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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

НА ТЕМУ:

Разработка базы данных для хранения и аналитики результатов статистического опроса

Студент _____ _____ Т. А. Казаева
(Группа) (Подпись, дата) (И.О. Фамилия)

Руководитель курсовой работы _____ Ю. В. Строганов
(Подпись, дата) (И.О. Фамилия)

Москва, 2022 г.

Содержание

Страница

ВВЕДЕНИЕ

В настоящее время социальные сети и форумы становятся частью повседневной жизни подавляющего количества пользователей сети Интернет. Активный обмен мнениями в сети привел к увеличению интереса к автоматическому извлечению тональности текста как со стороны научного сообщества, так и со стороны многих коммерческих организаций.

Например, крупнейшие интернет-магазины, получают многочисленное количество отзывов на свои товары. Выделение общих закономерностей в большом количестве неструктурированных текстов — это рутинная работа, требующая автоматизации. Таким образом, подбор эффективного алгоритма извлечения тональных оценок является актуальной задачей экономической и производственной аналитики.

Часто вся выводная часть анализа адекватности тональной разметки текста строится на сопоставлении частот, полученных по всему корпусу материалов, не учитывая региональные особенности. Получить статистические представления об оценке тональных прилагательных можно с помощью анкетирования – метода проведения опроса, при котором для сбора сведений используется оформленный список вопросов.

Цель работы – спроектировать и реализовать программное обеспечение для проведения анкетированного опроса экспертов о тональности прилагательных. Требуется учесть особенности модели данных, полученных при анкетировании при выборе базы данных.

- проанализировать варианты хранения данных и выбрать подходящий вариант;
- спроектировать базу данных, описать ее сущности и связи;
- разработать систему разметки интерфейса;
- провести нагрузочное тестирование спроектированного приложения с учетом особенностей выбранной СУБД.

1. Аналитический раздел

В разделе описан способ выражения эмоций в тексте и подходы к его анализу. Приведены существующие размеченные корпуса для русского языка и описаны шкалы оценки тональности. Описано существующее решение для англоязычной лексики и рассмотрены некоторые модели данных, применяемые в no-SQL системах.

1.1 Подходы к анализу эмоций в тексте

Лексическая тональность выражается в тексте на уровне лексем или коммуникационных фрагментов. Коммуникативные фрагменты – это отрезки речи различной длины, которые хранятся в памяти говорящего в качестве стационарных частиц его языкового опыта и которыми он оперирует при создании и интерпретации высказываний. [1] Эмотивный окрас текста в целом определяется лексической тональностью единиц, составляющих текст и правилами их сочетания.

Традиционный подход к определению тональной оценки текста чаще всего заключается в использовании одномерного эмотивного пространства: позитив-негатив, то есть хорошо-плохо. Тональность текста определяется тремя факторами: субъектом и объектом тональности, а также тональной оценкой. [2]. Под субъектом понимается автор высказывания, под объектом – то, о чем высказывается субъект и под тональной оценкой – эмоциональное отношение автора к такому объекту.

Коммуникативные фрагменты могут содержать в себе явно и неявно эмоционально окрашенную лексику. Часто для получения оценки того или иного слова обращаются к словарям оценочной лексики. Слово может менять свою полярность или терять ее вовсе в зависимости от контекста. Поэтому для конкретных предметных областей существуют собственные тональные словари. [3]

Однако, с трактовкой явной оценочной лексики могут возникать сложности. Общие словари не могут рассматриваться как полноценная альтернатива классическим подходам оценки мнений на основе массовых

опросов [4]. Для получения более точных результатов автоматического извлечения тональности требуются заранее подготовленные данные – размеченный словарь эмоций или размеченный корпус. Разметка должна предполагать не только обобщение данных до боле крупных групп населения (например, согласно типологии [5]), но и ассоциировние с социально-демографическими группами, учет пола и возраста.

1.1.1 Зависимость уровня энтузиазма от географического положения

Лексический фон вбирает в себя те ассоциативные сведения, которые накапливаются у носителей языка в процессе применения слова. [6] Соответственно, аппроксимация оценки экспрессивного слова неразрывно связана с регионом, в котором проживает субъект тональности. Утверждается, [7] что понятие «счастье» используется наряду с понятием «субъективное благополучие». Его оценку можно получить, опираясь на активность пользователей в региональных группах ВКонтакте. Для оценки предложен ряд критериев.

1. Географическая репрезентативность. Регионы представлены равномерно на всей территории РФ — все федеральные округа, все климатические и культуро-исторические зоны.

2. Социально-экономическая репрезентативность. Согласно типологии [5], населенные пункты были поделены по уровню социально-экономического развития на четыре типа, представленных ниже.

    a) Крупные города – Москва и города-миллионники, города с населением свыше 500 тыс. человек.

    b) Средние по размеру индустриальные города с населением от 20 до 25 – 300 тыс. человек.

    c) «Периферия» – деревни, сёла и небольшие города.

    d) Республики Северного Кавказа и Юга Сибири.

Результаты исследования показали существенную разницу между регионами России. Например, индекс социального (не)благополучия в Республике Алтай составил $-0,4632$, а в Алтайском крае $-18,2462$. Такая существенная разница, скорее всего, будет влиять на общий уровень энтузиазма, и, в результате, на измерение оценки тональных прилагательных. При подготовке обучающего набора данных следует уделить особое внимание разметке корпусов в соответствии с личными характеристиками субъекта – пол, возраст, географическое положение и т. п.

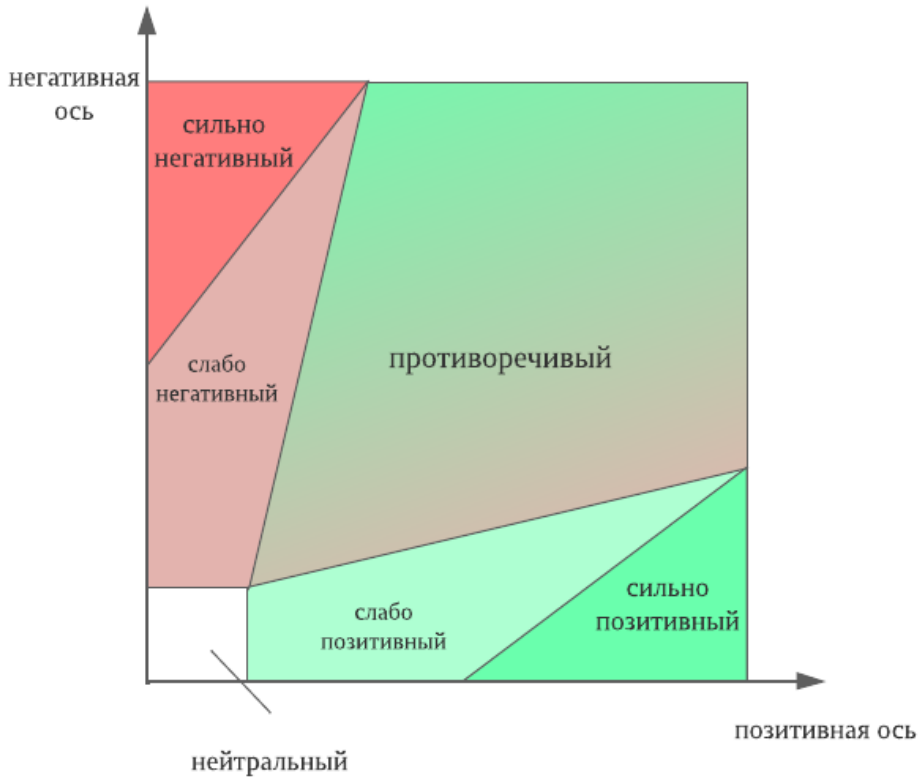
1.1.2 Шкалы оценки тональных слов

В большинстве случаев для оценки тонального слова используется одномерная шкала (1.1(а)). На такой шкале расположены шесть классов оценки: «сильно позитивный», «слабо позитивный», «сильно негативный», «слабо негативный», «нейтральный» (отсутствие выраженного эмотивного окраса) и «противоречивый» (в равной мере присутствует негативный и позитивный эмотивный окрас). Однако, в таком случае возникает неопределенность в отношении нулевого, «центрального» значения – ноль на плоской шкале может обозначать как отсутствие эмотивного окраса, так и присутствие позитивного и негативного окраса в равной мере.

Для разрешения этой неопределенности в качестве шаклы оценки используется двумерное пространство, представленное на рис. 1.1(б). Границы между классами изображены приблизительно, симметрия предположительна.

Минимальное количество классов в корпусе – два (позитивный и негативный). [8] Встречаются также случаи пятибальной шкалы оценивания, где «3» – это центральное значение и десятибальной шкалы, где «5» – это центральное значение.

(а)



(б)

Рисунок 1.1 – Эмотивное пространство – плоское(а), объемное(б)

1.1.3 Существующие размеченные эмоциональные корпуса

Корпус ROMIP-2012. [9] Чтобы составить коллекцию, один эксперт разметил отзывы на книги, фильмы и цифорвые камеры. Для проверки ответов систем тестовая коллекция поступила на оценку одному эксперту. В его задачу входило отобрать из всех имеющихся постов такие, которые релевантны заданным предметным областям, содержат оценку упоминаемых объектов, а также классифицировать отобранные посты по трем шкалам: двухбалльной (позитивный, негативный), трехбалльной (позитивный, негативный, удовлетворительный), пятибалльной (отлично, хорошо, средне, плохо, ужасно).

