Частное учреждение профессионального образования
«Высшая школа предпринимательства»
(ЧУПО «ВШП»)

КУРСОВОЙ ПРОЕКТ

«Разработка базы данных для мессенджера»

Выполнил:
студент 3-го курса специальности
09.02.07 «Информационные системы и программирование»
Грошелев Кирилл Дмитриевич
подпись: _____

Проверил:
преподаватель дисциплины,
преподаватель ЧУПО «ВШП»,
к.ф.н. Ткачев П.С.
оценка: _____
подпись: _____

Содержание

Тверь, 2024 г.

Введение

Актуальность темы

В наше время огромное количество фирм используют персональные компьютеры для сохранения и обработки любого вида информации. Эта информация содержится в базах данных. Базы данных играют важную роль в развивающемся мире технологий. Всё, с чем мы каждый день взаимодействуем в жизни, по всей видимости, зафиксировано в какой-нибудь базе. Работа с базами данных является важнейшим навыком в работе с компьютером, а специалисты данной области становятся всё более востребованными. Главные идеи нынешней информационной методики базируются на представлении, в соответствии чему информация должна быть образована в базы данных с задачей отображения динамически изменяющегося мира и удовлетворения всех потребностей в информации у пользователей. Базы данных формируются и работают под управлением специальных программных средств, называемых системами управления базами данных.

В настоящее время мессенджеры стали неотъемлемой частью нашей повседневной жизни, обеспечивая обмен сообщениями, файлами, аудио- и видеозвонками. Они служат удобным проводником в общении между лицом или гуппой лиц. Для обеспечения стабильной работы и удобного пользовательского интерфейса мессенджеры используют специальные базы данных, которые хранят информацию о пользователях, сообщениях, файлов и прочих данных. Разработка базы данных для мессенджера требует тщательного планирования и оптимизации, учитывая большое количество пользователей, высокую нагрузку на серверы и необходимость обеспечения безопасности и конфиденциальности данных. В данной курсовой работе мы рассмотрим основные принципы разработки базы данных для мессенджера, ее структуру, функциональность и возможные проблемы, с которыми может столкнуться разработчик.

Цель работы

Цель данной курсовой работы заключается в попытке разработать базу данных для мессенджера, обеспечивающую практичное хранение данных для сообщений, пользователей, чатов и т.д

Задачи

Для достижения заданной мне цели, мне нужно выполнить следующие задачи:

- Обозначить требования к функционалу базы данных
- Выбрать инструменты и технологии для разработки
- Спроектировать базу данных на основе требований к функционалу
- Реализовать базу данных на основе спроектированной схемы
- Определить требования к базе данных, обеспечивающие целостность системы

1. Теоритические основы разработки базы данных

1.1. Понятие базы данных и её основные функции

База данных - это организованная структура, которая предназначена для хранения информации. В то время, когда происходило развитие термина баз данных, в них сохранялись исключительно информация, однако уже в наши дни многие системы управления базами данных позволяет размещать в своих структурах и данные, и программный код, с помощью которого совершается связь с пользователями или с другими программно-аппаратными комплексами. При этом данные не должны противоречить друг другу, быть целостными и не избыточными. База данных создается для сохранения и непосредственного доступа к информации, содержащей сведения об искомой предметной области. Прежде всего, целью использования информации из баз данных и сложностью информационных процессов, существующих в пределах предметной области в конкретных условиях. Основными функциями базы данных являются:

- Хранение данных: БД обеспечивает безопасное и надёжное хранение информации.
- Обработка данных: с помощью запросов и операций можно выполнять различные действия с данными, такие как поиск, сортировка, фильтрация и т. д.
- Анализ данных: базы данных позволяют проводить анализ данных, выявлять закономерности и тенденции, а также принимать обоснованные решения на основе полученной информации.

