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

ИНДИВИДУАЛЬНОЕ ДОМАШНЕЕ ЗАДАНИЕ

по дисциплине «Разработка ПО информационных систем»

Тема: Анализ сложности книг

Студенты гр. 5381 _____ Розенкинд Е.А.

_____ Гаськов М.В.

_____ Кобылянский А.В.

Преподаватель _____ Заславский М.М.

Санкт-Петербург

2018

ЗАДАНИЕ

Студенты
Розенкинд Е.А.
Гаськов М.В.
Кобылянский А.В.
Группа 5381

Тема проекта: Разработка приложения для анализа сложности книг.

Исходные данные:
Необходимо реализовать приложение для анализа сложности книг с возможностью вывода статистики по каждой из книг, содержащейся в базе данных (MongoDB).

Содержание пояснительной записки:
«Содержание»
«Введение»
«Качественные требования к решению»
«Сценарий использования»
«Модель данных»
«Разработанное приложения»
«Заключения»
«Приложения»
«Список используемых источников»

Предполагаемый объем пояснительной записки:
Не менее 10 страниц.

Дата выдачи задания:

Дата сдачи реферата:

Дата защиты реферата:

Студенты гр. 5381	_____	Розенкинд Е.А.
	_____	Гаськов М.В.
	_____	Кобылянский А.В.
Преподаватель	_____	Заславский М.М.

АННОТАЦИЯ

В рамках данного курса предполагалась разработать проект в команде с использование одной из БД NoSQL. В качестве проекта было выбрано приложение для анализа книг с использованием БД MongoDB. Перед началом разработки, были написаны запросы для стемминга книги. Само приложение представляет собой веб-сайт, т.к. в браузере проще реализовать кроссплатформенность. В качестве языка разработки серверной части выбран Kotlin, а в качестве библиотеки для работы с MongoDB – Kmongo.

SUMMARY

As part of this course, it was proposed to develop a project in a team using one of the NoSQL databases. As a project, an application for analyzing books using MongoDB DB was chosen. Before the start of development, requests were written to stem the book. The application itself is a website, because in the browser, it is easier to implement cross-platform. Kotlin was chosen as the server development language, and Kmongo was chosen as the library for working with MongoDB.

