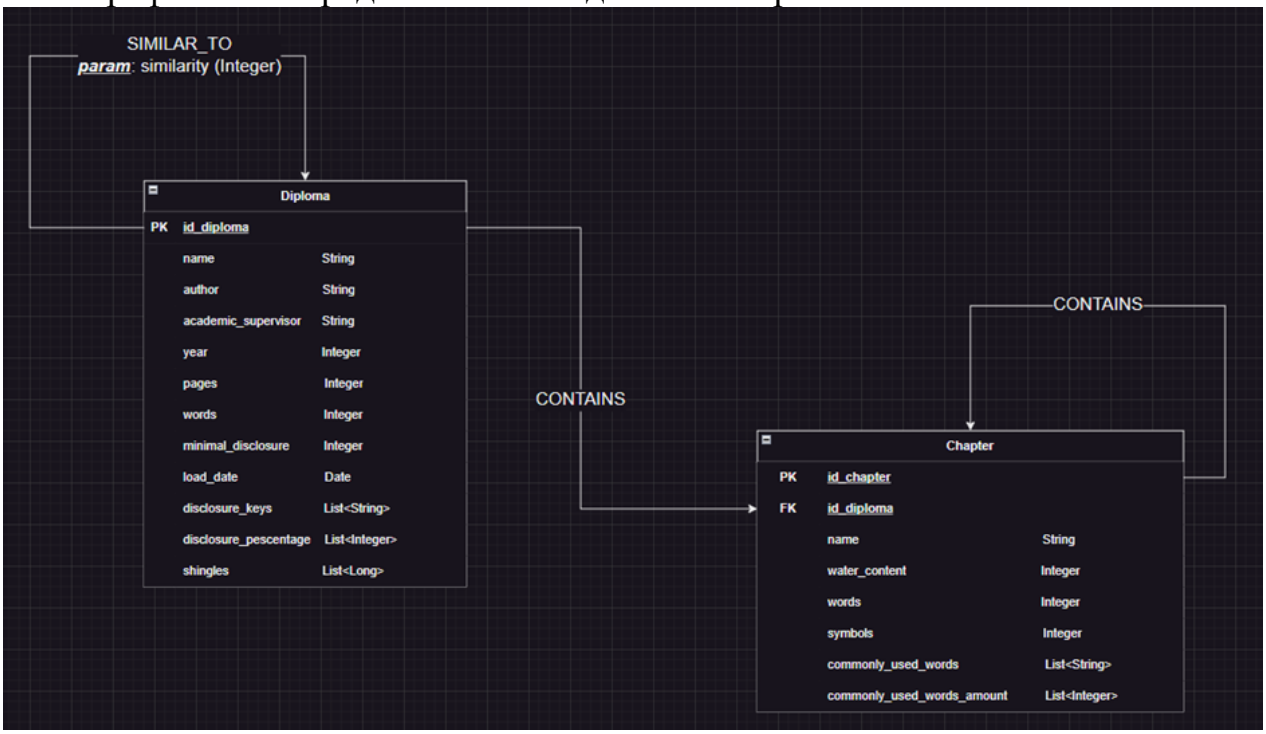

Рисунок 2 – Графическое представление нереляционной модели

Описание назначений коллекций, типов данных и сущностей

Общее описание коллекций

Коллекция Diploma

Основная коллекция, содержащая информацию о дипломной работе, включая метаданные и статистику.

Поля:

- **id** - Уникальный идентификатор диплома.
- **name** - Тема диплома.
- **author** - Автор работы.
- **academic_supervisor** - Научный руководитель.
- **year** - Год защиты.
- **pages** - Количество страниц.

- **words** - Общее количество слов.
- **minimal_disclosure** - Уровень минимального раскрытия тем, необходим для ускорения фильтрации.
- **load_date** - Дата загрузки в систему.
- **disclosure_keys** - Список раскрываемых в дипломе тем.
- **disclosure_persentage** - Список процентных значений раскрытия по каждому элементу **disclosure_keys**.
- **shingles** - Упорядоченный список шинглов, необходимый для ускорения сравнения разных дипломов.

Коллекция Chapter

Описывает свойства каждого раздела диплома, включая статистику текста, характеристики текста и ключевые элементы.

Поля:

- **id** - Уникальный идентификатор раздела.
- **id_diploma** - Идентификатор диплома, которому принадлежит раздел, нужен для ускорения извлечения данного поля без необходимости поиска корня дерева для каждого раздела (стоит учесть, что id родительского диплома неизменяем).
- **name** - Название раздела.
- **water_content** - Степень "водности" текста.
- **words** - Количество слов в разделе.
- **symbols** - Количество символов в разделе.
- **commonly_used_words** - Список наиболее часто встречающихся ключевых слов в тексте раздела.
- **commonly_used_words_amount** - Список количеств повторений для соответствующих слов из **commonly_used_words**.

Связи:

- **SIMILAR_TO** - Связь, описывающая схожесть с другим дипломом **[Diploma->Diploma]**

- **similarity** - Процент схожести между дипломами.

- **CONTAINS** - Связь, описывающая входящие в **[диплом/раздел]** разделы **[Diploma->Chapter/Chapter->Chapter]**

Оценка объема сущностей и связей

Коллекция Diploma (см. табл. 1)

Таблица 1 – Оценка объема Diploma

Поле	Тип данных	Размер(байт)	Примечание
id	-	8	Уникальный идентификатор
name	String	100	В среднем 100 байтовых символов
author	String	30	Максимум 30 байтовых символов
academic_supervisor	String	30	Максимум 30 байтовых символов
year	Integer	8	-
pages	Integer	8	-
words	Integer	8	-
minimal_disclosure	Integer	8	-
load_date	Date	8	-
disclosure_keys	List<String>	500	В среднем 5 тем по 100 байт
disclosure_persentage	List<Integer>	40	5 чисел по 8 байт
shingles	List<Integer>	80000	10000 шинглов по 8 байт

Итого размер элемента: 80,748 байт

Коллекция Chapter (см. табл. 2)

Таблица 2 – Оценка объема Chapter

Поле	Тип данных	Размер	Примечание
id	-	8	Уникальный идентификатор
id_diploma	-	8	-
name	String	100	В среднем 100 символов
water_content	Integer	8	-
words	Integer	8	-
symbols	Integer	8	-
commonly_used_words	List<String>	100	5 самых частых слов по 20 байт
commonly_used_words_amount	List<Integer>	40	5 чисел по 8 байт

Итого размер элемента: 280 байт

Связи между сущностями

Связь CONTAINS (см. табл. 3)

Таблица 3 – Оценка объема CONTAINS

Компонент	Размер (байт)
Указатель на начальный узел	8
Указатель на конечный узел	8

Итого размер связи: 16 байт

Связь SIMILAR_TO (см. табл. 4)

Таблица 4 – Оценка объема SIMILAR_TO

Компонент	Размер (байт)
Указатель на начальный узел	8
Указатель на конечный узел	8
similarity	8

Итого размер связи: 24 байта

Оценка объема информации, хранимой в модели

Формула общего объема памяти

Общий объем памяти базы данных оценивается по формуле: $V = V_D \times N_D + V_C \times N_C \times N_D + n_C \times v_C + n_S \times v_S$

Где:

- V_D - усредненное значение размера сущности Diploma (80748Б)
- N_D - количество сущностей Diploma
- V_C - усредненное значение размера сущности Chapter (280Б)
- N_C - среднее количество сущностей Chapter в дипломе
- v_C - размер связи CONTAINS (16Б)
- n_C - количество CONTAINS
- v_S - размер связи SIMILAR_TO (24Б)
- n_S - количество SIMILAR_TO

