МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ И.С. ТУРГЕНЕВА»

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по направлению подготовки 01.04.02 Прикладная математика и информатика

направленность (профиль): Технологии параллельных и распределённых вычислений

Студента Тютюнова Игоря Андреевича шифр 205017

Факультет (институт): Физико – математический

Разработка методов оптимизации взаимодействия веб-сайта с базой данных

Студент Тютюнов И.А.

Руководитель: к.ф.-м.н., доц. Федяев Ю.С.

Зав. Кафедрой Дорофеева В. И.

Орёл 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИМЕНИ И.С. ТУРГЕНЕВА»

Факультет физико-математический

Кафедра информатики

Направление подготовки 01.04.02 – Прикладная математика и информатика

Направленность (профиль) Технологии параллельных и распределённых вычислений

УТВЕРЖДАЮ:

Зав.кафедрой/РОП

\_\_\_\_\_\_\_\_\_\_\_\_ Дорофеева В.И.

«26» октября 2021г.

**ЗАДАНИЕ**

на выполнение выпускной квалификационной работы

студента Тютюнова Игоря Андреевича 01ПМ-М

1. Тема ВКР **Разработка методов оптимизации взаимодействия веб-сайта с базой данных**

Утверждена приказом по университету от «26» октября 2021 года № 2-2937

2. Срок сдачи студентом законченной работы «7» июля 2022 года

3. Исходные данные к работе: учебные пособия и курсы лекций по дискретной математике, прикладной комбинаторике, теории графов, аспектам разработки программного обеспечения, электронные источники и ссылки по программированию, алгоритмизации, разработке приложений.

4. Содержание ВКР

5. Перечень графического материала

рисунков – 52, таблиц – 14

6. Консультанты по ВКР (с указанием относящихся к ним разделов)

|  |  |  |  |
| --- | --- | --- | --- |
| Раздел | Консультант | Подпись, дата | |
| Задание выдал | Задание принял |
| \_ | \_ | \_ | \_ |
| \_ | \_ | \_ | \_ |

Дата выдачи задания«26» октября 2021 г.

Научный руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Федяев Ю.С.

Задание принял к исполнению \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тютюнов И.А.

КАЛЕНДАРНЫЙ ПЛАН

|  |  |  |
| --- | --- | --- |
| Наименование этапов  ВКР | Срок выполнения  этапов работы | Примечание |
| Подбор и анализ источников и научных изданий в соответствии с темой исследования | ноябрь-декабрь 2021 | Выполнено |
| Написание введения | май 2022 г. | Выполнено |
| Написание главы 1 | январь-февраль 2022 г. | Выполнено |
| Написание главы 2 | март-апрель 2022 г. | Выполнено |
| Написание главы 3 | март-апрель 2022 г. | Выполнено |
| Написание заключения | май 2022 г. | Выполнено |
| Оформление ВКР | май 2022 г. | Выполнено |
| Сдача ВКР | июль 2022 г. | Выполнено |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Тютюнов И.А.

Научный руководитель ВКР \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Федяев Ю.С.

**Содержание**

# ВВЕДЕНИЕ

**Обоснование выбора темы и ее актуальность**

В современном мире трудно переоценить значимость баз данных. Базы данных прочно укрепили свои позиции повсеместно в промышленных, образовательных, здравоохранительных, правоохранительных и в других общественно важных структурах, а также в сферах бизнеса. Также, базы данных активно используются в веб-сайтах и от оптимальности взаимодействия веб-сайтов с базой данных зависит скорость работы первых. В свою очередь, скорость работы веб-сайтов влияет на комфорт их использования пользователями.

Высокую нагрузку на базу данных вызывают операции создания, чтения, изменения и удаления данных. Эти операции можно объединить одной аббревиатурой — CRUD операции (Create, Read, Update, Delete).