Корпус SentiRuEval-2015. [10] В 2015 году было проведено тестирование автоматического анализа тональности русскоязычных текстов. Исследование было разделено на две части. В рамках первой части участникам было предложено найти слова и выражения, обозначающие важные характеристики сущности (аспектные термины), и классифицировать их по тональности и обобщенным категориям. Разметка экспертом проводилась по четырехбалльной шкале (позитивный, негативный, противоречивый и нейтральный), далее была проведена проверка. Корпус для первой части состоял из отзывов на автомобили.

В рамках второго задания анализировался корпус публикаций из социальной сети «Твиттер», разметка была проведена тремя экспертами. Для разметки была использована четырехбалльная шкала.

Корпус RuSentiment. Корпус составлен из публикаций социальной сети ВКонтакте. [11] Был размечен тремя экспертами по трехбалльной шкале – позитивный, негативный и нейтральный.

Корпус Kaggle Russian News Dataset. Интернет-ресурс Kaggle [12] содержит корпус, составленный из новостей Казахстана на русском языке. Был размечен по трехбалльной шкале – позитивный, негативный и нейтральный. Метод аннотации и ресурсы неизвестны.

Корпус LinisCrowd. PolSentiLex [13] – тональный словарь, ориентированный на тексты социальных медиа, разработанный в рамках сотрудничества с Лаборатории интернет-исследований (ЛИНИС) НИУ ВШЭ. Для

него была сформирована коллекция документов, посвященных социально-политической тематике. В качестве источника данных использовались записи блог-платформы Живой Журнал и социальной сети Фэйсбук. Далее был создан краудфандинговый веб-ресурс [14], позволяющий добровольцам размечать слова и тексты онлайн, а исследователям и практикам – использовать результаты разметки.

Корпус tweets. Tweets [15] – русскоязычный корпус сообщений социальной сети «Twitter». Является одним из немногих на данный момент корпусом текстов на общую тематику. Корпус был разделен на три класса: позитивно окрашенные, негативно окрашенные и нейтральные. Сбор и разметка сообщений производились с помощью специального скрипта и привлечения экспертов. При анализе корпуса была выявленна склонность использовать чаще ту или иную часть речи в зависимости от эмотивной окраски сообщения.

1.2 Обзор существующего решения

Британская международная компания «YouGov» провела исследование, в котором выявило, насколько различается численная оценка эмотивных прилагательных в зависимости от географического положения субъекта тональности. [16] В рамках исследования «YouGov» продемонстрировали респондентам список эмотивных прилагательных и предложили оценить каждое из них по шкале 0-10, где 0 – это «сильно негативный», а 10 – «сильно позитивный». Результаты исследования изображены на рис. 1.2.

Исследование было проведено в двух странах - США и Великобритании. Исследование выявило, что респонденты из Великобритании более пессимистичны, чем респонденты из Штатов. В списке было 31 прилагательное, которое в среднем оценили на 8 из 10, но респонденты из Великобритании дали 28 из них более низкий балл. Однако, 9 самых позитивных прилагательных респонденты из Великобритании оценили более высоко.

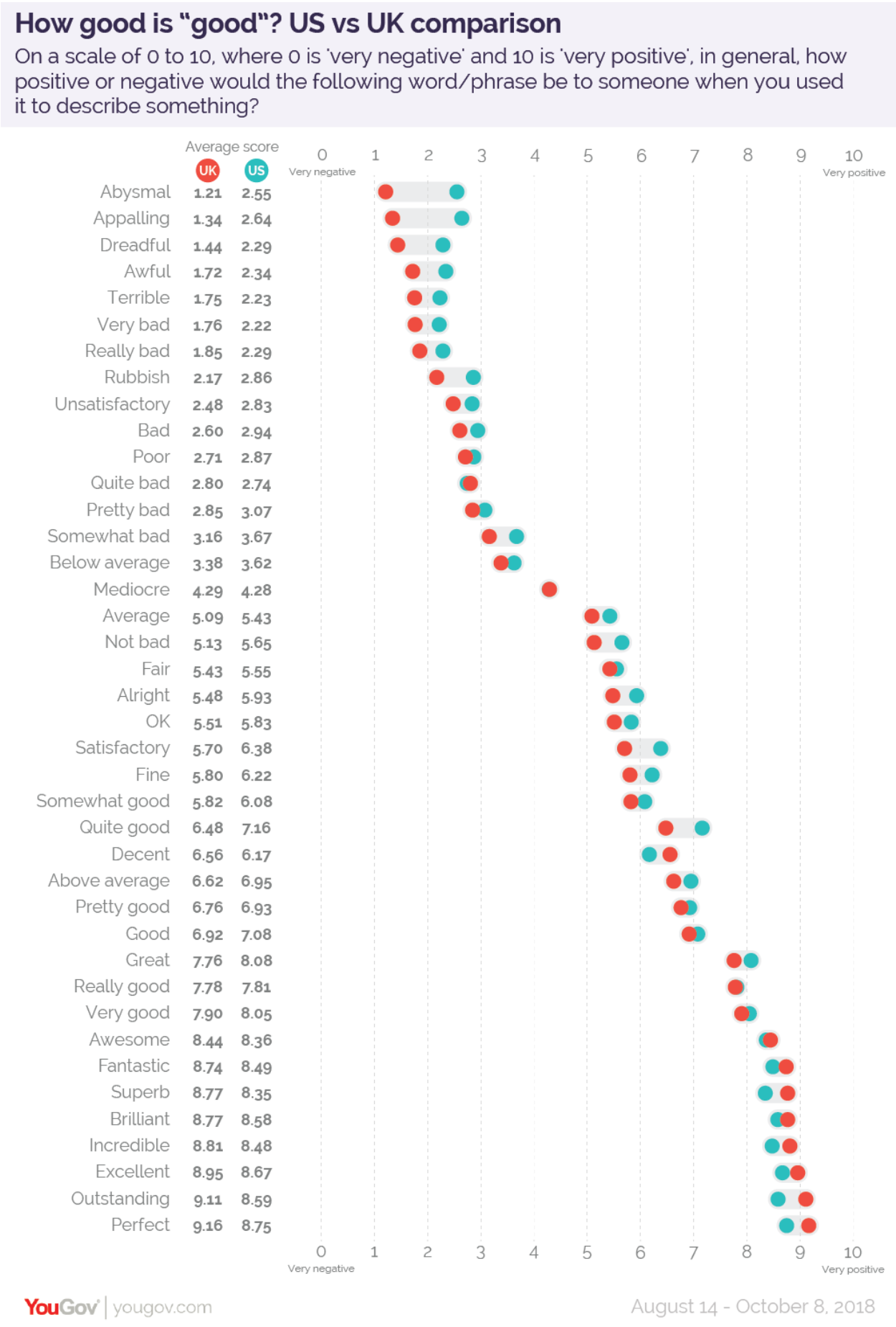


Рисунок 1.2 – Результаты исследования «YouGov»[16]

Самое большое различие наблюдается в более негативных прилагательных – разрыв оценки в слове «abysmal» составляет 1.34 балла – респонденты из Великобритании оценили на 1.21, из Штатов – на 2.55.

Похожие результаты наблюдаются и при оценке слова «appalling» – респонденты из Штатов присвоили слову в среднем более высокую оценку, чем респонденты из Великобритании – различие составляет 1.3 балла.

Самое небольшое различие наблюдается при оценке слова «mediocre» – разница не дотягивает и до половины балла. В Штатах слово оценили на 4.28, а в Великобритании – на 4.29.

1.3 Использование нереляционных баз данных

1.3.1 Нерелевантность реляционных баз данных

Нереляционные базы данных могут хранить неструктурированные данные в виде целостной сущности – база данных NoSQL не накладывает ограничений на типы хранимых данных. Потребность в использовании нереляционной базы данных обусловлена тем, что данные, полученные в результате выявления объектов и связей между ними принадлежат классу, который находится «между» Data Mining и Text Mining – Linked Objects mining(далее LOM-mining). [17]

1.3.2 Обзор моделей данных

Существует множество моделей данных в нереляционных базах данных. *Колоночное хранилище.* В колоночных нереляционных базах данных данные хранятся в ячейках, сгруппированных в колонки, а не в строки данных. Колонки логически группируются в колоночные семейства. Колоночные семейства могут состоять из практически неограниченного количества

колонок, которые могут создаваться во время работы программы или во время определения схемы. Чтение и запись происходит с использованием колонок, а не строк. Пример такой базы данных: Cassandra.[18]

Система ключ-значение. Представляет собой большую хэш-таблицу. Каждое значение сопоставляется с уникальным ключом, и хранилище ключей использует этот ключ для хранения данных, применяя к нему некоторую функцию хэширования. Выбор функции хэширования должен обеспечить равномерное распределение хэшированных ключей по хранилищу данных. [19]

Документо-ориентированные базы данных. Такие базы данных представляют собой усложненную систему «ключ-значение», которая позволяет к каждому ключу привязывать вложенные данные.