Система управления базами данных - это программный механизм, предназначенный для записи, поиска, сортировки, обработки и печати информации, содержащейся в базе данных. Иными словами, система управления базами данных - механизм, регулирующий работу самой базы данных.[14]

1.2. Типы баз данных

Существует множество типов баз данных, каждый из которых имеет свои особенности и преимущества в зависимости от поставленной задачи и требований проекта. В данной работе будут рассмотрены основные типы баз данных:

- Реляционные базы данных (РБД) — это наиболее распространённый тип баз данных. Они основаны на реляционной модели данных и используют таблицы для хранения информации. Реляционные базы данных имеют

множество преимуществ, таких как простота использования, надёжность и эффективность. Однако они также имеют некоторые недостатки, такие как сложность масштабирования и ограничения в производительности при работе с большими объёмами данных. [10]

- Объектно-ориентированные базы данных (ООБД) — этот тип баз данных основан на объектно-ориентированной модели данных. В ООБД данные представлены в виде объектов, а не таблиц. Это позволяет более точно моделировать сложные предметные области и обеспечивает более высокую гибкость при разработке приложений. Однако ООБД имеют более сложную структуру и требуют более высокой квалификации разработчиков. [2]
- Иерархические базы данных (ИБД) — в иерархических базах данных информация организована в виде древовидной структуры. ИБД используются для моделирования иерархических отношений между объектами, например, в системах управления проектами или в каталогах товаров. Иерархические базы данных просты в использовании и обеспечивают хорошую производительность при работе с небольшими объёмами данных, но они не подходят для сложных запросов и обработки больших объёмов данных. [8]
- Сетевые базы данных (СБД) — сетевые базы данных похожи на иерархические, но в них допускается наличие нескольких родительских узлов у одного дочернего узла. СБД используются для моделирования сложных отношений между объектами и обеспечивают большую гибкость по сравнению с иерархическими базами данных. Однако СБД сложнее в реализации и имеют более низкую производительность по сравнению с реляционными базами данных. [9]
- NoSQL базы данных — NoSQL базы данных представляют собой класс нереляционных баз данных, который включает в себя широкий спектр технологий, разработанных для обеспечения масштабируемости и гибкости. NoSQL базы данных не используют схемы и могут обрабатывать различные типы данных, включая документы, пары ключ-значение и графы. Они часто используются для обработки больших объёмов неструктурированных данных, таких как данные социальных сетей, журналов веб-сервера и т. д. [11]
- Графовые базы данных — графовые базы данных предназначены для работы с данными, представленными в виде графа. Графовые базы данных позволяют эффективно моделировать и обрабатывать отношения между объектами. Они используются в таких областях, как социальные сети, рекомендательные системы и анализ данных.[12]
- RDF (Resource Description Framework) — это стандарт представления данных, который используется для описания ресурсов в интернете. RDF

позволяет описывать ресурсы с помощью троек «субъект-предикат-объект», где субъект и объект являются ресурсами, а предикат описывает отношение между ними. RDF является основой для создания семантических веб-приложений, которые используют машинное понимание данных для предоставления более интеллектуальных услуг.[13]

1.3. Инструменты для разработки базы данных

Разработка баз данных включает в себя проектирование, создание, управление и оптимизацию баз данных. Для этих целей существует множество инструментов, каждый из которых предлагает определенные функции и возможности. В этом разделе будут рассмотрены основные инструменты для разработки баз данных.

Инструменты для разработки баз данных делятся на следующие категории:

- **Системы управления базами данных (СУБД)** - это программное обеспечение, предназначенное для создания, управления и модификации баз данных.
  Например, MySQL - распространенная реалиационная СУБД с открытым исходным кодом.

- **Инструменты моделирования данных:** помогают разработчикам создавать и визуализировать структуры баз данных.
  Например, Microsoft Visio - универсальный инструмент для создания диаграмм, включая диаграммы сущность-связь (ERD).

- **Инструменты управления базами данных:** предоставляют интерфейсы для администрирования и управления базами данных.
  Например, MySQL Workbench - инструмент для управления СУБД MySQL.