Высокая нагрузка на базу данных, которая может возникать при выполнении CRUD-операции, увеличивает нагрузку на веб-сервер, что отрицательным образом сказывается на скорости работы веб-сайта в целом. Высокая нагрузка на веб-сервер потребует дополнительных затрат на его поддержку. А низкая скорость работы веб-сайта отрицательным образом скажется на комфорте использования сайта пользователями. Пользователей, в свою очередь, не будет устраивать скорость работы веб-сайта, в следствии чего они будут выбирать более производительные сайты конкурентов.

Таким образом, проблема оптимизации взаимодействия веб-сайта с базой данных очень актуальна. Особенно сильно это проблема актуально для веб-сайтов, занимающихся коммерческой деятельностью, например, интернет магазины. Ведь уровень комфорта использования интернет-магазинов напрямую влияет на количество активных пользователей, а соответственно, от этого зависит прибыль. Для решения проблемы оптимизации взаимодействия веб-сайта с базой данных, в рамках выпускной квалификационной работы, будут разработаны веб-библиотеки, реализующие методы оптимизации взаимодействия веб-сайта с базой данных.

**Предмет исследования**

Оптимизация работы веб-сайта.

**Объект исследования**

Оптимизация взаимодействия веб-сайта с базой данных.

**Цель работы**

Разработать веб-библиотеки, реализующие методы оптимизации взаимодействия веб-сайта с базой данных.

**Основные задачи исследования**

1. Изучить операции взаимодействия с базой данных;
2. Изучить методы оптимизации взаимодействия с базой данных;
3. Разработать веб-библиотеки, реализующие методы оптимизации взаимодействия веб-сайта с базой данных.

**Структура работы**

Работа состоит из введения, трёх глав, заключения и списка источников.

Во введении рассматривается актуальность работы, ставится цель и формулируются задачи, необходимые для достижения поставленной цели.

Первая глава представляет собой описание операций взаимодействия с базой данных.

Во второй главе описываются методы оптимизации взаимодействия с базой данных.

В третьей главе описывается разработка веб-библиотек реализующих методы оптимизации веб-сайта с базой данных.

В заключении делаются выводы по проделанной работе.

В конце работы приводится список использованных источников

# ГЛАВА 1. ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ ОБ ОПТИМИЗАЦИИ РАБОТЫ С БАЗОЙ ДАННЫХ

## Описание реляционных баз данных

Реляционные базы данных приобрели свою популярность в связи с реализацией в системах управления реляционных моделей, что является очень

удобным при работе с данными. Само понятие СУБД разработал англичанин Эдгара Кодд. Реляционные модели управления характеризуются своей простотой, удобной табличной форме, а также применением формальной математики и реляционных вычислений для обрабатываемых данных [1].

Хранение данных, в реляционных базах, имеют вид таблиц, которые состоят из строк и столбцов. Поля таблицы имеют собственное наименование. Столбцы таблицы могут иметь данные скалярного вида, такие как даты, числа и др.

Отношение таблиц, в реляционной базе данных, представляются как один к одному, многие ко многим или один ко многим.

Строки записей могут представляться в неограниченном количестве, но каждая запись отвечает за свое запись строк записей в таблице неограниченно, и каждая запись соответствует своей сути [2].

Данные, в реляционных моделях, представляют собой двумерный массив и характеризуются следующими особенностями:

1. Любая составляющая таблицы является одной составляющей данных;
2. Любой столбец имеет свое уникальное имя;
3. Отсутствие одинаковых строк в таблице;
4. Все составляющие в столбцах имеют однородный тип;
5. Строки и столбцы имеют произвольный порядок.

В качестве самых распространённых реляционных СУБД, можно выделить:

1. MS SQL Server;
2. SQLite;
3. MySQL;
4. PostgreSQL.

## Описание операций взаимодействия с базой данных

CRUD — акроним, который обозначает четыре базовые операции, используемые при взаимодействии с базой данных:

1. Создание (англ. create),
2. Чтение (англ. read),
3. Модификация (англ. update),
4. Удаление (англ. delete).