Для вычисления числа связей между дипломами воспользуемся формулой вычисления числа ребер в полном графе:

$n_D = N_D \times (N_D - 1)/2$

Число заголовков дипломных работ можно оценить, исходя из предположения, что средняя глубина вложенности заголовков в дипломе составляет 3 уровня. При этом: разделов первого уровня в среднем $C_0 = 3$, каждый раздел содержит в среднем $C_1 = 3$ подразделов (уровень 2), каждый подраздел (уровень 2) содержит в среднем $C_2 = 3$ подподразделов (уровень 3). Получаем:

$N_C = C_0 \times (1 + C_1 + C_1 \times C_2) \times N_D = 3 \times (1 + 3 + 3 \times 3) \times N_D = 39 \times N_D$

Общее число связей **CONTAINS** для одного диплома составляет: $n_C = N_c = 39 \times N_D$

Пользуясь приведенными рассуждениями, получим итоговую формулу для вычисления необходимого объема памяти для хранения N_D дипломов в базе данных:

$$V = 80748 \times N_D + 280 \times 39 \times N_D + \frac{N_D(N_D - 1)}{2} \times 24 + 39 \times 16 \times N_D = 12 \times N_D^2 + 92280 \times N_D$$

Избыточность данных

Для оценки избыточности модели вычислим объем "чистых" данных, исключив дублирующую информацию:

В узловых разделах (не терминальных главах)

Избыточные поля, которые могут быть вычислены из дочерних разделов:

Таблица 5 – Избыточные поля узловых разделов

Поле	Тип	Размер	Описание	Комментарий
words	Integer	8	Общее количество слов в разделе	Сумма из полей **words** дочерних разделов
symbols	Integer	8	Общее количество символов	Сумма полей **symbols** подразделов
water_content	Integer	8	Водность текста	Средневзвешенное по подразделам

Таблица 6 - Избыточность в сущности Diploma

Поле	Тип	Размер
minimal_disclosure	Integer	8 байт

Таблица 7 - Избыточность в сущности Chapter

Поле	Тип	Размер	Описание
id_diploma	Integer	8 байт	Ссылка на родительский диплом

Формула избыточности:

$$\frac{V}{V_{\text{чистые}}} = \frac{V}{V - N_D \times \left[\left(n_{\text{words}} + n_{\text{sym}} + \text{water}\right) \times n_{\text{nodes}} + V_{\text{idDiploma}} \times N_C + \text{disclosure}\right]}$$

Где:

Параметр	Описание	Значение
n_{words}	Размер поля "количество слов"	8 байт
n_{sym}	Размер поля "количество символов"	8 байт
water	Размер поля "водность"	8 байт
n_{nodes}	Среднее количество не листовых разделов	16
$V_{idDiploma}$	Размер поля id_diploma	8 байт
disclosure	Размер поля minimal_disclosure	8 байт
N_C	Количество разделов	$39 \times N_D$ байт

$$\frac{V}{V_{\text{чистые}}} = \frac{V}{V - N_D \times [(8+8+8) \times 16 + 8 \times 39 + 8]} = \frac{V}{V - N_D \times 710} = \frac{12 \times N_D + 92280}{12 \times N_D + 91570}$$

Направление роста модели

Получим формулу зависимости величины объема данных от количества объектов каждой сущности:

$$V = V_D \times N_D + V_C \times N_C = 80748 \times N_D + 280 \times N_C$$

Основной вклад в рост объёма памяти вносит увеличение числа дипломных работ, поскольку каждая из них имеет значительный размер (80 748

байта). Дополнительным фактором является каскадный эффект: каждая новая работа сопровождается несколькими разделами (по 280 байт каждый). Таким образом, при добавлении одной дипломной работы общий объем данных возрастает не только за счёт её собственного размера, но и за счёт связанных с ней разделов.

Примеры запросов

Под **?** имеется в виду поступаемая информация извне, под **<name>** - полученная информация из БД. **N** - количество дипломов в БД.

Основной сценарий использования

Основной сценарий использования предполагает загрузку дерева документа в БД и сравнение с другими дипломами. В среднем **N + 46** запросов, задевая все коллекции.

1. Загрузка диплома

```
CREATE (d:Diploma {
    name: ?,
    author: ?,
    academic_supervisor: ?,
    year: ?,
    disclosure_persentage: ?,
    shingles: ?
})
RETURN ID(d);
```

Данный запрос возвращает id созданного диплома для дальнейших действий с ним (1 запрос).

2. Загрузка внешних разделов диплома (введение, содержание и т.д.)

В среднем нужно создать 3 раздела и 1 запросом связать их (4 запроса).

Создание раздела:

```
CREATE (c:Chapter {
    id_diploma: ?,
    name: ?,
    water_content: ?,
    words: ?,
    symbols: ?,
    commonly_used_words: ?[],
    commonly_used_words_amount: ?[]
})
RETURN ID(c);
```

В данном запросе возвращается id созданного раздела для дальнейших действий с ним.

Связывание с дипломом:

```
MATCH (d:Diploma), (c:Chapter)
WHERE ID(d) = :id_diploma AND c.id_diploma = :id_diploma
CREATE (d)-[:CONTAINS]->(c);
```

3. Загрузка подразделов

Загрузка подразделов подобна загрузке внешних разделов за исключением связи с разделом. В среднем 36 подразделов, поэтому 36 запросов на создание и 1 на связь (37 запросов).

Связывание подразделов:

```
MATCH (c1:Chapter), (c2:Chapter)
WHERE ID(c1) = :id_chapter AND ID(c2) IN :id_chapters
CREATE (c1)-[:CONTAINS]->(c2);
```

4. Извлечение списка шинглов (1 запрос)

```
MATCH (d:Diploma)
WHERE ID(d) <> :id_diploma
RETURN ID(d), d.shingles;
```

5. Занесение результатов сравнения (N запросов)

```
MATCH (d1:Diploma), (d2: Diploma)
WHERE ID(d1) = :id_diploma AND ID(d2) = :id_diploma_2
CREATE (d1)-[:SIMILAR_TO {similarity: ?}]->(d2);
```

6. Извлечение диплома и его разделов (3 запроса)

```
MATCH p = (d:Diploma)-[r:CONTAINS*0..]->(x)
WHERE ID(d) = :id_diploma
RETURN COLLECT(DISTINCT ID(x)) as nodes, [r IN COLLECT(DISTINCT LAST(r))
| [ID(startNode(r)), ID(endNode(r))]] as rels;

MATCH (d:Diploma)
WHERE ID(d) = :id_diploma
RETURN d;

MATCH (c:Chapter)
WHERE ID(c) IN :chapters
RETURN c;
```

Поиск по дипломам с фильтрацией и сортировкой.

Выполняется 1 запрос, задевая коллекцию Diploma.

```
MATCH (d:Diploma)-[:CONTAINS]->(c:Chapter)
WITH d, COLLECT(ID(c)) AS chapters
WHERE ID(d) >= :min_id AND ID(d) <= :max_id
AND toLower(d.name) CONTAINS toLower(:name)
AND toLower(d.author) CONTAINS toLower(:author)
AND toLower(d.academic_supervisor) CONTAINS toLower(:academic_supervisor)
AND d.year >= :min_year AND d.year <= :max_year
AND d.pages >= :min_pages AND d.pages <= :max_pages
AND d.words >= :min_words AND d.words <= :max_words
AND d.minimal_disclosure >= :min_minimal_disclosure AND
d.minimal_disclosure <= :max_minimal_disclosure
AND d.load_date >= :min_date AND d.load_date <= :max_date
AND ANY(chapter IN :chapters WHERE chapter IN d.chapters)
ORDER BY id/year/pages/words/load_date
LIMIT K SKIP M
RETURN d{.*, chapters: chapters} AS diploma;
```

Поиск по разделам с фильтрацией и сортировкой.