- **Инструменты разработки и интеграции:** облегчают процесс написания, тестирования и отладки SQL-кода и приложений, работающих с базами данных.
  Например, DBeaver - многофункциональный инструмент для работы с различными СУБД, поддерживающий множество форматов данных.

- **Инструменты мониторинга и оптимизации:** эти инструменты помогают отслеживать производительность баз данных и оптимизировать запросы.
  Например, Oracle AWR - инструмент для анализа производительности баз данных Oracle.

Для разработки следует в лучшем случае выбрать инструмент из каждой категории, изходя из используемой СУБД и из личных предпочтений. Использование данных инструментов оптимизирует работу разработчика и сделает ее намного проще и качественнее.

1.4. Нормализация баз данных

Нормализация — это процесс организации структуры базы данных таким образом, чтобы минимизировать избыточность данных и обеспечить целостность информации. Нормализация позволяет улучшить производительность базы данных, упростить её обслуживание и обеспечить более эффективное использование ресурсов.

Существует несколько уровней нормализации, каждый из которых устраняет определённые виды избыточности. Чем выше уровень нормализации, тем больше таблиц требуется для представления данных, но при этом повышается целостность данных и уменьшается вероятность возникновения аномалий обновления. Уровни нормализации существуют следуюшие:

- **Первый уровень нормализации**, или 1НФ, является базовым уровнем нормализации реляционных баз данных. Он требует, чтобы все атрибуты были атомарными, то есть не содержали в себе других значений. Это означает, что каждый атрибут должен представлять собой одно значение, а не набор значений или список.
- **Второй уровень нормализации**, или 2НФ, является следующим шагом после достижения 1НФ. Он требует выполнения 1НФ и отсутствия частичной зависимости неключевых атрибутов от первичного ключа. Частичная зависимость возникает, когда неключевой атрибут зависит только от части первичного ключа. Для устранения частичной зависимости необходимо разбить таблицу на две или более таблиц.

- **Третий уровень нормализации**, или 3НФ, является самым строгим уровнем нормализации реляционных баз данных. Он требует выполнения 2НФ и отсутствия транзитивной зависимости неключевых атрибутов от первичного ключа. Транзитивная зависимость возникает, когда неключевой атрибут зависит от другого неключевого атрибута, который, в свою очередь, зависит от первичного ключа.

- **Нормальная форма Бойса-Кодда (BCNF)** — для любой функциональной зависимости $X \rightarrow A$ в отношении R атрибут A должен функционально

зависеть от каждого элемента в X (то есть X должен быть суперключом отношения R).

- **Четвёртый уровень (4NF)** — отношение находится в 4NF, если оно находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от её потенциальных ключей.

- **Пятый уровень или нормальная форма проекции-соединения (5NF или PJ/NF)** — любая зависимость соединения в ней определяется потенциальными ключами отношения (ключами исходного отношения). Также отношение R соответствует 5NF тогда и только тогда, когда каждая нетривиальная зависимость соединения в R следует из существования некоторого потенциального ключа K для R.

- **Шестой уровень нормализации (6NF)** — также известен как Domain/Key Normal Form (DKNF) или идеальная нормальная форма. Это самый высокий уровень нормализации, который требует, чтобы каждый атрибут отношения зависел от первичного ключа этого отношения и не зависел функционально от любого другого атрибута. Другими словами, отношение находится в DKNF, когда оно уже находится в BCNF и каждый домен атрибутов полностью зависит от первичного ключа. Важно отметить, что шестой уровень нормализации редко используется на практике, так как он может привести к чрезмерной декомпозиции базы данных и усложнить её понимание и использование. Кроме того, многие системы управления базами данных не поддерживают этот уровень нормализации.