Графовая модель. Графовые базы данных предназначены для хранения взаимосвязей и навигации в них. В графовых базах данных используются узлы для хранения сущностей данных и ребра для хранения взаимосвязей между сущностями. Для запросов, соответствующих графовой модели, поиск в такой базе может быть эффективнее, чем в реляционной.

Для построения модели для решения задач, связанных с LOM-mining рекомендуют использовать методы теории графов и теории множеств. Самая подходящая модель хранилища данных – графовая. [17]

Вывод

Были проанализированы некоторые размеченные текстовые корпуса для русского языка. Ни один из проанализированных корпусов не учитывал регион проживания субъекта и его личностные характеристики. Чтобы получить более точную тональную оценку, необходимо при разметке учитывать ассоциирование с социально-демографическими группами, пол и возраст.

Для решения обозначенной проблемы необходимо разработать систему для получения данных, необходимых при построении корпуса, обладающего более тонкой разметкой, учитывающей множество различных характеристик, описанных в разделе. Задача сводится к поиску экспертов при помощи построения описания для каждого человека и поиска людей в рамках этого описания. [20]

Программный продукт, разрабатываемый в данной работе, должен содержать базу данных для хранения результатов статистического опроса о тональности прилагательных.

Тестирование должно представлять собой сопоставление оценочного прилагательного с численной оценкой по плоской шкале от 0 до 10, где 0 – это «сильно негативный», 10 – это «сильно позитивный».

База данных должна хранить информацию о респондентах и результатах тестирования. Аналогично исследованию [16], значимы следующие характеристики респондента: гендерная принадлежность, возраст, населенный пункт и место обучения.

Важная особенность задачи – это способ моделирования связи между списком тональных слов и человеком. Здесь модель представляет из себя граф, в котором узлы представляют собой людей, поставивших определенную отметку прилагательному.

2. Конструкторский раздел

2.1 Проектирование отношений сущностей

На рисунке 2.1 приведена концептуальная схема проектируемой БД в нотации Чена.

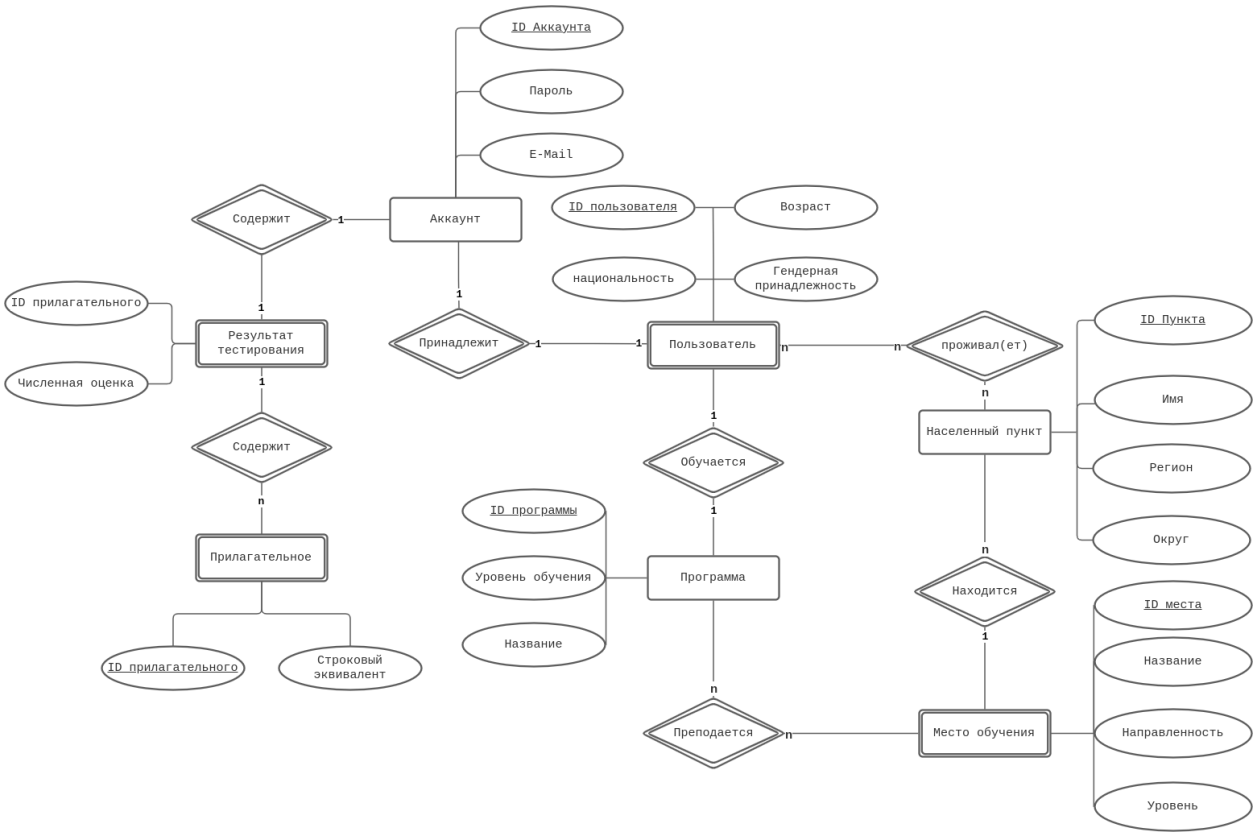


Рисунок 2.1 – ER-диаграмма сущностей базы данных в нотации Чена

Проектируемая база данных ориентирована на хранение информации, получаемой из web-приложения, содержащего систему оценки тональных прилагательных. Функционал приложения должен включать в себя регистрацию, авторизацию и возможность проставления оценок прилагательным. Соответственно, в базе данных можно выделить ряд сущностей.

1. Аккаунт хранит информацию, необходимую при регистрации – электронная почта и пароль.

2. Пользователь содержит информацию о личных характеристиках субъекта тональности – национальность, возраст и гендерная принадлежность.

3. Населенный пункт – это одна из характеристик субъекта тональности. Поскольку пользователь мог менять места проживания, связь этих двух сущностей – «многие ко многим».

4. Место обучения – сущность, содержащая характеристику об образовании, которое получает пользователь.

5. Результат тестирования включает в себя оценку тонального прилагательного, присвоенная пользователем во время рабочей сессии. Возможные значения оценки – числа от «1» до «10».

6. Тональное прилагательное.

2.2 Ролевая модель

В ролевой модели[21] операции, которые необходимо выполнять в рамках какой-либо служебной обязанности пользователя системы, группируются в набор, называемый «ролью».

При использовании ролевой политики управление доступом осуществляется в две стадии: для каждой роли указывается набор полномочий, представляющий набор прав доступа к объектам или частям приложения, затем каждому пользователю назначается его роль.

В разрабатываемой системе выделены три роли, каждой из которых соответствует конкретный функционал системы:

1. Опрашиваемому – функционал регистрации и авторизации, а также прохождения опроса и просмотр его результатов.

2. Модератору – функционал просмотра списка всех пользователей системы.

3. Администратору – функционал просмотра результатов опроса любого пользователя и общей статистики по опросу.

2.3 Нефункциональные характеристики графовых БД

Зачастую в приложениях, активно использующих операции ввода-вывода, при выполнении одной бизнес-операции происходят массовое чтение и запись связанных между собой данных. В графовых базах данных выполняется несколько операций внутри логического подграфа общего набора данных. Подобное множество можно преобразовать в набор более крупных, тесно связанных операций.

В реляционных базах данных с увеличением размеров и количества данных начинают проявляться недостатки соединения таблиц и ухудшается производительность. Использование смежности без индексов позволяет графовым базам данных перемещаться по сложным соединениям в графе эффективнее, независимо от общего размера набора данных.

2.3.1 Нерелевантность внешних кэшей

Графовые базы данных имеют более высокую эффективность поиска за счет графовой организации хранения данных, благодаря этому разница в скорости доступа к данным в основной СУБД и внешних кэшах незначительна.[22]

В случае реляционных СУБД кэши используются для выделения «горячих» данных и быстрого обращения к ним. Кэширующие базы данных как правило имеют тип «ключ-значение», реализующие хеш-таблицу, в которой находится уникальный ключ и указатель на конкретный объект данных. Релеватность использования кэшей с реляционными СУБД достигается за счет не только меньшей ассимптотической сложности, но и хранения в данных в оперативной памяти вместо диска, используемого самой СУБД.[23] Таким образом, из общего набора данных выделяются «горячие», скорость доступа к которым следует по возможности увеличивать. Ассимптотическая сложность алгоритма доступа к данным в графовых СУБД кэш-хранилищ совпадает. К тому же, использование сторонних кэширующих хранилищ накладывает ограничения на размерность и связан-

ность данных, достаточные, чтобы считать использование их лишенным смысла.

2.4 Проектирование клиентского приложения

На рисунке 2.2 представлена диаграмма вариантов использования приложения.