Выполняется 1 запрос, задевающий Chapter.

```
MATCH (c:Chapter)-[:CONTAINS*0..]->(c1:Chapter)
WITH c, COLLECT(ID(c1)) AS chapters
WHERE ID(c) >= :min_id AND ID(c) <= :max_id
AND c.id_diploma >= :min_id_diploma AND c.id_diploma <= :max_id_diploma
AND toLower(c.name) CONTAINS toLower(:name)
AND c.words >= :min_words AND c.words <= :max_words
AND c.symbols >= :min_symbols AND c.symbols <= :max_symbols
AND c.water_content >= :min_water_content AND c.water_content <=
:max_water_content
AND ANY(word IN :words WHERE word IN c.commonly_used_words)
AND ANY(chapter IN :chapters WHERE chapter IN chapters)
ORDER BY id/id_diploma/water_content/words/symbols
LIMIT K SKIP M
RETURN c {.*, chapters: chapters} AS chapter;
```

Анализ содержимого для года и научного руководителя.

Выполняется 1 запрос по Diploma.

```
MATCH (d:Diploma)
WITH d.academic_supervisor/d.year AS groupKey1,
d.year/d.academic_supervisor as groupKey2
ORDER BY groupKey1
RETURN groupKey1, groupKey2, COUNT(*) AS count;
```

Анализ содержимого для среднего количества слов/страниц/раскрытости.

Выполняется 1 запрос по Diploma.

```
MATCH (d:Diploma)
WITH d.year/d.academic_supervisor AS groupKey,
d.pages/d.words/d.minimal_disclosure AS metric
ORDER BY groupKey
RETURN groupKey, AVG(metric) AS avg;
```

Анализ содержимого для средней водности.

Выполняется 1 запрос по Diploma.

```
MATCH (d:Diploma)-[:CONTAINS]->(c:Chapter)
WITH d, SUM(c.water_content * c.words) * 1.0 / d.words AS water_content
WITH d.year/d.academic_supervisor as groupKey, water_content
ORDER BY groupKey
RETURN groupKey, AVG(water_content) AS avg;
```

2.2. Аналог модели для SQL СУБД

Графическое представление

Графическое представление модели см. на рис. 3.

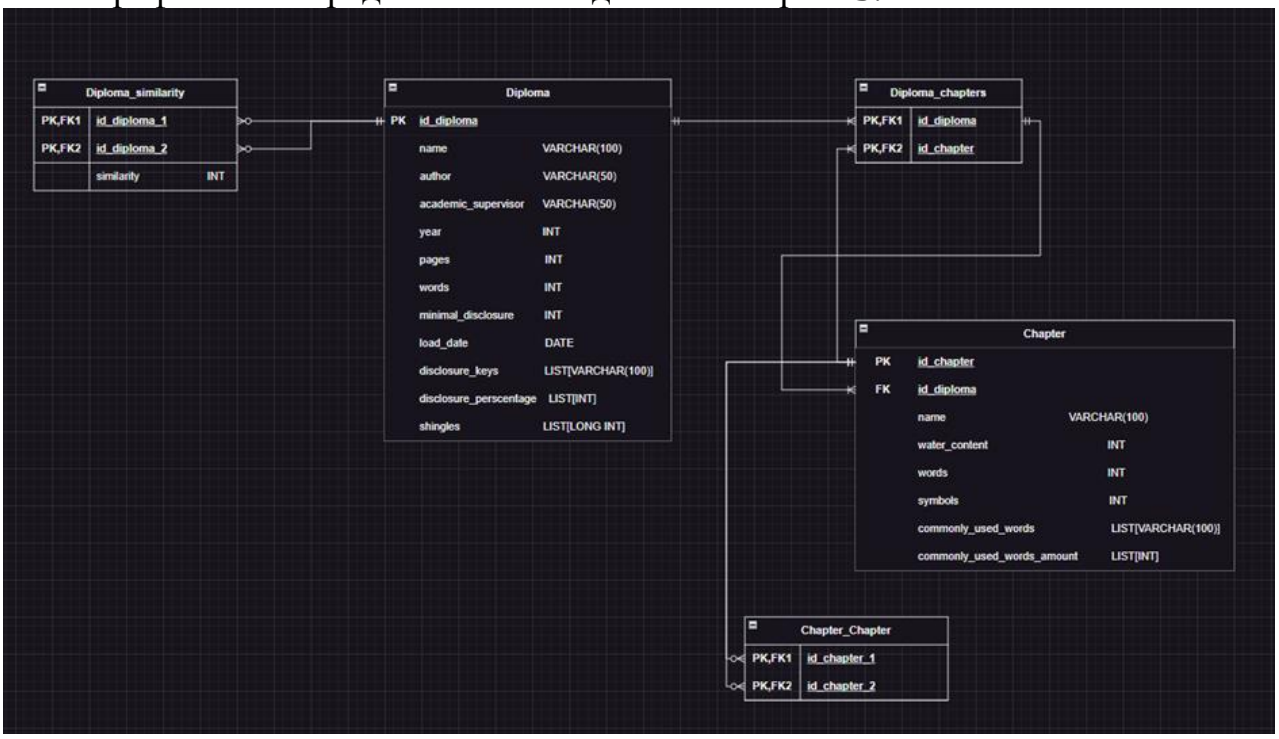

Рисунок 3 – Графическое представление реляционной модели

Описание назначений коллекций, типов данных и сущностей

Общее описание коллекций

Таблица Diploma

Основная коллекция, содержащая информацию о дипломной работе, включая метаданные и статистику.

Поля:

- **id** - Уникальный идентификатор диплома.
- **name** - Тема диплома.
- **author** - Автор работы.
- **academic_supervisor** - Научный руководитель.
- **year** - Год защиты.
- **pages** - Количество страниц.
- **words** - Общее количество слов.
- **minimal_disclosure** - Уровень минимального раскрытия тем, необходим для ускорения фильтрации.
- **load_date** - Дата загрузки в систему.
- **disclosure_keys** - Список раскрываемых в дипломе тем
- **disclosure_persentage** - Список процентных значений раскрытия по каждому элементу **disclosure_keys**.
- **shingles** - Упорядоченный список шинглов, необходимый для ускорения сравнения разных дипломов.

Таблица Chapter

Описывает свойства каждого раздела диплома, включая статистику текста, характеристики текста и ключевые элементы:

Поля:

- **id** - Уникальный идентификатор раздела.
- **id_diploma** - Внешний ключ на диплом, которому принадлежит раздел, нужен для ускорения извлечения данного поля без необходимости поиска корня дерева для каждого раздела (стоит учесть, что id родительского диплома неизменяем).
- **name** - Название раздела.

- **water_content** - Степень "водности" текста.
- **words** - Количество слов в разделе.
- **symbols** - Количество символов в разделе.
- **commonly_used_words** - Список наиболее часто встречающихся ключевых слов в тексте раздела.
- **commonly_used_words_amount** - Список количеств повторений для соответствующих слов из **commonly_used_words**.

Таблица Diploma_Similarity
Хранит степень схожести между парами дипломов.

Поля:

- **id_diploma_1** - Внешний ключ на диплом — первый участник пары для сравнения.
- **id_diploma_2** - Внешний ключ на диплом — второй участник пары.
- **similarity** - Целое число, отражающее процент схожести между двумя дипломами.

Таблица Diploma_Chapter
Связующая таблица между дипломами и внешними разделами (подобные связи используются для возможности отличия дочернего диплома от раздела, поскольку родителем раздела может являться как диплом, так и другой раздел, также подобная связь подходит и для поиска разделов от корня):

Поля:

- **id_diploma** - Внешний ключ на диплом.
- **id_chapter** - Внешний ключ на внешний раздел диплома.