1.5. Связи между таблицами в базах данных

В базах данных (БД) связи между таблицами играют ключевую роль в обеспечении целостности данных, а также в упрощении процесса обработки информации. Связь между двумя таблицами представляет собой зависимость между данными, которые хранятся в этих таблицах. Существует несколько типов связей, каждый из которых имеет свои особенности и преимущества:

- **Один к одному**. В этом случае каждой записи в одной таблице соответствует только одна запись в другой таблице. Этот тип связи используется редко, так как он не позволяет эффективно использовать данные.

- **Один ко многим**. Это наиболее распространённый тип связи, который используется для моделирования отношений «один ко многим». Например, один клиент может иметь несколько заказов.
- **Многие ко многим**. Этот тип связи требует использования третьей таблицы, которая будет служить связующим звеном между двумя другими таблицами. Например, многие студенты могут изучать несколько предметов, и каждый предмет может изучаться многими студентами.
- **Самосвязь**. Этот тип связи не требует использования отдельных таблиц для реализации. При таком типе связи внешний ключ определенной таблицы ссылается на запись с этой же таблицы. Например, сущность пользователя хранит в себе внешний ключ сущности другого пригласившего на сайт пользователя,

Стоит отметить, что связи используются только в реалиационных БД. Другие типы БД не поддерживают данный функционал.

1.6. Безопасность баз данных

Безопасность базы данных (БД) — это совокупность мер и стратегий, направленных на защиту данных от несанкционированного доступа, утечек, повреждений и других угроз. В современных условиях обеспечения безопасности информации защита баз данных становится критически важной задачей.

Основные аспекты безопасности базы данных:

- **Аутентификация** — это процесс проверки подлинности пользователя или системы, пытающихся получить доступ к базе данных.
- **Авторизация** — это процесс определения прав и привилегий пользователя после аутентификации.
- **Ролевое управление доступом (RBAC):** Предоставляет доступ к данным на основе ролей, назначенных пользователям.
- **Политики доступа (ACL):** детализируют права доступа на уровне таблиц, строк или столбцов.
- **Шифрование:** защищает данные, делая их нечитаемыми для неавторизованных пользователей. Возможен вариант как шифровки данных, хранящиеся на диске, с использованием алгоритмов AES или RSA, так и шифровки данных, передаваемых между клиентом и сервером, с использованием SSL/TLS.

- **Управление уязвимостями:** регулярное обновление и исправление программного обеспечения базы данных для устранения известных уязвимостей. Важно своевременно устанавливать обновления безопасности.
- **Мониторинг и аудит:** Использование систем обнаружения вторжений (IDS) и ведение журналов аудита для выявления и анализа подозрительной активности.
- **Бэкапы и восстановление данных:** резервное копирование данных помогает восстановить базу данных в случае утраты или повреждения данных.
- **Управление сеансами и время жизни паролей:** контроль сеансов для отключения нежелательных пользователей.
- **Контроль доступа:** ограничение физического и логического доступа к серверам баз данных.

Применение этих мер позволяет значительно уменьшить риски утечки данных и обеспечить защиту информации в базе данных.

1.7 Проектирование схемы базы данных

Проектирование схемы базы данных представляет собой процесс создания структуры базы данных, включающий определение таблиц, столбцов, связей, индексов, ограничений и т.д.

Проектирование схемы базы данных включает следующие шаги:

- **Определение требований к базе данных:** анализ требований, определение целей и задач, которые должна решать база данных. Изучение потребностей пользователей и процессов, которые могут быть автоматизированы с помощью базы данных.

- **Определение сущностей и атрибутов:** создание таблиц (сущностей), создание необходимых по требованиям полей, в которых будет храниться нужная информация.

- **Выбор СУБД:** выбор подходящей СУБД для проектирования на основе выбранного типа базы данных.

- **Определение связей между сущностями:** Определение первичных и внешних ключей для обеспечения целостности и согласованности данных в базе данных.

- **Создание диаграммы (ERD):** визуально отображает сущности и их связи. Это помогает лучше понять структуру данных и связи между ними.