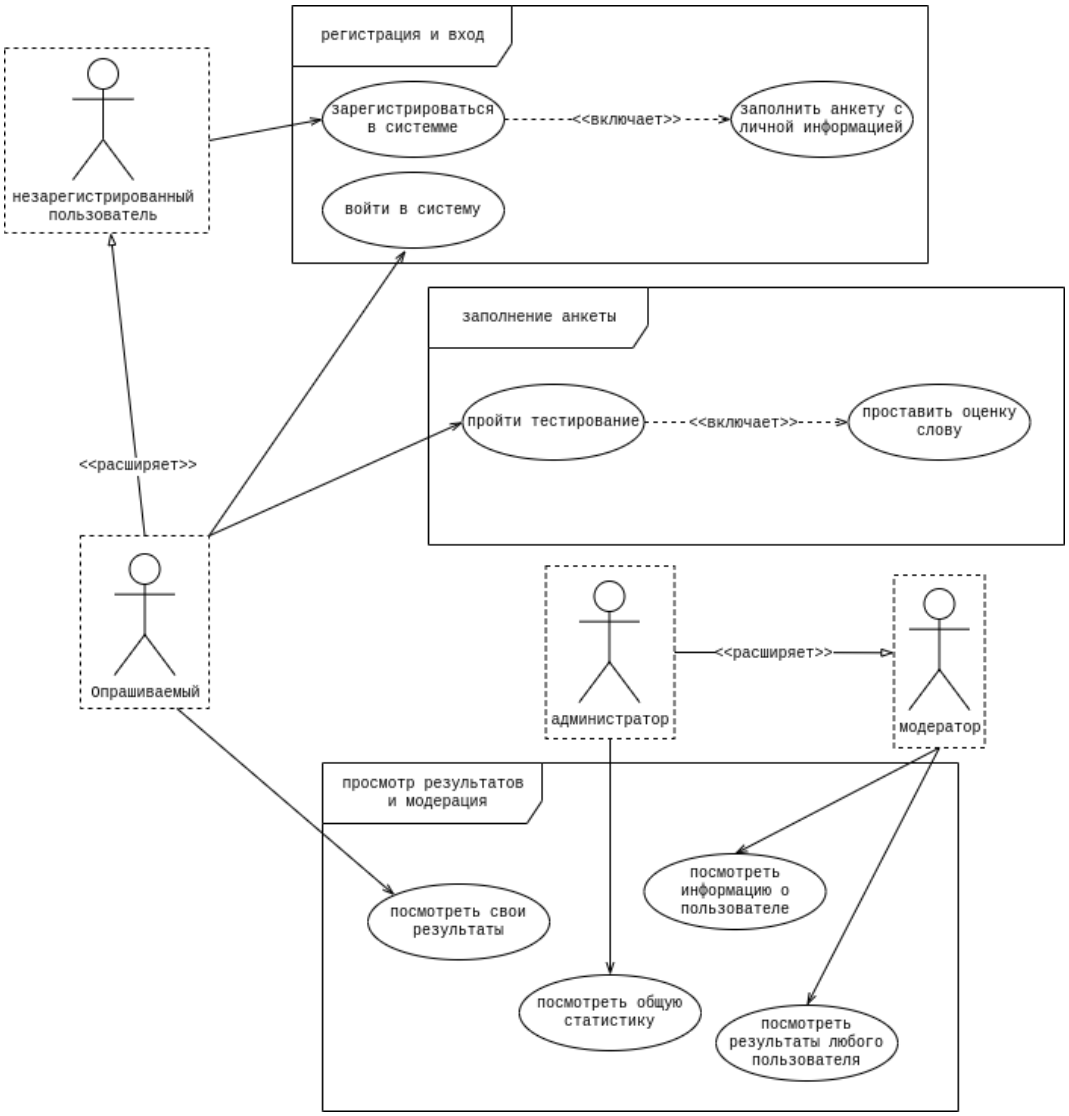


Рисунок 2.2 – Диаграмма вариантов использования системы

Пользователь системы, имеющий роль «Модератор», может просматривать информацию о всех пользователях, сохраненных в базе данных. Интерфейс пользователя с ролью «Администратор» расширяет интерфейс для «Модератора» возможностью просматривать общую статистику опроса и результатов всех пользователей.

В случае посещения страниц, для которых у пользователя не соответствующая роль, в интерфейсе отображается соответствующее сообщение и доступ к запрашиваемому функционалу не предоставляется.

Интерфейс незарегистрированного пользователя включает возможность регистрироваться и осуществлять вход в систему. После регистрации или авторизации интерфейс расширяется: опрашиваемый может проходить тестирование и смотреть свои результаты.

На рисунке 2.3 представлена модель бизнес-процессов в нотации BPMN.

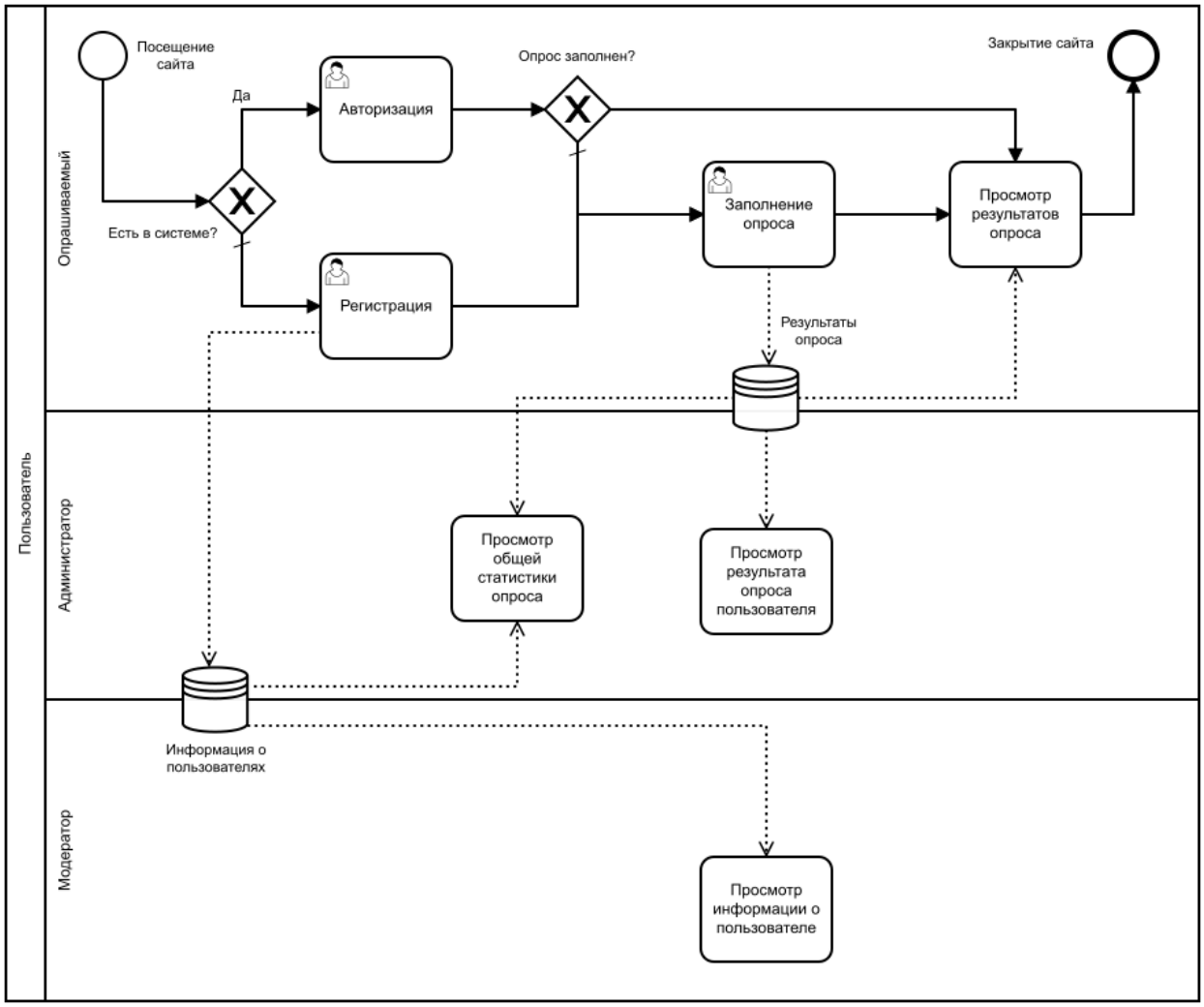


Рисунок 2.3 – модель бизнес-процессов в нотации BPMN

Бизнес-процесс состоит одного пула «пользователь», поделенного на три дорожки – администратор, модератор и опрашиваемый. Дорожки определяются ролью пользователя, хранимой в базе данных.

Опрашиваемый посещает сайт и проходит либо регистрацию, либо авторизацию в зависимости от того, был ли он уже зарегистрирован в системе. Далее ему предлагается пройти опрос либо просмотреть результаты в случае если опрос уже пройден. Результаты опроса после его заполнения отправляются в хранилище данных.

Модератор имеет доступ к части хранилища с информацией о всех зарегистрированных пользователях и может просматривать информацию о любом пользователе. Администратор имеет доступ к результатам и может просматривать общую статистику опроса.

На рисунке 2.4 приведена диаграмма последовательностей для процесса получения результатов опроса от пользователя.

Для заполнения опроса требуется пройти авторизацию. Если пользователь не авторизован, то он получит соответствующее сообщение об ошибке.