Таблица Chapter_Chapter
Хранит информацию о вхождении одного раздела в другой.

Поля:

- **id_chapter_1** - Внешний ключ на материнский раздел.
- **id_chapter_2** - Внешний ключ на дочерний раздел.

Оценка объема сущностей и таблиц

Таблица Diploma (см. табл. 8)

Таблица 8 – Поля таблицы Diploma

Поле	Тип	Размер	Комментарий
id	serial	4	-
name	varchar(100)	100	в среднем 100 символов
author	varchar(30)	30	максимум 30 символов
academic_supervisor	varchar(30)	30	максимум 30 символов
year	integer	4	-
pages	integer	4	-
words	integer	4	-
minimal_disclosure	integer	4	-
load_date	date	4	-
disclosure_keys	varchar(100)[]	500	в среднем 5 тем по 100 байт
disclosure_persentage	integer[]	20	5 целых чисел по 4 байта
shingles	bigint[]	80000	в среднем 10000 слов

Итого: 80,704 байт

Таблица Chapter (см. табл. 9)

Таблица 9 – Поля таблицы Chapter

Поле	Тип	Размер	Комментарий
id	serial	4	-
id_diploma	serial	4	-
name	varchar(100)	100	-
water_content	integer	4	-
words	integer	4	-
symbols	integer	4	-
commonly_used_words	varchar(20)[]	100	5 20 байтовых слов
commonly_used_words_amount	integer[]	20	5 целых чисел по 4 байта

Итого: 240 байт

Таблица Diploma_Similarity (см. табл. 10)
Таблица 10 – Поля таблицы Diploma_Similarity

Поле	Тип	Размер
id_diploma_1	serial	4
id_diploma_2	serial	4
similarity	integer	4

Итого: 12 байт

Таблица Diploma_Chapter (см. табл. 11)
Таблица 11 – Поля таблицы Diploma_Chapter

Поле	Тип	Размер
id_diploma	serial	4
id_chapter	serial	4

Итого: 8 байт

Таблица Chapter_Chapter (см. табл. 12)
Таблица 12 – Поля таблицы Chapter_Chapter

Поле	Тип	Размер
id_chapter_1	serial	4
id_chapter_2	serial	4

Итого: 8 байт

Оценка объема информации, хранимой в модели

Формула общего объема памяти

Общий объем памяти базы данных оценивается по формуле: $V = V_D \times N_D + V_C \times N_C \times N_D + n_D \times v_D + n_{CC} \times v_{CC} + n_{DC} \times v_{DC}$

где:

- V_D - усредненное значение размера сущности Diploma (80704Б)
- N_D - количество сущностей Diploma
- V_C - усредненное значение размера сущности Chapter (240Б)
- N_C - среднее количество сущностей Chapter в дипломе
- v_D - размер сущности-связки Diploma_Similarity (12Б)

- n_D - количество Diploma_Similarity
- v_{CC} - размер сущности-связки Chapter_Chapter (8Б)
- n_{CC} - количество Chapter_Chapter
- v_{DC} - размер сущности-связки Diploma_Chapter (8Б)
- n_{DC} - количество Diploma_Chapter

Для вычисления числа связей между дипломами воспользуемся формулой вычисления числа ребер в полном графе:

$$n_D = N_D \times \frac{(N_D - 1)}{2}$$

Для оценки числа связей между Diploma и Chapter сделаем предположение, что среднее число разделов в одном дипломе равняется 3, тогда:

$$n_{DC} = 3 \times N_D$$

Число связей между Chapter и Chapter (вложенных заголовков дипломной работы) можно оценить, исходя из предположения, что средняя глубина вложенности заголовков в дипломе составляет 3 уровня. При этом: разделов первого уровня в среднем $C_0 = 3$, каждый раздел содержит в среднем $C_1 = 3$ подразделов (уровень 2), каждый подраздел (уровень 2) содержит в среднем $C_2 = 3$ подподразделов (уровень 3). Получаем:

$$N_C = (n_{DC} \times C_1 + n_{DC} \times C_1 \times C_2) \times N_D = (3 \times 3 + 3 \times 3 \times 3) \times N_D = 36 \times N_D$$

Тогда общее число сущностей Chapter для одного диплома составляет:

$$N_C = n_{DC} + n_{CC} = (3 + 36) \times N_D = 39 \times N_D$$

Пользуясь приведенными рассуждениями, получим итоговую формулу для вычисления необходимого объема памяти для хранения N_D дипломов в базе данных:

$$V = 80704 \times N_D + 240 \times 39 \times N_D + \frac{N_D(N_D - 1)}{2} \times 12 + 36 \times 8 \times N_D + 3 \times 8 \times N_D = 6 \times N_D^2 + 90370 \times N_D$$

Избыточность данных

Для оценки избыточности модели вычислим объем "чистых" данных, исключив дублирующую информацию:

В узловых разделах (не терминальных главах) (см. табл. 13)

Таблица 13 – Избыточные поля разделов

Поле	Тип	Размер	Описание	Комментарий
words	integer	4	Количество слов в разделе	Сумма полей **words** дочерних разделов
symbols	integer	4	Количество символов	Сумма полей **symbols** подразделов
water_content	integer	4	Водность текста	Средневзвешенное по подразделам

Таблица 14 - Избыточность в сущности Diploma

Поле	Тип	Размер
minimal_disclosure	integer	4 байт

Таблица 15 - Избыточность в сущности Chapter

Поле	Тип	Размер	Описание
id_diploma	integer	4 байт	Ссылка на родительский диплом

Формула избыточности:

$$\frac{V}{V_{\text{чистые}}} = \frac{V}{V - N_D \times \left[\left(n_{\text{words}} + n_{\text{sym}} + \text{water}\right) \times n_{\text{nodes}} + V_{\text{idDiploma}} \times N_C + \text{disclosure}\right]}$$

где:

Параметр	Описание	Значение
n_{words}	Размер поля "количество слов"	4 байт
n_{sym}	Размер поля "количество символов"	4 байт
water	Размер поля "водность"	4 байт
n_{nodes}	Среднее количество не листовых разделов	16
$V_{idDiploma}$	Размер поля id_diploma	4 байт
disclosure	Размер поля minimal_disclosure	4 байт
N_C	Количество разделов	$39 \times N_D$ байт

$$\frac{V}{V_{\text{чистые}}} = V/(V - N_D \times [(4+4+4) \times 16 + 4 \times 39 + 4]) = \frac{V}{V - N_D \times 352} = \frac{6 \times N_D + 90370}{6 \times N_D + 90018}$$

Направление роста модели

Получим формулу зависимости величины объема данных от количества объектов каждой сущности:

$$V = V_D \times N_D + V_C \times N_C = 80704 \times N_D + 240 \times N_C$$

где:

- V_D - усредненное значение размера сущности Diploma (80704 байта)
- N_D - количество сущностей Diploma
- V_C - усредненное значение размера сущности Chapter (240 байта)
- N_C - количество сущностей Chapter

Основной вклад в рост объёма памяти вносит увеличение числа дипломных работ, поскольку каждая из них имеет значительный размер (80704 байта). Дополнительным фактором является каскадный эффект: каждая новая работа сопровождается несколькими разделами (по 240 байт каждый). Таким образом, при добавлении одной дипломной работы общий объем данных возрастает не только за счёт её собственного размера, но и за счёт связанных с ней разделов.

Примеры запросов

Запросы пишутся на диалекте PostgreSQL, под **?** имеется в виду поступаемая информация извне, под **<name>** - полученная информация из БД. **N** - количество дипломов в БД.