- **Нормализация данных: о**рганизации данных в таблицы для уменьшения избыточности и обеспечения целостности данных

- **Определение первичных и внешних ключей:** Первичные ключи уникально идентифицируют записи в таблице. Внешние ключи устанавливают связи между таблицами и обеспечивают целостность данных. Тем самым, данными методами мы обеспечиваем связи межу таблицами.

- **Создание физической схемы базы данных:** реализация логической схемы в конкретной СУБД. Она включает в себя создание таблиц, индексов, представлений и других объектов базы данных.

При проектировании схемы базы данных необходимо учитывать требования к производительности, безопасности и надежности. Важно обеспечить, чтобы база данных была масштабируемой и могла обрабатывать увеличивающиеся объемы данных без потери производительности.

1.8. Заключение первой главы

В заключение первой главы можно сделать вывод о важности теоретических основ разработки базы данных и её значении для эффективного управления и обслуживания клиентов.
Основные принципы и рекомендации по разработке базы данных включают:

- Анализ по требованиям к функционалу базы данных.
- Выбор типа базы данных
- Выбор инструментов для разработки базы данных по техническим требованиям и личным предпочтениям.
- Определние нормальной формы базы данных
- Проектирование схемы базы данных с учетом требований к производительности, безопасности и надежности.

Эти принципы и рекомендации являются основой для успешной разработки и внедрения базы данных, которая будет эффективно поддерживать деятельность бара "Кооператив" и обеспечивать высокое качество обслуживания клиентов.

2. Разработка базы данных для мессенджера

2.1. Анализ требований

Проектирование базы данных для мессенджера требует тщательного анализа функциональных и нефункциональных требований системы. В данном параграфе я попытаюсь изложить требования к структуре базы данных мессенджера, которые охватывают ключевые аспекты хранения данных, обеспечения их целостности и производительности.

Пользователи:

Для начала, нужно спроектировать сущность пользователя для реализации основного функционала, такого как авторизация, регистрация, отправка сообщений, удаление сообщений и т.д. Данная сущность должна быть связана с другими сущностями для идентификации действий в мессенджере, а так же для возможность взаимодействия пользователя с другими пользователями.

Требования, которые я выделил для проектирования данной сущности:

- **Уникальный идентификатор:** для идентификации сущности пользователя системой и другими сущностями требуется реализовать первичный ключ.
- **Основная информация:** для лучшего распознания пользователя другими пользователями следует добавать сущности пользователя возможность добавлять информацию о себе: имя, фамилия и никнейм.
- **Пользователи должны быть уникальными**: для индентификации и поиска пользователей в чатах другими пользователями требуется сделать пользователей уникальными, а точнее их имена.
- **Сущность пользователя должна включать в себя захешированный пароль:** требование обусловлено тем, что для авторизации и регистрации пользователя по паролю требуется хранить зашифрованные данные пароля, в ином случае авторизация и регистрация были бы не возможны.
- **Дата регистрации:** данное требование будет нужно для реализации административной панели для удобной фильтрации пользователей. Значение аттрибута должно автоматически проставляться на текущую дату.
- **Дата последней смены пароля:** нужна для уведомления пользователя о рекомендации по смене пароля, если пользователь его давно не менял.

- **Хранение фотографий пользователя:** для упрощенного распознования пользователя другими участниками чата следует добавить возможность добавлять изображения в профили пользователей.

Чаты:

Для хранения списка чатов пользователя требуется реализовать сущность чата, которая будет хранить в себе сообщений и участников чата (пользователей).