При посещении опрашиваемым страницы заполнения опроса на сервер поступает запрос на получение списка прилагательных, которые хранятся в базе данных. Из базы данных список поступает на сервер, затем на клиентское приложение и возвращается пользователю. Пользователь размечает прилагательные, на сервер отправляется список прилагательных с разметкой и результат записывается в базу данных.

При запросе на просмотр результата на сервере происходит анализ пользовательских данных и сервер отправляет запрос в базу на поиск результата опроса пользователя..

На рисунке 2.5 изображена диаграмма последовательностей для процесса получения полного списка пользователей через защищенную страницу. Выполнять данное действие может только модератор, поэтому перед посещением страницы происходит проверка прав пользователя, отправившего запрос.

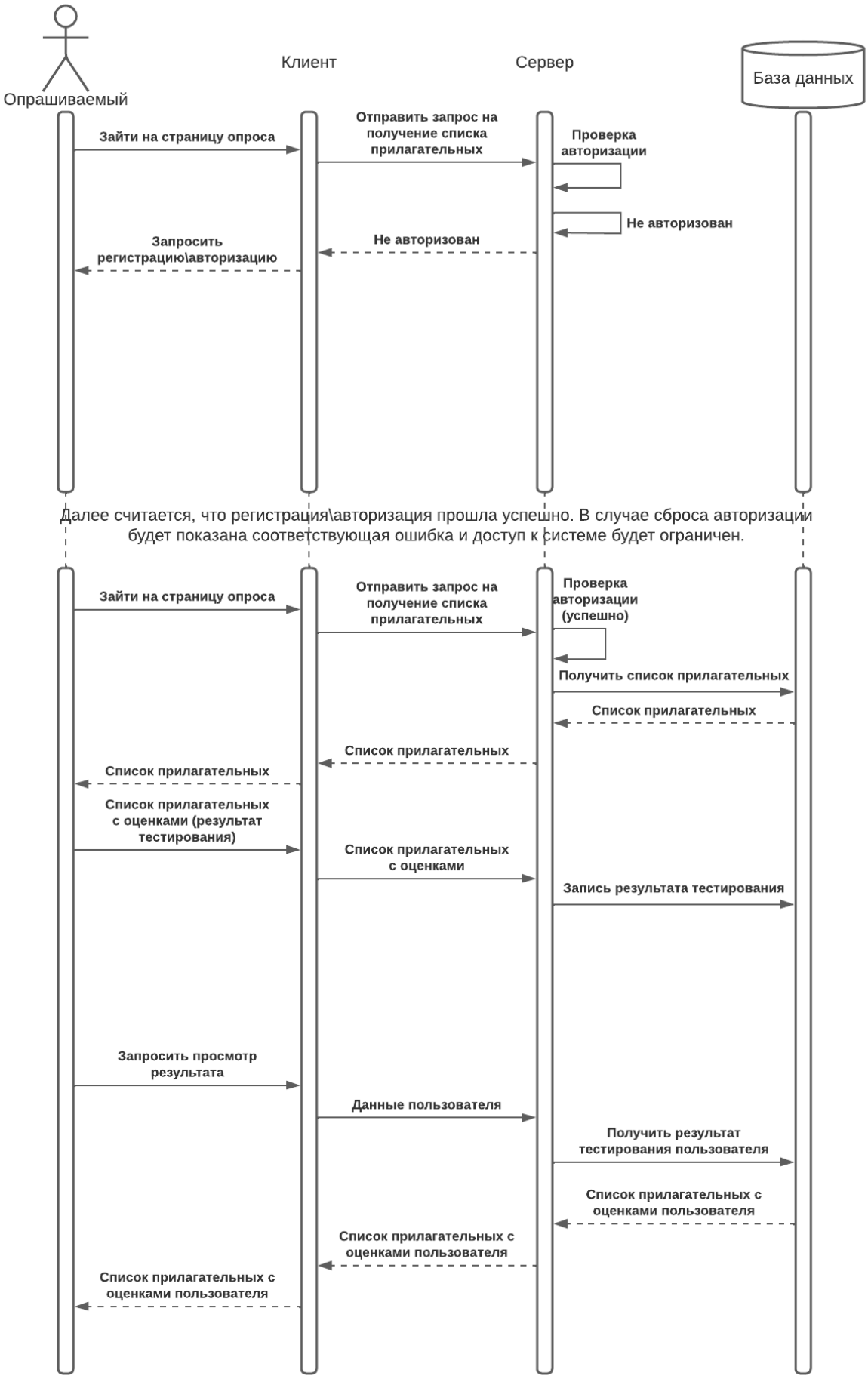


Рисунок 2.4 – Диаграмма последовательностей для процесса получения результатов опроса от пользователя

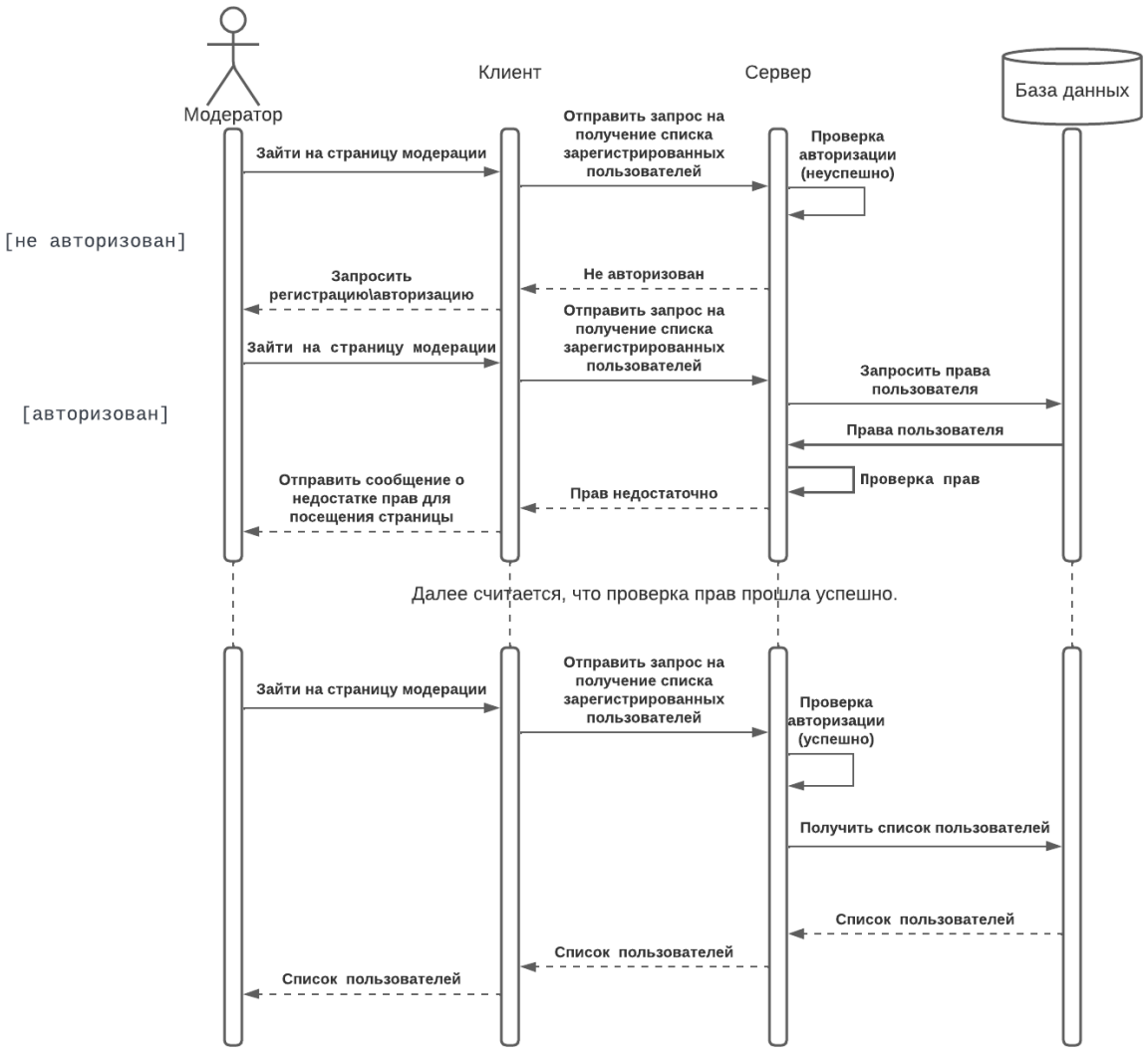


Рисунок 2.5 – Диаграмма последовательностей для процесса получения списка зарегистрированных пользователей

На рисунке 2.6 изображена диаграмма последовательностей для процесса получения статистики опроса через защищенную страницу. Выполнять данное действие может только администратор, поэтому перед посещением страницы происходит проверка прав пользователя, отправившего запрос.