Основной сценарий использования

Основной сценарий предполагает загрузку дерева документа в БД и сравнение с другими дипломами. В среднем выполняется **N + 95 запросов**, затрагивающих все таблицы.

1. Загрузка диплома

В данном запросе возвращается id созданного диплома для дальнейших действий с ним) (1 запрос)

```
INSERT INTO Diploma (
    name,
    author,
    academic_supervisor,
    year,
    pages,
    words,
    minimal_disclosure,
    load_date,
    disclosure_keys,
    disclosure_persentage,
    shingle
) VALUES (
    ?, ?, ?, ?, ?, ?, ?,
    current_date,
    '{?[]}',
    '{?[]}',
    '{?[]}'
)
RETURNING id;
```

2. Загрузка внешних разделов диплома (введение, содержание и т.д.)

Решается 2 запросами: вставка раздела и его связь с дипломом. Всего в среднем 3 внешних раздела, то есть 6 запросов.

Вставка раздела:

```
INSERT INTO Chapter (
    id_diploma,
    name,
    water_content,
    words,
    symbols,
    commonly_used_words,
    commonly_used_words_amount
) VALUES (
    :id_diploma, ?, ?, ?, ?,
    '{?[]}',
    '{?[]}'
)
RETURNING id;
```

В данном запросе возвращается id созданного раздела для дальнейших действий с ним.

Связывание с дипломом:

```
INSERT INTO Diploma_Chapter VALUES (
    :id_diploma,
    :id_chapter
);
```

3. Загрузка подразделов

Загрузка подразделов подобна загрузке внешних разделов за исключением связи с разделом. В среднем 36 подразделов, поэтому 72 запроса.

```
INSERT INTO Chapter_Chapter VALUES (
    :id_chapter_1,
    :id_chapter_2
);
```

4. Извлечение списка шинглов для сравнения(1 запрос)

```
SELECT id, shingles
FROM Diploma
WHERE id <> :id_diploma;
```

5. Занесение результатов сравнения (N запросов)

```
INSERT INTO Diploma_Similarity VALUES (
    :id_diploma,
    :id_diploma_cmp,
    ?
);
```

6. Извлечение диплома и его разделов (2 запроса)

```
SELECT * FROM Diploma WHERE id = :id_diploma;
SELECT * FROM Chapter WHERE id_diploma = :id_diploma;
```

7. Извлечение связи разделов

Данный запрос служит для правильной компоновки на странице (в среднем 13 запросов, учитывая количество родительских разделов)

```
SELECT * FROM Diploma_Chapter WHERE id_diploma = :id_diploma;
```

запрос на каждый родительский раздел:

```
SELECT * FROM Chapter_Chapter WHERE id_chapter_1 = :id_chapter;
```

Поиск по дипломам с фильтрацией и сортировкой.

Выполняется 1 запрос, задевая коллекцию Diploma.

```
SELECT *, array_tag(id_chapter) AS chapters FROM Diploma
JOIN Diploma_Chapter ON Diploma_Chapter.id_diploma = Diploma.id
WHERE id BETWEEN :min_id AND :max_id
AND '{:chapters[]}' && chapters
AND LOWER(:name) LIKE '%' || LOWER(name) || '%'
AND LOWER(:author) LIKE '%' || LOWER(author) || '%'
AND LOWER(:academic_supervisor) LIKE '%' || LOWER(academic_supervisor) ||
'%'
AND year BETWEEN :min_year AND :max_year
AND pages BETWEEN :min_pages AND :max_pages
AND words BETWEEN :min_words AND :max_words
AND minimal_disclosure BETWEEN :min_minimal_disclosure AND
:max_minimal_disclosure
AND load_date BETWEEN :min_date AND :max_date
ORDER BY id/year/pages/words/load_date
LIMIT K OFFSET M;
```

Поиск по разделам с фильтрацией и сортировкой.

Выполняется 1 запрос, задевающий Chapter и Chapter_Chapter.

```
SELECT *, array_tag(id_chapter_2) AS chapters FROM Chapter
JOIN Chapter_Chapter ON Chapter.id = Chapter_Chapter.id_chapter_2
WHERE id BETWEEN :min_id AND :max_id
AND id_diploma BETWEEN :min_id_diploma AND :max_id_diploma
AND '{:words[]}' && commonly_used_words
AND '{:chapters[]}' && chapters
AND LOWER(:name) LIKE '%' || LOWER(name) || '%'
AND water_content BETWEEN :min_water_content AND :max_water_content
AND words BETWEEN :min_words AND :max_words
AND symbols BETWEEN :min_symbols AND :max_symbols
ORDER BY id/id_diploma/water_content/words/symbols
LIMIT K OFFSET M;
```

Анализ содержимого для года и научного руководителя.

Выполняется 1 запрос по Diploma.

```
SELECT year/academic_supervisor, COUNT(*) FROM Diploma
GROUP BY year/academic_supervisor
```

Анализ содержимого для среднего количества слов/страниц/раскрытости.

Выполняется 1 запрос по Diploma.

```
SELECT AVG(pages/words/minimal_disclosure) FROM Diploma
GROUP BY year/academic_supervisor
```

2.3. Сравнение моделей

Удельный объем информации

Сравнение удельного объема информации см. в табл. 16

Таблица 16 – Сравнение удельного объема информации моделей

Параметр	NoSQL	SQL
Оценка объема	$12N^2 + 92280N$	$6N^2 + 90370N$
Избыточность для 10 сущностей	1,007	1,004

В целом показатели очень схожие, поскольку схемы хранения данных похожи (в силу немалой реляционности Neo4j), поэтому модели выделяются лишь объемом данных, необходимых для хранения объектов. PostgreSQL хранит данные компактно, в то время как Neo4j использует по умолчанию довольно большие типы данных для тех же чисел и связей, что на самом деле лишь частично описывает их объем, ведь на самом деле связь без доп. полей весит 34Б,

что еще сильнее увеличивает объем, но за это мы платим удобством и скоростью запросов. Также стоит учесть, что больший объем данных занимают списки шинглов, необходимые для оптимального сравнения дипломов.

Запросы по отдельным юзкейсам

В целом можно сказать, что запросы между двумя моделями очень схожи, что неудивительно, ведь Neo4j имеет очень схожий язык запросов и структуру в целом. Но при этом есть места, где связанность данной СУБД позволяет значительно сократить число запросов. Например, она позволяет получить всё дерево диплома по его корню всего за 1 запрос, в отличие от SQL, для которого нужно вручную писать своего рода BFS, чтобы шаг за шагом получить все узлы дерева. Также при использовании Neo4j сократилось количество запросов в местах связи узлов, ведь он позволяет создавать новые связи в рамках одного запроса. Благодаря данным упрощениям для основного сценария использования получилось уменьшить кол-во запросов в среднем с **N + 95** до **N + 46**. То есть если оптимизировать подход и не добавлять связи для каждого диплома (например, для 5 самых похожих), то количество запросов уменьшилось в **2** раза. По остальным сценариям нет больших отличий, поскольку запросы чтения очень схожи между собой (за исключением извлечения диплома с его разделами).

В целом можно сказать, что в силу графового подхода Neo4j позволяет писать меньше запросов, причем без JOIN, также задевая меньше сущностей благодаря нереляционному подходу к организации связей. Но при этом стоит отметить, что некоторые операции легче описать в SQL.

Вывод

В целом обе модели схожи в силу устройства графовых СУБД как более удобного способа описывать связи. При этом Neo4j выделяется меньшим числом запросов, необходимых для выполнения сценариев использования, но большим объемом данных. Учитывая современные реалии и относительно небольшой разрыв, можно опустить данную разницу и выбрать графовую БД как способ облегчения работы с древовидными структурами данных.

3. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

3.1. Краткое описание

Реализованное приложение представляет собою монолит, формально делящийся на несколько компонент: API приложения на Flask (предоставляет доступ ко всем страницам приложения, авторизацию, а также операции добавления/извлечения файлов), представления страниц на шаблонизаторе Jinja2, репозитории на py2neo (необходимы для связи с БД и исполнения запросов), парсеры дипломов, использующие NLTK, py2morphy.

Поскольку как такового фронтенд-слоя нет, всё приложение имеет одну точку входа, поэтому используется два контейнера: app (основной контейнер, предоставляющий доступ к сервису) и db (контейнер Neo4j).

3.2. Использованные технологии

- Backend: Python, Flask
- Frontend: Jinja2, HTML, CSS, JS
- БД: Neo4j с драйвером py2neo
- Морфологические анализаторы: NLTK, py2morphy
- Docker

3.3. Снимки экрана приложения

Снимки экранов представлены на рис. 4 – 11.

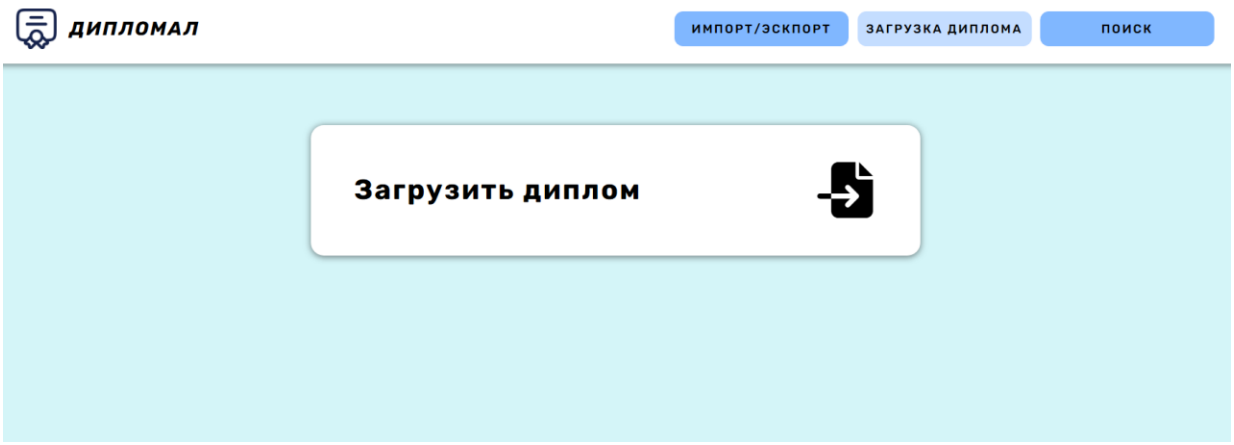

Рисунок 4 – Экран загрузки диплома