Акроним CRUD введён Джеймсом Мартином (англ. James Martin) в 1983 году как классификация функций по манипуляции данными [3].

CRUD функциям соответствуют следующие реализации:

1. «Insert» — вставка новой строки;
2. «Select» — выполняет функцию выборки данных;
3. «Update» — позволяет редактировать записи и последняя операция;
4. «Delete» — позволяет удалять записи.

В системах, реализующих доступ к базе данных с помощью архитектурного стиль взаимодействия REST-API, CRUD функции реализуются через типы запросов POST, GET, PUT и DELETE, соответственно [4].

Рассмотрим подробнее процесс создания новой записи. Функция «Create» предназначена для добавления новых строк в таблицу. Это можно сделать с помощью команды INSERT INTO — ключевого слова, за которым следует название таблицы. Далее указываются имена столбцов и значения, которые нужно вставить (листинг 1). Функция «INSERT INTO» добавит новые строки в таблицу, и каждой созданной записи будет присвоен свой уникальный идентификатор.

INSERT INTO table\_name

VALUES (value1, value2, value3, ...);

INSERT INTO table\_name (column1, column2, column3, ...)

VALUES (value1, value2, value3, ...);

Листинг 1 – Примеры запросов на добавление записи

Функция «Read», позволяет извлекать определенные записи из базы данных и считывать их значения. Ниже приведен пример такого запроса (листинг 2).

SELECT \* FROM table

Листинг 2 – Пример запроса вывода данных

Этот запрос не внесет никаких изменений в таблицу, а отобразит все существующие записи в этой таблице. Также можно указать критерий поиска записей посредством добавления секции where (листинг 3)

SELECT \* FROM table

WHERE NAME LIKE ‘%a’

Листинг 3 – Пример запроса вывода данных с условием

Обновление «Update» — данная операция позволяет изменять существующих записей в базе данных. При выполнении операции «Update» необходимо определить целевую таблицу и столбцы, которые необходимо обновить (листинг 4). Также необходимо указать условие, по которому будут выбираться строки для обновления. В противном случае будут обновлены все указанные столбы таблицы.

UPDATE table

SET column1 = value1, column2 = value2, ...

WHERE ID = 3;

Листинг 4 – Примеры запроса на обновление

Операция «Delete» используется для удаления записи из таблицы (листинг 5). В SQL реализована возможность удаления, как одной конкретной записи, посредством указания соответствующего, так и удаление всех записей, содержащихся в таблице.

DELETE FROM table

WHERE ID = 3;

Листинг 5 – Пример запроса на удаление

Благодаря функциям чтения, создания, обновления и удаления удается организовать простое и правильное взаимодействие с хранимыми данными. Также позволяя разграничивать доступ пользователей по группам. Для одной группы разрешать только чтение записей, а другой предоставлять доступ к созданию и обновлению, удалению записей. Операции CRUD являются минимально необходимыми как для взаимодействия пользователей с системой, так и для разработчиков системы.

CRUD-операции требовательны к ресурсам веб-сервера, особенно это касается баз данных с большим количеством таблиц и строк в таблицах.

Операция «Read» способна спровоцировать высокую нагрузку на веб-сервер, в случае большого количества запросов к базе данных, например, по посещении страниц веб-сайта пользователями. А операции «Create», «Update» и «Delete» инициируют пересоздание индексов в таблицах баз данных, которые используются для более быстрого выполнения «Read» операций, что также создаёт дополнительную нагрузку на веб-сервер.

Следовательно, крайне важно оптимизировать взаимодействие с базой данных при выполнение CRUD-операций.

# ГЛАВА 2. ТЕОРИТИЧЕСКИЕ СВЕДЕНИЯ ОБ ИНСТРУМЕНТАХ, ИСПОЛЬЗОВАННЫХ ПРИ РАЗРАБОТКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Описание языка программирования PHP

PHP (Hypertext Preprocessor) — это интерпретируемый язык программирования общего назначения с открытым исходным кодом. PHP специально сконструирован для веб-разработок и его код может внедряться непосредственно в HTML [1]. Главная область применения PHP — это написание скриптов, которые выполняются на веб-сервере.