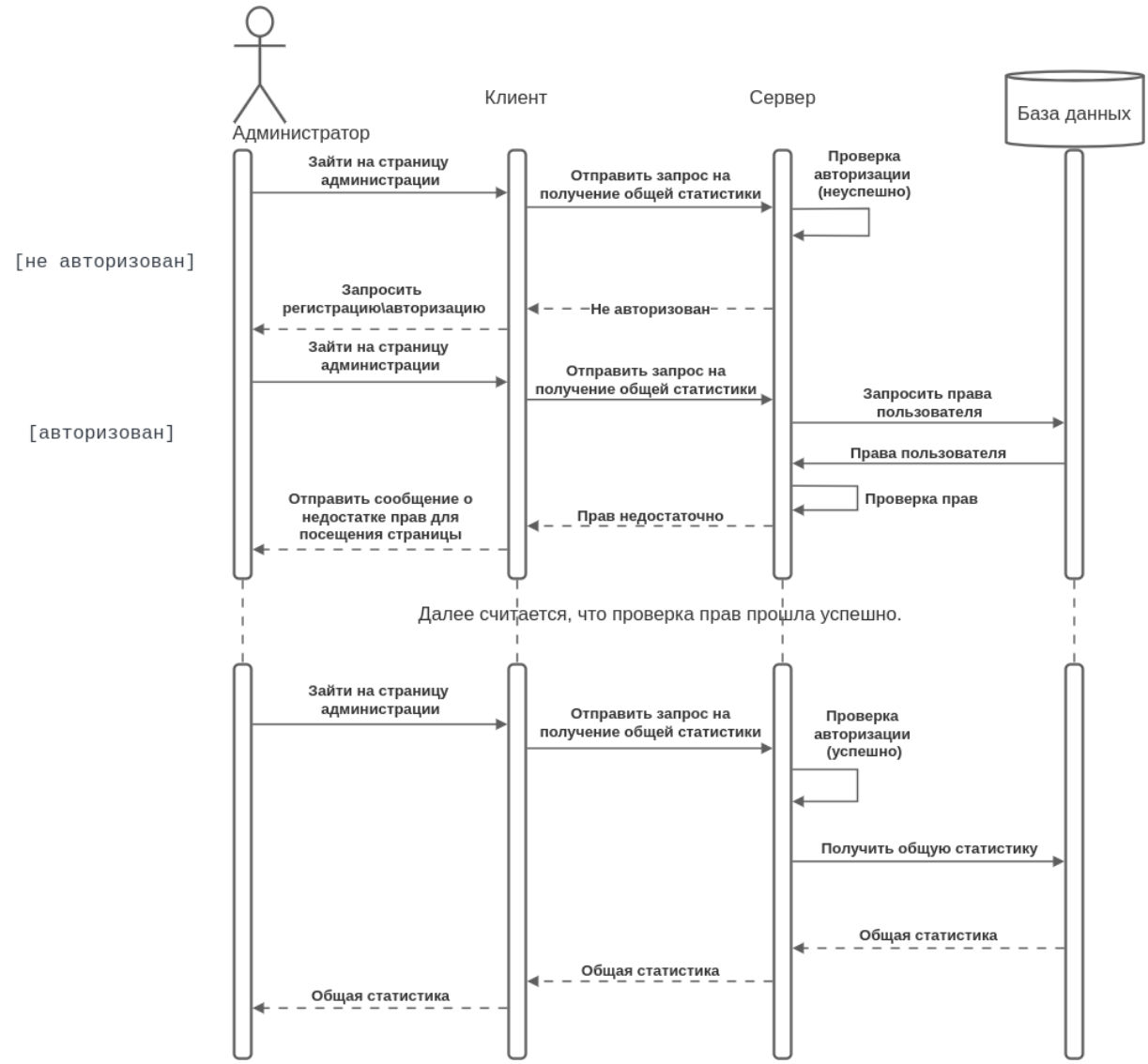


Рисунок 2.6 – Диаграмма последовательностей для процесса получения общей статистики опроса

Вывод

Проектируемая база данных хранит информацию, получаемую из web-приложения, содержащего систему оценки тональных прилагательных. В базе выделен соответствующий ряд сущностей, приведенный на рисунке 2.1. Для базы выделено три роли: «Опрашиваемый», «Модератор» и «Администратор», каждая из которых соответствует дорожке пула «пользователь», присутствующей в бизнес-процессе (рисунок 2.3).

Использование сторонних кэширующих хранилищ накладывает ограничения на размерность и связанность данных, достаточные, чтобы считать использование их лишенным смысла – графовые базы данных имеют более высокую эффективность поиска за счет графовой организации хранения данных.

3. Технологический раздел

3.1 Выбор средств разработки

В разделе 1.3.2 обоснована необходимость использования графовых баз данных. В качестве системы управления базами данных была выбрана СУБД Neo4j[24], поскольку может быть доступна из ПО, написанного на практически любом языке с использованием языка запросов Cypher, через обращение к маршруту отдельным HTTP методом или через протокол «bolt».

Язык Golang[25] обладает достаточным набором инструментов для написания одностраничных приложений: предоставляет базовые возможности маршрутизации, интерфейс работы с базами данных и конфигурационными файлами, поэтому он был выбран в качестве реализации клиентского приложения. Базовый интерфейс маршрутизации был расширен с помощью пакета Gorilla[26].

3.2 Реализация клиентского приложения

Модельные структуры для сущностей соответствуют сущностям базы данных, представленным на рисунке 2.1.

На листинге 3.1 представлена модельная структура сущности «Аккаунт».

Листинг 3.1 – Модельная структура сущности «Аккаунт»

```
type Account struct {
    UUID              string `json:"uuid"`
    Username          string `json:"username,omitempty"`
    Email             string `json:"email,omitempty"`
    Password          string `json:"password,omitempty"`
    Role              string `json:"role,omitempty"`
    EncryptedPassword string `json:"-"`
}
```

На листинге 3.2 представлена модельная структура сущности «Пользователь».

Листинг 3.2 – Модельная структура сущности «Пользователь»

```
type User struct {
    UUID   string `json:"uuid,omitempty"`
    Age    int    `json:"age"`
    Race   string `json:"race"`
    Gender string `json:"gender"`
}
```

На листинге 3.3 представлены модельные структуры сущностей «Населенный пункт», «Программа» и «Место обучения».

Листинг 3.3 – Модельные структуры сущностей «Населенный пункт» «Программа» и «Место обучения»

```
type EducationPlace struct {
    UUID string `json:"uuid"`
    Name string `json:"name"`
}
type EducationProgram struct {
    Field string `json:"field"`
    Level string `json:"level"`
}
type Location struct {
    UUID     string `json:"uuid"`
    Name     string `json:"name"`
    Region   string `json:"region"`
    District string `json:"district"`
}
```

На листинге 3.4 представлены модельные структуры сущностей «Результат тестирования» и «Прилагательное».

Листинг 3.4 – Модельные структуры сущностей «Результат тестирования» и «Прилагательное»

```
type Answer struct {
    Word string
    Mark int
}
type Poll struct {
    Answer []Answer
}
```

База данных хранит пароли в качестве функции свертки с модифика-

тором входа. Хеширование и проверка пароля представлены на листинге 3.5.

Листинг 3.5 – Хеширование и проверка пароля

```
func (a *Account) EncryptPassword() error {
    if len(a.Password) > 0 {
        enc, err := encryptString(a.Password)
        if err != nil {
            return err
        }
        a.EncryptedPassword = enc
    }
    return nil
}

func (a *Account) ComparePassword(password string) bool {
    return bcrypt.CompareHashAndPassword(
        []byte(a.EncryptedPassword), []byte(password)) == nil
}

func encryptString(p string) (string, error) {
    b, err := bcrypt.GenerateFromPassword([]byte(p), bcrypt.MinCost)
    if err != nil {
        return "", err
    }
    return string(b), nil
}
```

Сервис, работающий с базой данных аналогичен для всех модельных структур. На листинге 3.6 представлен механизм установки соединения и регистрации пользователя.

Листинг 3.6 – Установка соединения и отправка запросов

```
func newDB(DBUri, DBUsername, DBPassword string) (neo4j.Driver, error) {
    db, err := neo4j.NewDriver(DBUri, neo4j.BasicAuth(DBUsername,
         DBPassword, ""))
    if err != nil {
        return nil, err
    }
    return db, nil
}

func (r *AccountRepository) Create(a *model.Account) error {
    if err := a.Validate(); err != nil {
        return err
    }
```

```
14      if err := a.EncryptPassword(); err != nil {
15          return err
16      }
17      session := r.storage.db.NewSession(neo4j.SessionConfig{AccessMode:
18           neo4j.AccessModeWrite, DatabaseName: "goodisgood"})
19      defer session.Close()
20
21      if _, err := session.WriteTransaction(func(
22      tx neo4j.Transaction) (interface{}, error) {
23          return r.createQuery(tx, a)
24      }); err != nil {
25          return err
26      }
27      return nil
28  }
29
30  func (r *AccountRepository) createQuery(tx neo4j.Transaction,
31          a *model.Account) (*model.Account, error) {
32      result, err := tx.Run(
33      `create (a: account{id: apoc.create.uuid(),
34                          username: $username,
35                          email: $email,
36                          password:$password, role:"user"})
37      return a.id as id`,
38      map[string]interface{}{
39          "username": a.Username,
40          "email":    a.Email,
41          "password": a.EncryptedPassword,
42      },
43      )
44
45      if err != nil {
46          return nil, err
47      }
48
49      record, err := result.Single()
50
51      if err != nil {
52          return nil, err
53      }
54
55      id, _ := record.Get("id")
56      a.UUID = id.(string)
57
58      return a, nil
59  }
```