Требования, которые я выделил для проектирования данной сущности:

- **Уникальный идентификатор:** для работы с данными сущности чата на сервере требуется создать уникальный идентификатор для распознания сервером, какой чат нужно изменить или удалить.
- **Хранение данных об участниках:** требование нужно для того, чтобы пользователи молги видеть список чатов, в которых они присутствуют, а так же для просмотра участников чата пользователем, приглашения и удаление пользователей из чата.
- **Профильная часть:** у каждого чата должно быть свое название и изображение для простого восприятия пользователем.
- **Дата создания чата:** требование нужно для дальнейшей реализации фильтрации чатов по дате создания в административной панели. Значение должно самостоятельно проставляться на текущую дату.

Сообщения:

Для хранения, отображения, отправки и удаления сообщений пользователям следует реализовать сущность сообщений. Без данной сущности обмен сообщениями невозможен. Сущность сообщения должна быть связана с пользователем, чтобы идентифицировать отправителя сообщения.

Требования, которые я выделил для реализации данной сущности:

- **Сообщение должно быть привязано к отправителю (пользователю):** у каждого сообщения должен быть отправитель, чтобы пользователи могли знать, кто отправил сообщение в чат.
- **Длина сообщения должна быть не более 255 символов:** данное требование нужно для экономии ресурсов БД.
- **Сообщение должно быть привязано к чату, в котором оно было отправлено:** требование нужно для того, чтобы сообщения хранились в определенных чатах.
- **Разновидность типов сообщения:** требование обусловлено тем, что сообщение должно отправляться как пользователем, так и сервером для

уведомления участников чата о каком-либо событии, например, о вступлении нового участиника в чат.

- **Возможность добавлять вложения:** помимо текста, сообщение должно иметь возможность содержать в себе вложения по типу картинок, видел, аудио и т.д
- **Дата создания:** требование нужно для того, чтобы пользователи могли видеть когда было отправлено сообщение, а также для добавления возможность фильтрации сообщений по дате создания как для пользователей, так и для административной панели.

Вложения:
Для хранения медиа-файлов, таких как изображения, видео, фото, аудио и т.п. следует реализовать сущность вложения, которая будет хранить в себе путь к файлу на жестком диске к медиа-файлу.
Требования, которые я выделил для реализации данной сущности:

- **Путь к медиа-файлу на жестком диске:** требование нужно для подгрузки клиентом медиа-файла для дальнейшего его отображения пользователю.
- **Отправитель:** у каждого вложения должен быть свой отправитель, поэтому сущность вложения нужно свзять с сущностью пользователя, чтобы узнавать отправителя вложения.
- **Тип вложения:** для более удобного распознавания клиентом типа вложения (изображение, или видео, или аудио) требуется добавить информацию о типе вложения в сущность вложения.

2.2. Типовые запросы

Типовые запросы в SQL — это стандартные команды для выполнения часто используемых операций с базой данных, таких как выборка (SELECT), вставка (INSERT), обновление (UPDATE) и удаление (DELETE) данных. Они облегчают управление и манипуляцию данными.

Запрос для вывода всех сообщений, отправленных пользователем "Sunday”:

```
SELECT * FROM `messages`
JOIN `users` ON `users`.`id` =
`messages`.`from_user`
WHERE `users`.`username` = 'Sunday'
```

Запрос для вывода всех чатов пользователя “Sunday”:

```
SELECT `chats`.* FROM `chats`
```

```
JOIN `chats_users` ON
`chats_users`.`chat` = `chats`.`id`
JOIN `users` ON `users`.`id` =
`chats_users`.`user`
WHERE `users`.`username` = 'Sunday';
```

Запрос для вывод всех сообщений, отправленных пользователем “Sunday” в чат “Тестовый чат”:

```
SELECT `messages`.* FROM `messages`
JOIN `users` ON `users`.`id` =
`messages`.`from_user`
JOIN `chats` ON `chats`.`id` =
`messages`.`chat`
WHERE (
   `users`.`id` !=
`messages`.`from_user`
   AND
   `chats`.`name` = 'Тестовый чат'
);
```