PHP является веб-ориентированным языком программирования с динамической типизацией. Этот язык программирования можно сочетать с HTML кодом. Как правило, программы, написанные на языке программирования РНР, выполняются на веб-сервере, а результат отправляется браузеру в виде HTML разметки. Синтаксис языка программирования PHP схож с синтаксисом языка Си.

Основные возможности языка программирования PHP:

1. Автоматическое извлечение POST- и GET-параметров, а также переменных окружения веб-сервера в предопределённые массивы;
2. Взаимодействие с большим количеством различных систем управления базами данных через дополнительные модули (MySQL, MySQLi, SQLite, PostgreSQL, Oracle Database (OCI8), Microsoft SQL Server, Sybase, ODBC, mSQL, IBM DB2, Cloudscape и Apache Derby, Informix, Ovrimos SQL, Lotus Notes, DB++, DBM, dBase, DBX, FrontBase, FilePro, Ingres II, SESAM, Firebird и InterBase, Paradox File Access, MaxDB, интерфейс PDO, Redis);
3. Автоматизированная отправка HTTP-заголовков;
4. Работа с HTTP-авторизацией;
5. Работа с cookies и сессиями;
6. Работа с локальными и удалёнными файлами, сокетами;
7. Обработка файлов, загружаемых на сервер;
8. Создание и работа с API;
9. Создание приложение с графическим интерфейсом пользователя при использовании Фреймворка Qt Designer;
10. Создание консольных приложений;

## Инструмент хранения данных Radis

Redis (REmote DIctionary Server) — это нереляционное хранилище структур данных в памяти с открытым исходным кодом (под лицензией BSD).

Redis предоставляет такие структуры данных как строки, хэши, списки, наборы, отсортированные наборы с запросами диапазона. С этими структурами данных можно выполнять атомарные операции, такие как добавление к строке, увеличение значения в хэше, добавление элемента в список, вычисление пересечения, объединения и разности множеств, или получение элемента с наивысшим рейтингом в отсортированном наборе [2].

Для достижения максимальной производительности Redis работает с набором данных в оперативной памяти. В зависимости от варианта использования, Redis может периодически сохранять данные на постоянное запоминающие устройство.

Radis может быть установлен на такие операционные системы, как: Linux, Windows, macOS.

Отличие Radis от реляционных СУБД:

1. Данные хранятся в оперативной памяти. Благодаря этому Radis выигрывает в производительности у реляционных СУБД;
2. Отсутствует язык SQL;
3. Данные хранятся не в виде таблиц, а в виде строк, списков, хешей, множеств, в том числе отсортированных.

В роли чего можно использовать Radis:

1. Как хранилище пользовательских сессий;
2. Как брокер сообщений;
3. Как СУБД для небольших приложений, блогов;
4. Для кэширования данных из основного хранилища, что значительно снижает нагрузку на реляционную базу данных. В качестве инструмента для кеширования Radis будет использован при разработке программного обеспечения для кеширования программных алгоритмов в рамках текущей работы;
5. Для хранения «быстрых» данных — когда важны скорость и критичны задержки передачи (аналитика и анализ данных, финансовые и торговые сервисы).

Основные команды для управления данными в Radis:

1. HSET — сохраняет значение по ключу;
2. HGET — получение значения по ключу (для определённого поля);
3. HGETALL — получение всех пар «ключ-значение»;
4. HKEYS и HVALS — получение всех ключей и соответствующих им значений.

С помощью подобных команд можно управлять данными непосредственно из командной строки операционной системы, что используется крайне редко. Зачастую, манипуляции с данными происходят с помощью языков программирования, для который существуют готовые библиотеки, позволяющие взаимодействовать с Radis.