Механизм маршрутизации запроса на авторизацию пользователя представ-

лен на листинге 3.7

Листинг 3.7 – Маршрутизация запроса аутентификации

```
func (s *server) handleSessionsCreate() http.HandlerFunc {
    type request struct {
        Username string `json:"username"`
        Email    string `json:"email"`
        Password string `json:"password"`
    }
    return func(w http.ResponseWriter, r *http.Request) {
        req := &request{}
        if err := json.NewDecoder(r.Body).Decode(req); err != nil {
            s.logger.Error(w, r, http.StatusBadRequest, err)
            return
        }
        a, err := s.ustorage.Account().FindByEmail(req.Email)
        if err != nil || !a.ComparePassword(req.Password) {
            if err != nil {
                s.logger.Infof(err.Error())
            }
            s.error(w, r, http.StatusUnauthorized,
                errInvalidEmailOrPassword)
            return
        }
        session, err := s.sessionStorage.Get(r, sessionName)
        if err != nil {
            s.error(w, r, http.StatusInternalServerError, err)
            return
        }
        session.Values["account_id"] = a.UUID
        if err := s.sessionStorage.Save(r, w, session); err != nil {
            s.error(w, r, http.StatusInternalServerError, err)
            return
        }
        s.respond(w, r, http.StatusOK, nil)
    }
}
func (s *server) error(w http.ResponseWriter, r *http.Request,
     code int, err error) {
    s.respond(w, r, code, map[string]string{"error": err.Error()})
}
func (s *server) respond(w http.ResponseWriter, r *http.Request,
     code int, data interface{}) {
    w.WriteHeader(code)
    if data != nil {
        json.NewEncoder(w).Encode(data)
    }
}
```

3.3 Запросы к базе данных

Neo4j использует язык запросов Cypher[27].

На листинге 3.8 приведен запрос связи места обучения с пользователем.

Листинг 3.8 – Запрос связи места обучения с пользователем

```
match (u:user)-[o:OWNS]->(a:account{id:$auuid})
match(e:education_place{name: $ename})
merge((u)-[s:STUDIES{field:$field, level: $level}]->(e))
```

На листинге 3.9 приведен запрос для создания пользователя, связанного с конкретным аккаунтом.

Листинг 3.9 – Запрос для создания пользователя связанного с конкретным аккаунтом

```
match
    (a:account)
where
    a.id=$auuid
create (u:user{id: apoc.create.uuid(),
                race: $race,
                age: $age,
                gender:$gender})
create (u)-[o:OWNS]->(a)
return a.id as id
```

На листинге 3.11 приведен запрос для заполнения опроса (разметки прилагательного).

Листинг 3.10 – Запрос для заполнения опроса

```
match (u:user)-[o:OWNS]->(a:account{id:$auuid})
match (w:word{name: $wname})
optional match (u)-[m:MARKED]->(w)
call apoc.do.when(
m is null,
' create (u)-[m:MARKED{mark: mrk}]->(w)
return m is not null as ok',
'return true as ok',
{u:u, w:w, mrk: $mark}) yield value
return value.ok as ok
```

На листинге 3.11 приведен запрос для получения общей статистики.

Листинг 3.11 – Запрос для получения общей статистики

```
match ()-[r:MARKED]->(w:word)
with w.name AS name, collect(r.mark) AS marklist, COUNT(r.mark) AS cnt
unwind marklist as ml
return distinct name, avg(ml) as avg
```

3.4 Тестирование

Тестирование осуществлялось с помощью стандартных механизмов Golang – пакета Testing[28]. На листинге 3.12 приведено тестирование функции поиска пользователя.

Листинг 3.12 – Тестирование функции поиска пользователя

```
func TestAccountRepository_FindByEmail(t *testing.T) {
    d := neo4jStorage.TestDB(t, DBuri, DBUsername, DBPassword)
    s := neo4jStorage.NewStorage(d)
    e2 := "gopher@gopher.go"
    a, err := s.Account().FindByEmail(e2)
    assert.NoError(t, err)
    assert.NotNil(t, a)
}
func TestAccountRepository_Find(t *testing.T) {
    d := neo4jStorage.TestDB(t, DBuri, DBUsername, DBPassword)

    s := neo4jStorage.NewStorage(d)
    u1, err := s.Account().FindByEmail("gopher@gopher.go")
    if err != nil {
        t.Fatal(err)
    }
    e2 := u1.UUID
    a, err := s.Account().Find(e2)
    logrus.New().Info(err)
    assert.NoError(t, err)
    assert.NotNil(t, a)
}
```

4. Исследовательский раздел

4.1 Технические характеристики

Тестирование выполнялось на устройстве со следующими техническими характеристиками:

- операционная система Ubuntu 22.04 LTS;
- память 7 GiB;
- процессор Intel(R) Core(TM) i3-8145U CPU @ 2.10GHz.

4.2 Оптимизация производительности

Существуют три пути оптимизации производительности [29]:

- увеличение размера кучи виртуальной машины Java;
- увеличение соотношения объема кэшированных данных к общему объему хранимых данных;
- применение быстрых дисковых накопителей: SSD или промышленных флеш-накопителей.

В работе рассматривается первый подход.

4.2.1 Настройка размера кучи виртуальной машины Java

Куча виртуальной машины Java – отдельная динамическая область памяти, выделяемая СУБД и используемая для хранения Java-объектов. Для оптимизации производительности важно учитывать поведение сборщика мусора Java, который удаляет неиспользуемые объекты.

Размер кучи может быть изменен с помощью настроек `dbms.memory.heap.initial_size` и `dbms.memory.heap.max_size`, расположенных в конфигурационных файлах системы.

Тестовый запрос включает в себя получение всех узлов типа «населенный пункт».

В таблице 4.2 приведены тестовые данные для размера кучи виртуальной машины Java равного 256Мб. Среднее значение для выборки – 18 мс, медианное – 17 мс.

В таблице 4.4 приведены тестовые данные для размера кучи виртуальной машины Java равного 512Мб. Среднее значение для выборки – 17 мс, медианное – 16 мс.

4.2.2 Настройка размера страничного кэша

Одна из возможных конфигураций СУБД Neo4j – возможность контроля количество паямти, используемое для кэширования файлов – вытеснения страниц в разделяемую память виртуальной машины Java. Стандартный страничный кэш Neo4j вычисляется из предположения, что машина, на которой развернут сервер, предназначена для запуска Neo4j и эвристически сконфигурирована на 50% RAM не считая памяти, требуемой для размещения кучи виртуальной машины Java. Размер страничного кэша может быть изменен с помощью настройки `dbms.memory.pagecache.size`.

В таблице 4.6 приведены тестовые данные для размера кучи виртуальной машины Java равного 256Мб и страничного кэша равного 256Мб. Среднее значение для выборки – 15 мс, медианное – 13 мс. В таблице 4.8 приведены тестовые данные для размера кучи виртуальной машины Java равного 256Мб и страничного кэша равного 256Мб. Среднее значение для выборки – 14 мс, медианное – 13 мс.

Таблица 4.1 – Тестовые данные для размера кучи 256Мб

Запрос (шт.)	Время(мс.)
0	90
1	45
2	39
3	40
4	49
5	32
6	31
7	34
8	29
9	30
10	41
11	24
12	27
13	25
14	23
15	25
16	22
17	25
18	24
19	27
20	24
21	21
22	36
23	24
24	28
25	22
26	23
27	22
28	21
29	19
30	25
31	18
32	23
33	19
34	20
35	23

Таблица 4.2 – Тестовые данные для размера кучи 256Мб

Запрос (шт.)	Время(мс.)
566	13
567	14
568	14
569	14
570	18
571	13
572	13
573	22
574	13
575	14
576	13
577	17
578	15
579	15
580	24
581	15
582	14
583	13
584	19
585	14
586	14
587	14
588	14
589	14
590	13
591	16
592	15
593	18
594	12
595	13
596	15
597	14
598	14
599	14
600	15

Таблица 4.3 – Тестовые данные для размера кучи 512Мб

Запрос (шт.)	Время(мс.)
0	102
1	44
2	48
3	45
4	33
5	31
6	29
7	33
8	28
9	29
10	47
11	26
12	34
13	32
14	26
15	41
16	34
17	28
18	28
19	26
20	24
21	25
22	23
23	20
24	29
25	24
26	22
27	19
28	21
29	17
30	26
31	20
32	21
33	27
34	29
35	18

Таблица 4.4 – Тестовые данные для размера кучи 512Мб

Запрос (шт.)	Время(мс.)
566	14
567	16
568	14
569	13
570	12
571	15
572	18
573	13
574	13
575	14
576	13
577	14
578	12
579	11
580	13
581	16
582	15
583	14
584	33
585	15
586	14
587	14
588	16
589	15
590	15
591	15
592	13
593	18
594	14
595	17
596	46
597	62
598	32
599	27
600	25