СОДЕРЖАНИЕ

1. ВВЕДЕНИЕ

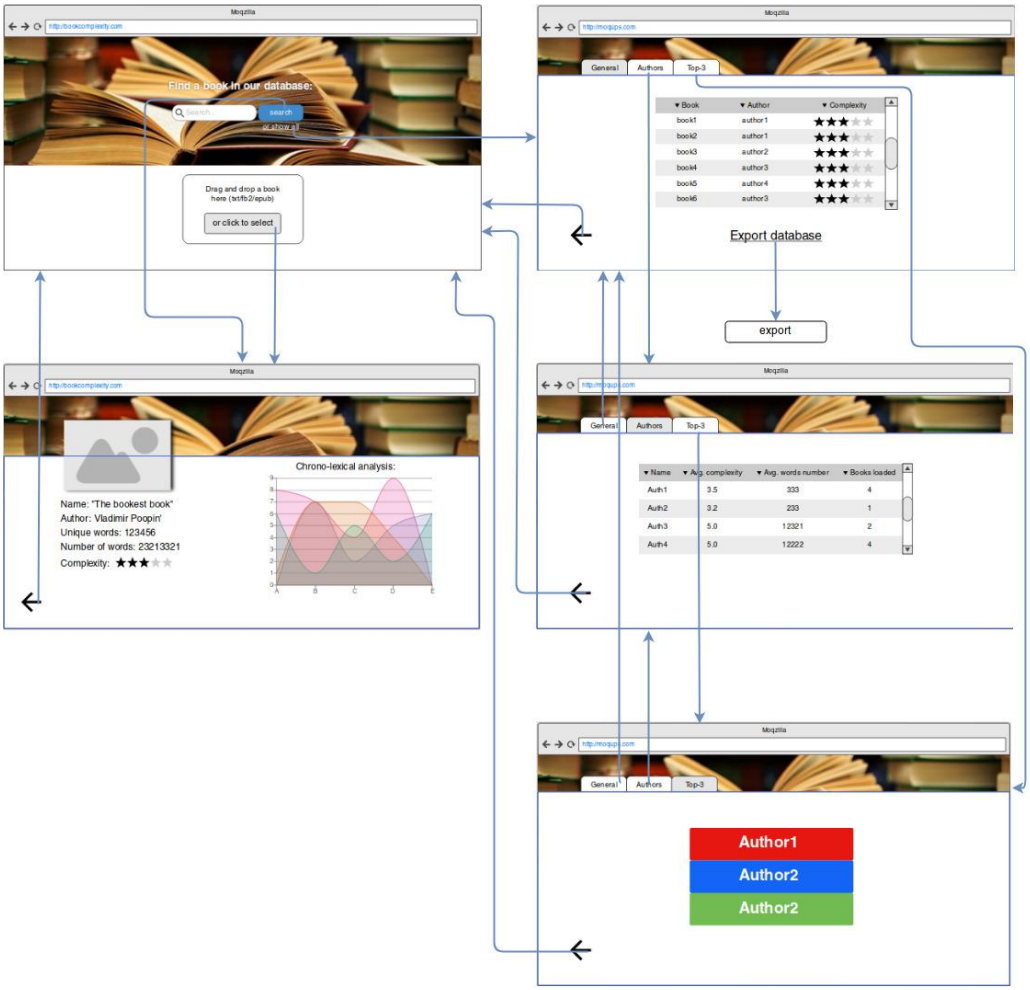
Цель работы – создать приложение, анализирующее сложность книг и выдающее статистику по каждой из книг, содержащихся в БД.

2. КАЧЕСТВЕННЫЕ ТРЕБОВАНИЯ К РЕШЕНИЮ

Требуется разработать приложение для анализа сложности книг, весь процесс стемминга которого реализован только с использованием запросов БД MongoDB.

3. СЦЕНАРИЙ ИСПОЛЬЗОВАНИЯ

Макет UI



Описание сценариев использования

1. Use Case #1: «Поиск книги в БД»

- Пользователь находится на начальной странице.
- Пользователь вводит в поле автора / название книги.
- Пользователь нажимает на кнопку «search».

- Сервис выдает статистику по книге на следующей странице.
  - Или информацию о том, что такой книги нет в базе.
- У пользователя есть возможность вернуться к начальной странице.

2. Use Case #2: «Загрузка новой книги в БД»

- Пользователь находится на начальной странице.
- Пользователь переносит файл в нужном формате (txt/epub/fb2) в область загрузки.
  - Или нажимает кнопку «or click to select» для загрузки через менеджер файлов.
- Сервис выдает статистику по книге на следующей странице:
  - Общее количество слов.
  - Количество уникальных слов.
  - Графики Google ngram.
- У пользователя есть возможность вернуться к начальной странице.

3. Use Case #3: «Отображение всей базы книг»

- Пользователь находится на главной странице.
- Пользователь нажимает на кнопку «or show all» под полем поиска.
- Сервис отображает страницу с тремя вкладками:
  - «General» - отображение таблицы рейтинга книг.

- «Authors» - отображение таблицы рейтинга авторов.
- «Top 3» - отображение трех авторов из топа.

У пользователя есть возможности:

- Сменить вкладку на одну из трех.
- Вернуться к предыдущей странице по кнопке.
- Нажать на кнопку экспорта БД.

При выборе последнего, на компьютер пользователя загружается база данных книг.

4. МОДЕЛЬ ДАННЫХ

Нереляционная модель данных (MongoDB)

У нас есть 2 основные коллекции - `books_stats` и `words_stats`.

В коллекции `books_stats` хранятся документы со статистикой по книгам.