Язык программирования PHP имеет встроенные средства для взаимодействия с Radis и не требует установки сторонних библиотек. В рамках выпускной квалификационной работы Radis использовался при разработке программного обеспечения для кеширования результаты работы программных алгоритмов.

## Описание протокола обмена сообщениями AMQP и брокера сообщений RabbitMQ

AMQP (Advanced Message Queuing Protocol) — открытый протокол для передачи сообщений между компонентами системы. Основная идея состоит в том, что отдельные подсистемы (или независимые приложения) могут обмениваться произвольным образом сообщениями через AMQP-брокер, который осуществляет маршрутизацию, возможно гарантирует доставку, распределение потоков данных, подписку на нужные типы сообщений. [4] Протокол AMQP вводит три понятия:

1. Exchange (обменник или точка обмена) — в неё отправляются сообщения. Обменник распределяет сообщение в одну или несколько очередей. Он маршрутизирует сообщения в очередь на основе созданных связей (binding) между ним и очередью
2. Queue (очередь) — структура данных на диске или в оперативной памяти, которая хранит ссылки на сообщения и отдает копии сообщений consumers (потребителям). Одна очередь может использоваться несколькими потребителями
3. Binding (привязка) — правило, которое сообщает точке обмена в какую из очередей эти сообщения должны попадать. Обменник и очередь могут быть связаны несколькими привязками

Протокол AMQP работает поверх протокола TCP/IP.

RabbitMQ — это реализация AMQP с открытым исходным кодом. Сервер написан на Erlang и поддерживает несколько клиентов, таких как: Python, Ruby, .NET, Java, JMS, C, PHP, ActionScript, XMPP, STOMP. Он используется для хранения и пересылки сообщений в распределенной системе и хорошо работает с точки зрения простоты использования, масштабируемости и высокой доступности. [RabbitMQ](https://www.rabbitmq.com/) маршрутизирует сообщения по всем базовым принципам протокола [AMQP](https://ru.wikipedia.org/wiki/AMQP). Отправитель передает сообщение брокеру, а тот доставляет его получателю. RabbitMQ реализует и дополняет протокол AMQP.

Основная идея модели обмена сообщениями в RabbitMQ заключается в том, что producer (издатель) не отправляет сообщения непосредственно в очередь. На самом деле и довольно часто издатель даже не знает, будет ли сообщение вообще доставлено в какую-либо очередь. Вместо этого издатель может отправлять сообщения только на обмен. С одной стороны, обмен получает сообщения от издателей, а с другой — отправляет их в очереди.[5]

Работу RabbitMQ можно описать следующим образом:

1. Издатель отправляет сообщение определенному обменнику;
2. Обменник, получив сообщение, маршрутизирует его в одну или несколько очередей в соответствии с правилами привязки между ним и очередью;
3. Очередь хранит ссылку на это сообщение. Само сообщение хранится в оперативной памяти или на диске;
4. Как только потребитель готов получить сообщение из очереди, сервер создает копию сообщения по ссылке и отправляет;
5. Потребитель получает сообщение и отправляет брокеру подтверждение;
6. Брокер, получив подтверждение, удаляет копию сообщения из очереди. Затем удаляет из оперативной памяти и с диска;

В рамках выпускной квалификационной работы, брокер сообщений RabbiqMQ использовался при разработке программного обеспечения для обмена сообщениями по протоколу AMQP. Для взаимодействия PHP и RabbitMQ использовалась сторонняя PHP-библиотека php-amqplib/php-amqplib.

# ГЛАВА 3. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

## Постановка задачи

Одни из самых ресурсоёмких операций с базой данных — это CRUD операции со строками таблиц. Под аббревиатурой CRUD скрываются такие операции как:

1. Создания записей (Create);
2. Чтение записей (Read);
3. Изменение записей (Update);
4. Удаление записей (Delete).