Таблица 4.5 – Тестовые данные для страничного кэша равного 512Мб

Запрос (шт.)	Время(мс.)
0	11
1	9
2	13
3	13
4	13
5	12
6	12
7	15
8	13
9	11
10	15
11	12
12	12
13	15
14	13
15	16
16	11
17	11
18	13
19	11
20	13
21	13
22	12
23	11
24	14
25	12
26	12
27	11
28	13
29	14
30	14
31	12
32	11
33	13
34	11
35	11

Таблица 4.6 – Тестовые данные для страничного кэша равного 512Мб

Запрос (шт.)	Время(мс.)
565	13
566	10
567	12
568	11
569	14
570	11
571	13
572	10
573	11
574	21
575	10
576	12
577	11
578	13
579	11
580	12
581	12
582	12
583	13
584	18
585	13
586	18
587	20
588	16
589	19
590	19
591	22
592	17
593	14
594	16
595	14
596	16
597	14
598	24
599	16
600	14

Таблица 4.7 – Тестовые данные для страничного кэша равного 256Мб

Запрос (шт.)	Время(мс.)
0	12
1	13
2	12
3	12
4	13
5	12
6	13
7	13
8	12
9	17
10	11
11	15
12	16
13	13
14	14
15	14
16	12
17	13
18	15
19	13
20	12
21	14
22	18
23	13
24	12
25	13
26	12
27	13
28	13
29	11
30	14
31	12
32	18
33	14
34	12
35	14

Таблица 4.8 – Тестовые данные для страничного кэша равного 256Мб

Запрос (шт.)	Время(мс.)
565	12
566	14
567	12
568	12
569	36
570	23
571	14
572	13
573	29
574	15
575	11
576	15
577	12
578	12
579	11
580	12
581	12
582	12
583	16
584	12
585	11
586	11
587	14
588	11
589	13
590	11
591	14
592	12
593	13
594	15
595	13
596	12
597	10
598	12
599	13
600	12

4.3 Вывод

При настройке кучи JVM рекомендуется увеличивать ее размер постепенно, контролируя сборку мусора, пока не будет обнаружена точка резкого увеличения затрат на сборку мусора.[29] В рассмотренном случае точка увеличения затрат на сборку мусора – 256 Мб.

На рассмотренном тестовом наборе размер данных не позволяет достичь значительного прироста производительности за счет изменения количества паямти, используемое для кэширования файлов. Однако, при явной конфигурация этого параметра наблюдается уменьшение среднего времени, затрачиваемого на выполнения запроса.

ЗАКЛЮЧЕНИЕ

Автоматизация определения тональности текста – алгоритмически сложная задача, включающая в себя ряд не менее сложных подзадач. Одна из них – определение характеристик субъекта тональности и выявление связи между ними и тональной оценкой. Классический подход на основе массовых опросов предоставляет более точные данные, чем подход на основе анализа общих словарей.

Данные, полученные при массовом опросе принадлежат классу Linked Object Mining. Такой набор связей является неструктурированным, следовательно, не подходит для хранения в реляционных базах данных. Графовые базы данных не накладывают ограничений на хранение такого типа хранимых данных.

В представленной работе проведена работа с графовой базой данных с использованием СУБД Neo4j. Было проведено нагрузочное тестирование с изменением размера кучи виртуальной машины Java и контроля количества памяти, используемого для кэширования файлов.

При корректной настройке этих параметров под нужды системы, на которой работает Neo4j скорость запросов в среднем возрастает.

Поскольку Neo4j обладает открытым исходным кодом, перспективой исследования может быть изучение программной реализации СУБД для наиболее удачного подбора параметров конфигурации.

Список литературы

1. *Гаспаров Б.* Язык, память, образ. Лингвистика языкового существования. — Новое литературное обозрение, 1996.
2. *Пазельская А. Г.*, *Соловьев А. Н.* Метод определения эмоций в текстах на русском языке // Компьютерная лингвистика и интеллектуальные технологии: «Диалог-2011». Сб. научных статей / Вып. 11. — 2011.
3. *Kanayama H.*, *Nasukawa T.* Fully automatic lexicon expansion fordomain-oriented sentiment analys // Proceedings of EMNLP-2006. — 2006.
4. *Дудина В. И.*, *Юдина Д. И.* Извлекая мнения из сети Интернет: могут ли методы анализа текстов заменить опросы общественного мнения? // Мониторинг общественного мнения : Экономические и социальные перемены. — 2017.
5. *Zubarevich N.* Russia 2025: Scenariosfor the Russian Future. // / под ред. M. Lipman, N. Petrov. — Palgrave Macmillan, London, 2013. — Гл. Four Russias: Human Potential and Social Differentiation of Russian Regions and Cities. С. 67—85.
6. *Матвеева Т.* Экспрессивность русского слова. — LAP LAMBERT Academic Publishing, 2013.
7. *Щекотин Е.*, *Мягков М.*, *Гойко В.*, *Кашпур В.*, *Г.Ю. К.* Субъективная оценка (не)благополучия населения регионов РФ на основе данных социальных сетей // Мониторингобщественного мнения : Экономические исоциальные перемены. — 2020.
8. *Котельников Е. В.* Текущее состояние русскоязычных корпусовдля анализа тональности текстов // Computational Linguistics and Intellectual Technologies:Proceedings of the International Conference Dialogue 2021. — 2021.

9. *Четверкин И.*, *Браславский П. И.*, *Лукашевич Н.* Sentiment Analysis Track at ROMIP 2011 // Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference «Dialog-2012». — 2012.

10. *Loukachevitch N.*, *Blinov P.*, *Kotelnikov E.*, *Rubtsova Y.*, *Ivanov V.*, *Tutubalina E.* SentiRuEval: Testing object-oriented sentiment analysis systems in Russian // Computational Linguistics and IntellectualTechnologies: Proceedings of the International Conference «Dialog-2015». — 2015.

11. *Rogers A.*, *Romanov A.*, *Rumshisky A.*, *Volkova S.*, *Gronas M.*, *Gribov A.* RuSentiment: An Enriched Sentiment Analysis Datasetfor Social Media in Russian // Proceedings of the 27th International Conference onComputational Linguistics. — 2018.

12. Sentiment Analysis in Russian. — Дата обновления: 04.08.2022. — URL: `https://www.kaggle.com/c/sentiment-analysis-in-russian`.

13. *Koltsova O. Y.*, *Alexeeva S. V.*, *N. K. S.* An Opinion Word Lexicon and a Training Dataset for Russian Sentiment Analysis of Social Media // Computational Linguistics and Intellectual Technologies: Proceedings of the International Conference "Dialogue 2016". — 2016.

14. Общедоступный тональный словарь PolSentiLex и краудсорсинговая платформа для его создания. — Дата обновления: 04.08.2022. — URL: `http://linis-crowd.org/`.

15. *Рубцова Ю.* Автоматическое построение и анализ корпуса коротких текстов (постов микроблогов) для задачи разработки и тренировки тонового классификатора // инженерия знаний и технологии семантического веба. — 2012.

16. How good is «good»? — Дата обновления: 02.01.2022. — URL: `https://yougov.co.uk/topics/lifestyle/articles-reports/2018/10/02/how-good-good`.

17. *Попов И.*, *Фролкина Н.* Анализ связанных объектов и визуализация результатов // Доклады международной конференции Диалог 2004. — 2004.

18. Apache Cassandra Documentation. — URL: https://cassandra.apache.org/_/index.html.

19. Нереляционные данные и базы данных NoSQL. — Дата обновления: 04.09.2022. — URL: https://docs.microsoft.com/ru-ru/azure/architecture/data-guide/big-data/non-relational-data.

20. *Petkova D.*, *Croft W. B.* Hierarchical language models for expert finding in enter-prise corpora // Proceedings of the IEEE International Conference on Tools withArtificial Intelligence. — 2006.

21. *Leonard J. L.*, *D. Elliott B.* Secure Computer Systems: A Mathematical Model // MITRECorporation Technical Report 2547, Volume II. — 1973.

22. *Sholichah R.*, *Imrona M.*, *Alamsyah A.* Performance Analysis of Neo4j and MySQL Databases using Public Policies Decision Making Data. — 2020.

23. Обзор кэширования. — Дата обновления: 05.09.2022. — URL: https://aws.amazon.com/ru/caching/.

24. Neo4j documentation. — Дата обновления: 04.09.2022. — URL: https://neo4j.com/docs/.

25. Documentation. — URL: https://go.dev/doc/.

26. Gorilla web toolkit. — URL: https://www.gorillatoolkit.org/.

27. Cypher Query Language. — URL: https://neo4j.com/developer/cypher/.

28. testing. — URL: https://pkg.go.dev/testing.

29. *Ян Р.*, *Джим В.*, *Эифрем Э.* Графовые базы данных. Новые возможности для работы со связанными данными. — ДМК Пресс, 2016.

30. Cassandra Documentation: Storage Engine. — Дата обновления: 04.09.2022. — URL: https://cassandra.apache.org/doc/latest/cassandra/architecture/storage_engine.html.

31. DB-Engines Ranking of Graph DBMS. — Дата обновления: 09.04.2022. — URL: https://db-engines.com/en/ranking/graph%20dbms.

32. *Balog K.*, *Azzopardi L.*, *Rijke M. de.* Formal models for expert finding in enterprisecorpora // Proceedings of the Annual International ACM SIGIR Conference on Re-search and Development in Information Retrieval. — 2006.