Схема документа из этой коллекции:

```
{
    _id: ObjectId,
    cover: ObjectId,
    title: String,
    author: String,
        published: Date,

    words_count: int,
    unique_words_count: int,
    unique_stems_count: int,

    lexicon_years: byte[60],
    lexicon_rarity: double,
```

```
        difficulty: double,
    }
```

В коллекции `words_stats` хранится статистика для отдельных слов. Эта информация понадобится в процессе обработки книги для вычисления `lexicon_years` и `lexicon_rarity`. Схема документа из этой коллекции:

```
{
    _id: String,
    years: double[60]
}
```

Описание назначений коллекций, типов данных и сущностей

books_stats

Основная коллекция, в которой хранятся документы со статистикой по всем книгам из бд.

- _id: ObjectId - идентификатор книги
- cover: ObjectId - идентификатор обложки, хранимой в GridFs
- title: String - название книги
- author: String - автор книги

- published: Date - дата публикации книги
- words_count: int64 - количество слов в книге
- unique_words_count: int64 - количество уникальных слов
- unique_stems_count: int64 - количество уникальных словоформ
- lexicon_years: byte[60] - нормированная гистограмма к каким годам относится лексика в книге
- lexicon_rarity: double - индекс редкости слов, используемых в книге
- difficulty: double - сложность книги

words_stats

Вспомогательная коллекция. Хранит статистику для отдельных слов, которая нужна для вычисления lexicon_years и lexicon_rarity из основной коллекции.

- _id: String - слово
- years: double[60] - популярность слова в определенные года

books_stats

Байт на один документ из `books_stats`:

`2x12 + ~20 + ~15 + 8 + 3x8 + 60x1 + 2x8 + 112= 289 b`

Плюс у каждой книги может быть обложка, лежащая в GridFS размером примерно в `100 kb`.

Всего в источнике, откуда мы будем брать книги, - Project Gutenberg примерно 57 000 книг. Тогда получаем:

*57000*x(289 + 100x1024) = 5.71 Gb

В общем виде формула примет следующий вид:

n x (k + p x 1024), где n – количество книг, k – байт на один документ, p - размер картинки книги в kb.

words_stats

Байт на один документ из words_stats:

~4.96 + 60x8 + 8 = 492.96 b

где 4.96 - средняя длина слова в английском языке.

В Oxford English Dictionary за 1989 год было 218632 слов. Тогда получаем:

218632x484.96 = 101.11 Mb

будет занимать коллекция words_stats

В общем виде формула примет следующий вид:

n x c, где n – количество слов в базе данных, c – константа = 492.96 b

Индексы

- Текстовый индекс для books_stats по полям author и title для поиска книги в базе. Размер индекса линейно зависит от количества уникальных слов в этих полях.

- Индекс для `words_stats` по полю `_id` для ускорения обработки книги. Размер индекса линейно зависит от количества документов в `words_stats`.

Запросы к модели, с помощью которых реализуются сценарии использования

Use Case #1 - поиск книги в бд

```
db.books_stats.find({ $text: { $search: <текст запроса> } })
```

Полученный список книг используется для составления страницы с результатами поиска. При выборе конкретной книги получаем информацию о ней по `_id`:

```
db.books_stats.find({ _id: <id> })
```

И достаем обложку из GridFS:

```
mongofiles -d fridFS --local <имя файла в локальной фс> get <имя файла в GridFS>
```

Три запроса к бд.

Use Case #2 - добавление информации о новой книге в бд

Загружаем обложку в GridFS:

```
mongofiles -d fridFS --local <имя файла в локальной фс> put <имя файла в GridFS>
```

При загрузке книги происходит обработка, ссылку на запросы которой можно найти на вики проекта.

10 запросов для обработки книги.