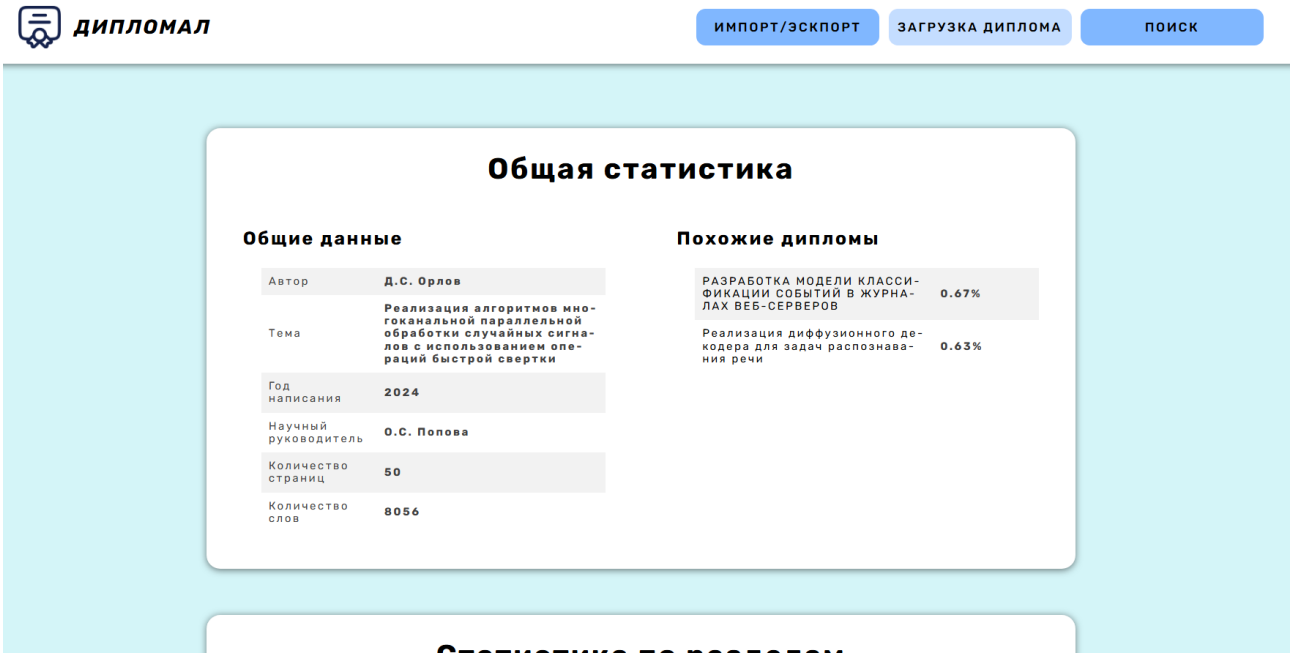

Рисунок 5 – Общая статистика диплома

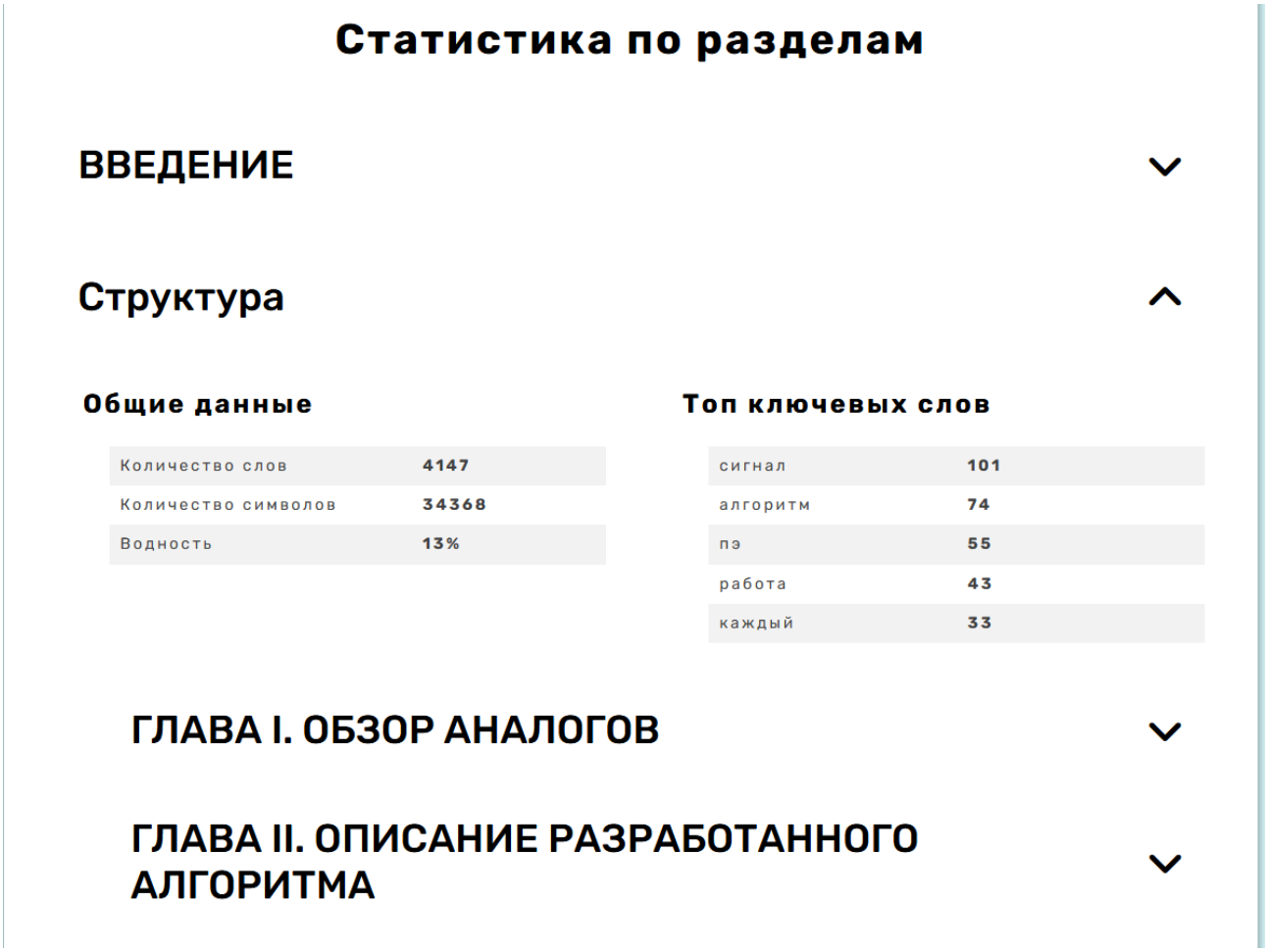

Рисунок 6 – Статистика по разделам

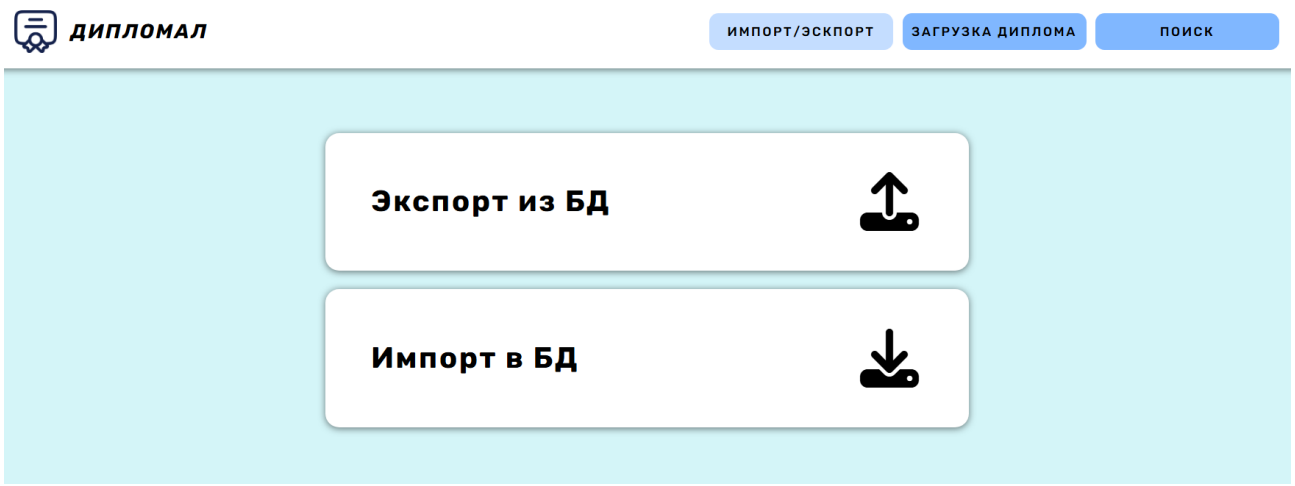

Рисунок 7 – Страница Импорт/Экспорта

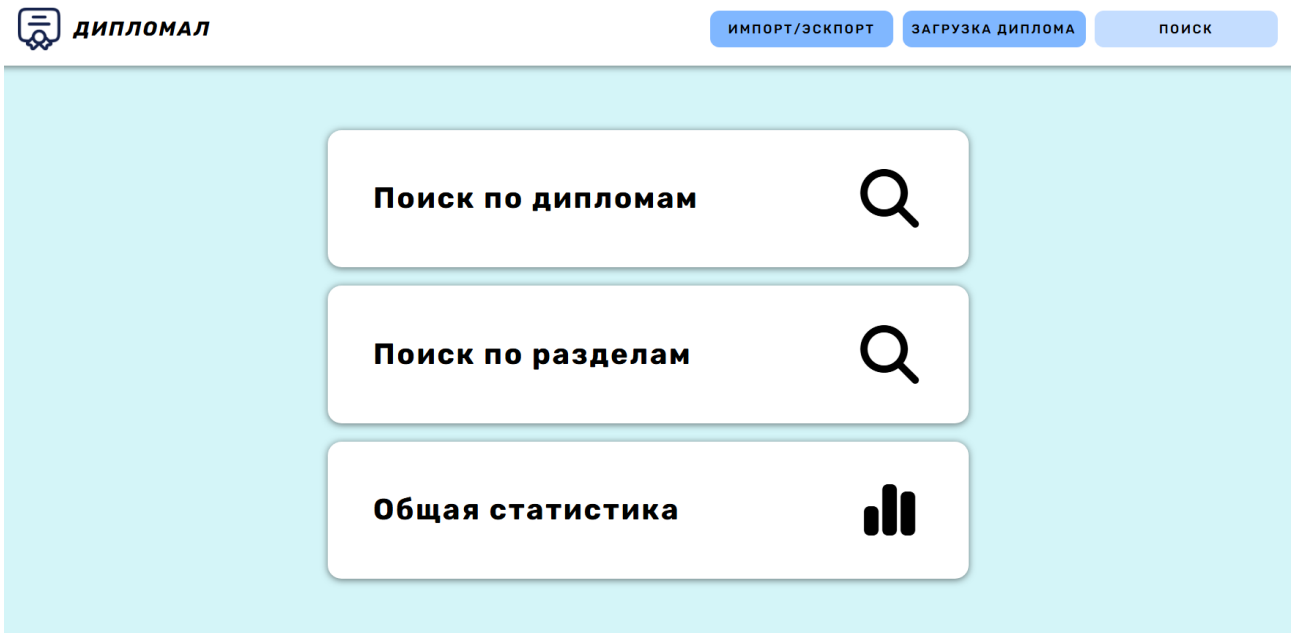

Рисунок 8 – Страница поиска

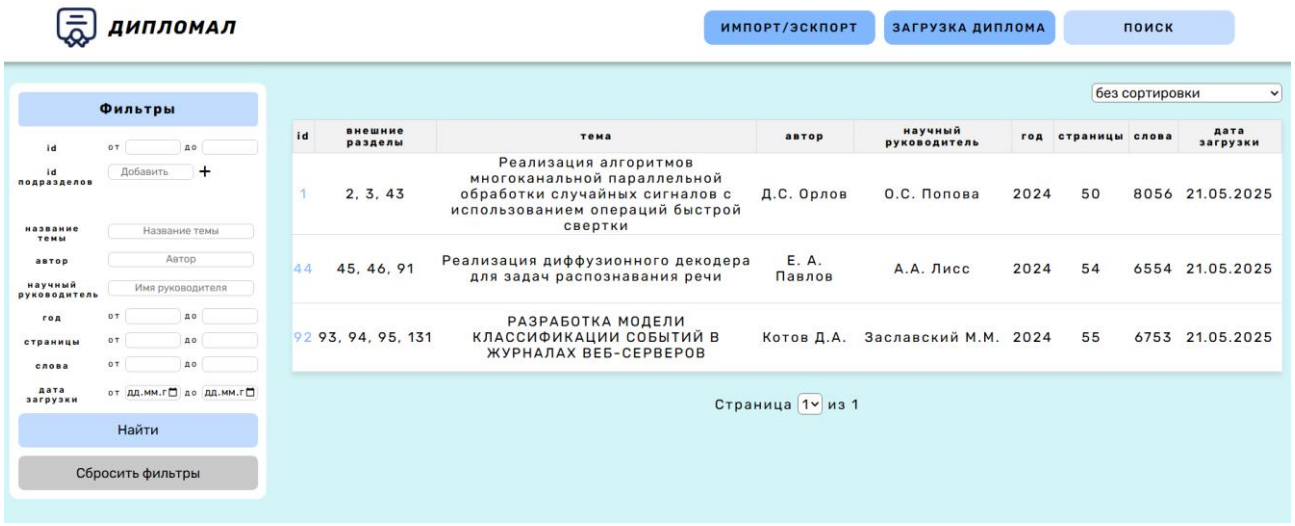

Рисунок 9 – Страница поиска дипломов

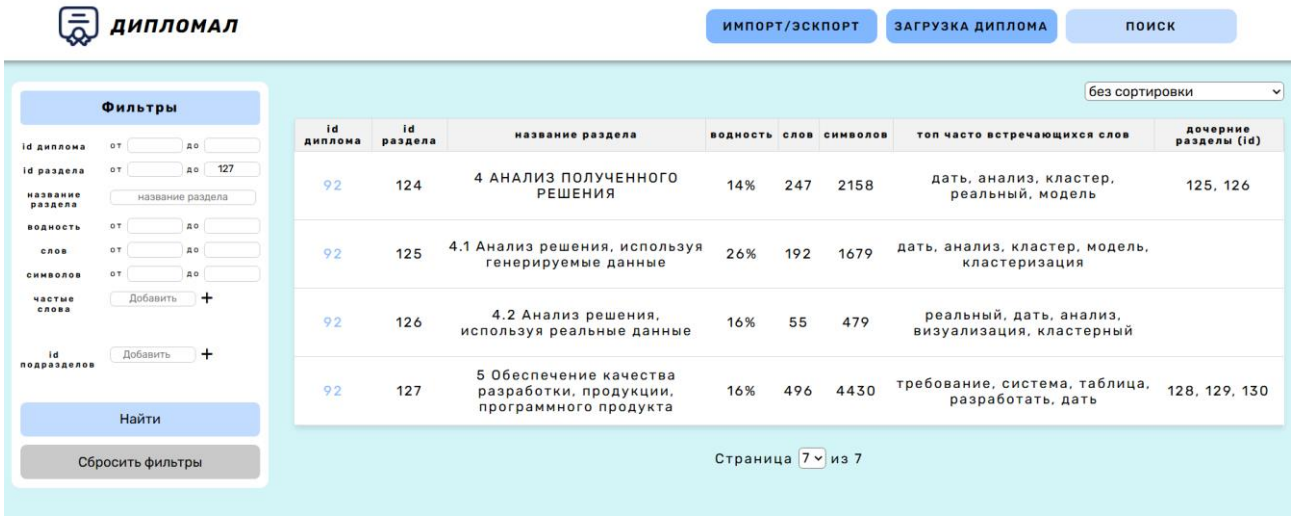

Рисунок 10 – Страница поиска разделов

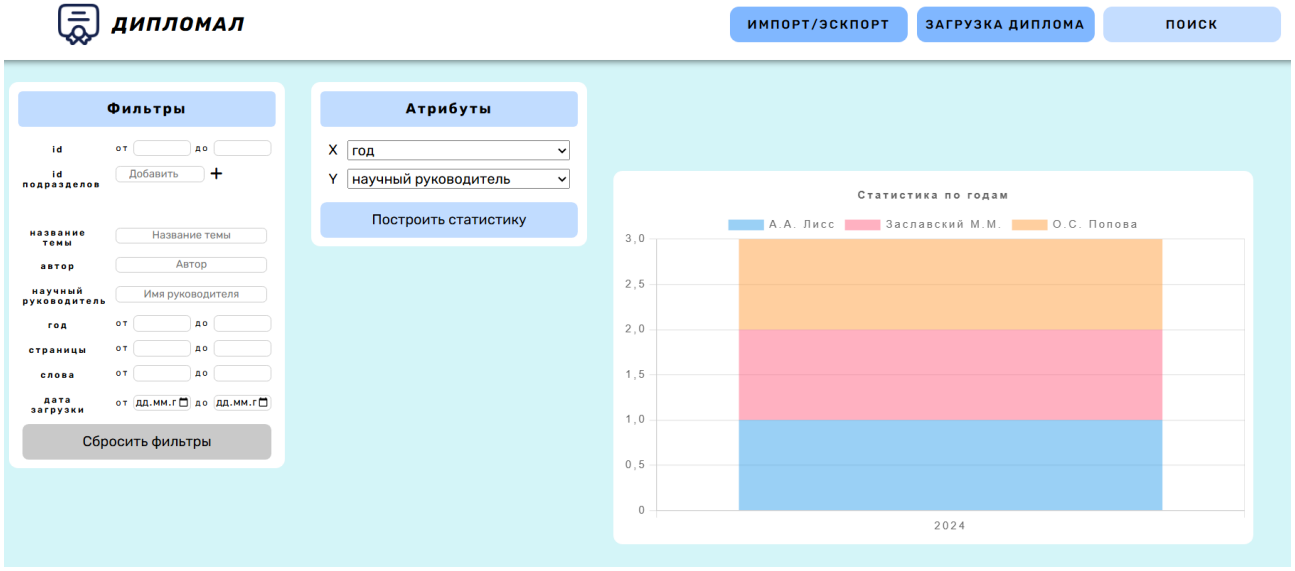

Рисунок 11 – Страница кастомизируемой статистики

ЗАКЛЮЧЕНИЕ

Достигнутые результаты

В итоге реализовано приложение, позволяющее анализировать дипломы, предоставляя основные сведения о дипломе и его разделах. Также есть возможность поиска по существующей базе, импорт/экспорт всей БД и построения кастомизируемой статистики. Данные хранятся в графовой БД Neo4j, само приложение реализовано на Python в связке с Flask. Приложение докеризовано, что позволяет более гибко распространять его.

Недостатки и пути для улучшения полученного решения

Основным недостатком решения является достаточно грубое вычисление различных метрик, для улучшения которых можно улучшить алгоритмы морфологического анализа и применить языковые модели.

Будущее развитие решения

Дальнейшее развитие решение предполагает увеличение количества и качества метрик дипломов, увеличение производительности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Репозиторий проекта // GitHub. URL: https://github.com/moevm/nosql1h25-reports.

2. Документация Neo4j // Neo4j Documentation. URL: https://neo4j.com/docs/.

3. Документация Flask // Flask Documentation. URL: https://flask.palletsprojects.com/en/stable/.

ПРИЛОЖЕНИЕ А

ДОКУМЕНТАЦИЯ ПО СБОРКЕ И РАЗВЕРТЫВАНИЮ ПРИЛОЖЕНИЯ

1. Склонировать репозиторий

```
git clone https://github.com/moevm/nosql1h25-reports.git
```

2. Запустить приложение

```
docker compose build --no-cache && docker compose up -d
```

3. Приложение будет доступно по адресу *localhost:5000*