Запрос для вывода списка всех вложений, отправленных пользователем “Sunday”:

```
SELECT `media`.*, `users`.`username` FR
`media`
JOIN `attachments` ON `attachments`.`me
= `media`.`id`
JOIN `users_photos` ON
`users_photos`.`media` = `media`.`id`
JOIN `users` ON (
  `users`.`id` = `attachments`.`from_user`
  OR `users`.`id` = `users_photos`.`user`
)
WHERE `users`.`username` = 'Sunday';
```

Запрос для вывода всех участников чата с названием “Тестовый чат”:

```
SELECT `users`.* FROM `users`
JOIN `chats_users` ON
`chats_users`.`user` = `users`.`id`
JOIN `chats` ON `chats`.`id` =
`chats_users`.`chat`
WHERE `chats`.`name` = 'Первый чат';
```

2.3. Хранимые процедуры

Хранимая процедура — объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. У них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. [1]

В своей базе данных я предусмотрел 2 хранимых процедуры для отправки и удаления сообщений. Для отправки сообщения в чат я реализовал данную процедуру: [См. Приложение 2]. Для удаления сообщения я реализовал следующую процедуру: [См. Приложение 3]

2.4. Триггеры

Триггеры в MySQL представляют собой специальные объекты базы данных, которые автоматически выполняются при определенных событиях или действиях с данными. Они позволяют программистам определить пользовательские действия, которые должны быть выполнены перед или после выполнения операции в базе данных. Иными словами, это определяемая пользователем SQL-команда, которая автоматически вызывается во время операций INSERT, DELETE или UPDATE. [1]
В своей базе данных я буду использовать тригеры для автоматического выставления при создании записей значений, связанных с датами и временем, для того, чтобы не указывать, например, дату создания записи каждый раз вручную.

Пользователи:
Сущность пользователя содержит в себе данные о дате регистрации и дате смены пароля. Для того, чтобы упростить запрос добавления записи, следует реализовать триггер, который будет автоматически устанавливать на каждую новую запись перед добавлением текущую дату в поле даты регистрации и поле даты смены пароля.

```
CREATE TRIGGER
`users_registration_timestamp`
BEFORE INSERT ON `users`
FOR EACH ROW
SET
   NEW.`date_joined` = NOW(),
   NEW.`date_password_changed`
NOW(),
;
```

Сообщения:
Сущность сообщения также содержит в себе данные о дате отправки сообщения, которая и так должна являться датой на момент отправки сообщения. Для того, чтобы упросить процесс добавления данных о сообщении, требуется создать триггер, который будет автоматически перед добавлением записи устанавливать текущие время в поле даты отправки.

```
CREATE TRIGGER
`messages_creation_timestamp`
BEFORE INSERT ON `messages`
```

```
FOR EACH ROW
SET NEW.`date_created` = NOW()
```

Чаты:

Сущность чата так же хранит в себе поле с датой создания чата. Для того, чтобы автоматизировать процесс установки даты создания чата, требуется создать триггер, который так же будет устанавливаться дату создания чата сам перед добавлением новой записи.

```
CREATE TRIGGER
`chats_creation_timestamp`
BEFORE INSERT ON `chats`
FOR EACH ROW
SET NEW.`date_created` = NOW
```

2.5. Представления

Представления (Views) в MySQL — это виртуальные таблицы, которые основаны на результатах выполнения запроса SELECT. Они представляют собой логические структуры данных, которые могут быть использованы для упрощения и оптимизации запросов к базе данных.[1]

В моей базе данных я реализовал только 1 представление для вывода всех пользователей, не отправивших ни одного сообщения за последние 7 дней.

```
CREATE VIEW afkUsers AS
SELECT * FROM `users`
JOIN `messages` ON `messages`.`from_user`
`users`.`id`
WHERE `messages`.`date_created` >= DATE(N
- INTERVAL 7 DAY);
```

Данное представление выводит сущности пользователей, которые давно не отправляли сообщения в чаты (за последние 7 дней). Представление нужно для того, чтобы оперировать надо пользователями, давно не проявляющих активность в чате.