Use Case #3 - статистика по книгам в бд

Топ 10 самых сложных книг

```
db.books_stats.find().sort({difficulty: -1}).limit(10)
```

Топ 10 самых сложных авторов

```
db.books_stats.aggregate(
[
    {
        $group: {
            _id: "$author",
            difficulty: { $avg: "$difficulty"}
        }
    },
    { $sort: { difficulty: -1 } },
    { $limit: 10 },
    { $project: { author: "$_id", difficulty: 1, _id: 0,  }}
])
```

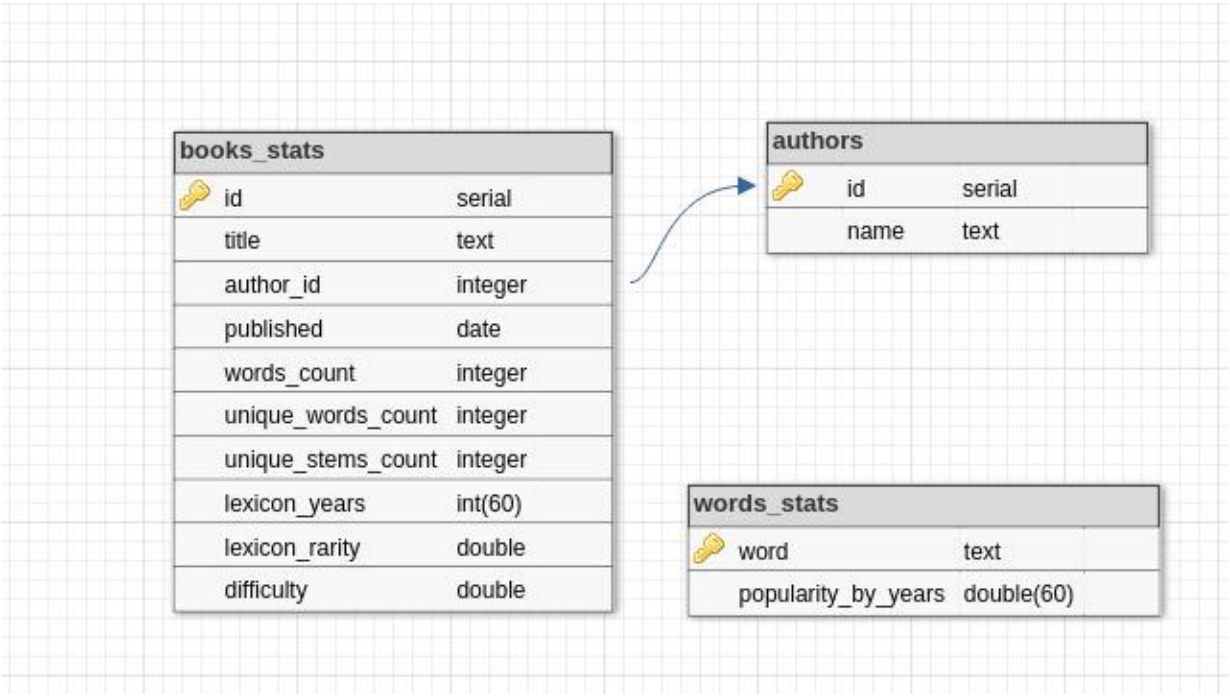
Средняя сложность книг по годам

```
db.books_stats.aggregate(
[
    {
        $group: {
            _id: {$multiply: [10, {$floor: {$divide: ["$published",
 10]}}]},
            difficulty: { $avg: "$difficulty"}
        }
    },
    { $sort: { _id: 1 } }
])
```

Три запроса.

РЕЛЯЦИОННАЯ МОДЕЛЬ ДАННЫХ

Графическое представление модели данных



Оценка удельного объема информации, хранимой в модели

Объем данных будет примерно такой же, как в нереляционной модели.

Запросы к модели, с помощью которых реализуются сценарии использования

Use Case #1 - поиск книги в бд

```
SELECT books_stats.title, books_stats.published, authors.name
FROM books_stats
INNER JOIN authors ON books_stats.author_id = authors.id
WHERE concat(authors.name, ' ', books_stats.title) = <текст запрос
a>;
```

Достаем книгу по id

```
SELECT books_stats.*, authors.name
FROM books_stats
INNER JOIN authors ON books_stats.author_id = authors.id
WHERE books_stats.id = <id>;
```

Два запроса к бд.