CRUD операции соответствуют следующие HTTP-методы:

1. Создать запись — метод POST;
2. Прочитать записать — метод GET;
3. Изменить запись — метод PUT;
4. Удалить запись — метод DELETE.

Зачастую, на веб-сайтах, CRUD-операции инициируют пользователи. Например, при посещении страниц веб-сайта, на которых присутствует динамически-изменяемая информация, хранящаяся в базе данных, пользователи инициируют операцию чтения данных. На крупных веб-сайтах, особенно в интернет-магазинах, где количество товаров исчисляется десятками тысяч и тысячами активных пользователей, операция чтения данных способна вызвать серьёзную нагрузку на сервер, в результате чего скорость загрузки страниц веб-сайта будет на неудовлетворительном уровне, а поддержка веб-сервера будет требовать больших финансовых затрат. Пользователей, в свою очередь, не будет устраивать скорость работы веб-сайта, в следствии чего будет уменьшаться количество активных пользователей, а следовательно, прибыль от работы интернет-магазина будет падать.

Для снижения нагрузки на базу данных, которую создают большое количество запросов на чтение записей, можно воспользоваться кешированием запросов к базе данных. Но простых в использовании веб-библиотек, предоставляющих возможность кешировать запросы к базе данных, найдено не было. Поэтому было решено разработать веб-библиотеку, которая позволит кешировать не только запросы к базе данных, а также и кешировать работу программных алгоритмов. А храниться кеш будет в оперативной памяти сервера, что положительно скажется на скорости сохранения кеша и получения кеша.

Что касается операций создания, изменения, удаления записей. Данные операции также инициируют пользователи веб-сайтов. Подобные операции могут вызываться, например, при отправке каких-либо заявок с сайта, оформление заказа, добавление и удаление товаров из корзины интернет-магазина. Когда тысячи пользователей инициируют подобные операции в один момент времени — это отрицательным образом сказывается на нагрузке как на базу данных, так и в целом на веб-сервер. В следствии чего, как и в случае запросов на чтение записей из базы данных, скорость работы сайта падает, количество активных пользователей уменьшается, финансовые затраты на поддержку веб-сервера увеличиваются.

Для снижения нагрузки на базу данных, которую вызывают операции создания, изменения, удаления записей, подойдёт отложенная обработка этих операций путём добавления их в очередь задач и дальнейшего выполнения задач, находящихся в очереди. Готовых, простых в использовании решений, позволяющих реализовать механизм добавления задач в очередь и механизм чтения задач из очереди, не было найдено. Поэтому, было решено разработать веб-библиотеку, которая позволит добавлять задачи в очередь, забирать задачи из очереди и их исполнять. Также разрабатываемая веб-библиотека будет отличаться высокой масштабируемостью. Под масштабируемостью подразумевается лёгкое создание произвольного количества очередей, а также создание произвольного количества обработчиков очередей. Для одной очереди может быть создано несколько обработчиков, что позволит ускорить обработку задач.

В итоге, была поставлена задача по разработке веб-библиотек:

1. Веб-библиотека для кеширования работы программных алгоритмов;
2. Веб-библиотека для добавления задач в очередь и выполнение этих задач.

Веб-библиотеки должны отличаться простотой в использовании, высокой производительностью и масштабируемостью.

## Разработка программного обеспечения для кеширования работы программных алгоритмов

## Разработка программного обеспечения для обмена сообщениями по протоколу AMQP

## Результат использования разработанного программного обеспечения

# ЗАКЛЮЧЕНИЕ

# СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Мейер М. Теория реляционных баз данных. -М.: Мир, 1987. - 608 с.
2. Васильева К. Н., Хусаинова Г. Я. Реляционные базы данных // Colloquium-journal. 2020. №2 (54).
3. Основы построения баз данных: учебное пособие: / Д. В. Чмыхов, А. С. Сазонова, А. А. Тищенко. — Москва.
4. https://restfulapi.net/http-methods/

# ПРИЛОЖЕНИЕ