Use Case #2 - добавление информации о новой книге в бд

При загрузке книги происходит обработка, которую можно найти на вики проекта.

В SQL нет aggregation pipline и map-reduce, так что обработка бы выполнялась внутри основной программы. Для выполнения обработки понадобился бы один SELECT, чтобы достать из `words_stats` данные по словам в обрабатываемой книге. Один запрос к бд.

Use Case #3 - статистика по книгам в бд

Топ 10 самых сложных книг

```
SELECT books_stats.id, books_stats.title, books_stats.difficulty,
authors.name
FROM books_stats
INNER JOIN authors ON books_stats.author_id = authors.id
ORDER BY books_stats.difficulty DESC
LIMIT 10;
```

Топ 10 самых сложных авторов

```
SELECT top.avg_difficulty, authors.name
FROM (
    SELECT AVG(difficulty) as avg_difficulty, author_id
    FROM books_stats
    GROUP BY author_id
    ORDER BY avg_difficulty DESC
    LIMIT 10
```

```
) as top
INNER JOIN authors ON top.author_id = authors.id
```

Средняя сложность книг по годам

```
SELECT date_part('year', published) as year,  AVG(difficulty) as a
vg_difficulty
FROM books_stats
GROUP BY year
ORDER BY year ASC;
```

3 запроса к бд.

Оценка объема данных в реляционной модели

books_stats

Байт на один документ из `books_stats`:

```
2x12 + ~20 + ~15 + 8 + 3x8 + 60x1 + 2x8 = 167 b
```

Плюс у каждой книги может быть обложка, лежащая на диске размером примерно в `100 kb`.

Всего в источнике, откуда мы будем брать книги, - Project Gutenberg примерно `57 000` книг. Тогда получаем:

```
57000x(167 + 100x1024) = 5.44 Gb
```

В общем виде формула примет следующий вид:

n x (k + p x 1024), где n – количество книг, k – байт на один документ, p - размер картинки книги в kb.

words_stats

Байт на один документ из `words_stats`:

```
~4.96 + 60x8 = 484.96 b
```

где `4.96` - средняя длина слова в английском языке.

В Oxford English Dictionary за 1989 год было 218632 слов. Тогда получаем:

218632x484.96 = 101.11 Mb

будет занимать коллекция words_stats

В общем виде формула примет следующий вид:

n x c, где n – количество слов в базе данных, c – константа = 484.96 b

Сравнение моделей

В нереляционной модели объем данных оказался незначительно больше, чем в реляционной (5.44 Gb – в реляционной, 5.71Gb – в нереляционной, т.е. разница объема данных составила ~5%). Объем данных в нереляционной базе меньше, потому что не надо хранить названия полей в текстовом виде и из-за нормализации поля author.

5. РАЗРАБОТАННОЕ ПРИЛОЖЕНИЕ

Краткое описание архитектуры

Back-end представляет из себя Kotlin-приложение. Для работы с MongoDB использовался фреймворк KMongo. Все данные с клиентской части передаются через post, get методы на сервер, где соответственно обрабатываются. Некоторые из запросов KMongo не поддерживает. В этом случае использовался процессор запуска команд MongoDB.

Front-end – это веб приложение, верстка которого совпадает с ранее представленным макетом UI.

Использованные технологии

БД: MongoDB.

Back-end: Kotlin, KMongo, Ktor.

Front-end: HTML, CSS, JS.

Ссылка на приложение

1. https://github.com/moevm/nosql2018-book_complexity.

6. ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было разработано приложение для анализа сложности книг и предоставления статистики по книгам пользователям. Приложение так же предоставляет возможность экспорта и импорта базы данных.

7. ПРИЛОЖЕНИЯ

1. Скачать проект по ссылке, указанной ранее.
2. Запустить проект в IDEA.
3. Открыть приложение в браузере на локалхосте.

8. СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Документация MongoDB: https://docs.mongodb.com/manual/
2. Ktor: https://ktor.io/
3. KMongo: https://litote.org/kmongo/