2.6. Пользовательские функции

Пользовательские функции (User-Defined Functions, UDF) в MySQL — это функции, создаваемые пользователями для выполнения специфических задач, которые не могут быть легко выполнены с помощью встроенных функций MySQL. UDF могут принимать параметры и возвращать скалярные значения (числа, строки, даты и т. д.). Они полезны для создания повторно используемых логических блоков, которые можно вызывать в SQL-запросах.
Для данной базы данных я реализовал одну пользовательскую функцию для отображения всех сообщений определенного чата.

```
DELIMITER //
CREATE FUNCTION
getChatMessagesCount(chatId INT)
RETURNS int
READS SQL DATA
BEGIN
   RETURN (
       SELECT COUNT(*) FROM
`messages`
       WHERE `messages`.`chat` =
chatId
   );
END //
DELIMITER ;
```

2.6. Роли

Для обеспечения безопасности в базе данных, необходимо создать роли с определёнными правами и привилегиями.

В моей базе данных я реализовал только одну роль для сервера, который будет оперировать данными по RestfulAPi. Данная роль ограничивает пользователя в удалении пользователей, обновлении медиа-файлов, обновлении вложений и обновлении сообщений для того в целях безопасности от несанционированного доступа, и во избежание некоторых конфликтов.

[См. приложение 1]

2.7. Выбор инструментов и технологий для разработки базы данных

Для проектикрования, разработки и визуализации базы данных требуется выбрать определенные инструменты, без которых реализация базы данных невозможна, или которые значительно упростят и ускорят разработку.

Система управления базами данных (СУБД)

Так как я выбрал релиационный тип базы данных для своего проекта, я буду использовать СУБД MySQL. MySQL — это одна из самых популярных систем управления реляционными базами данных (СУБД) с открытым исходным кодом. Она была разработана и поддерживается компанией Oracle Corporation. MySQL используется в широком спектре приложений, включая веб-сайты, корпоративные системы и платформы электронной коммерции, благодаря своей надежности, производительности и простоте использования [4]. Именно на данной СУБД будут реализованы все. Ниже приведу отличительные черты MySQL, как положительные, так и отрицательные.

Достоинства MySQL:

- Открытый исходный код
- Высокая производительность
- Масштабируемость
- Широкая поддержка различных платформ
- Широкий спектров поддерживаемых языков программирования
- Поддержка транзакций
- Обширная документация

Недостатки MySQL**:**

- Ограниченная поддержка сложных запросов
- Ограниченная поддержка JSON и NoSQL
- Меньшая поддержка стандартов SQL

Инструмент для моделирования и работы с базой данных

В качестве инструмента для моделирования базы данных, а также для работы с ее данными я выбрал универсальный инструмент MySQL Workbench. MySQL Workbench — это интегрированная среда разработки (IDE) для работы с базами данных MySQL. Она предоставляет инструменты для проектирования схем баз данных, написания и выполнения SQL-запросов, администрирования серверов MySQL и создания отчетов. MySQL Workbench является официальным продуктом Oracle Corporation и доступна для различных операционных систем, включая Windows, macOS и Linux. У данного инструмента есть как и свои достоинства, так и свои недостатки.

Достоинства MySQL Workbench:

- Интуитивно понятный интерфейс
- Возможность проектирования схем данных
- Возможность администрирования серверов MySQL
- Возможность написания и выполнение SQL-запросов

- Возможность миграции данных
- Возможность генерации отчетов
- Поддержка различных операционных систем

Недостатки MySQL Workbench:

- Требовательность к ресурсам
- Проблемы с производительностью
- Ограниченные возможности для работы с большими данными
- Сложность при выполнении сложных операций
- Переодические ошибки и сбои
- Ограниченная поддержка командной строки

MySQL Workbench — это мощный и удобный инструмент для работы с базами данных MySQL, предоставляющий широкий спектр возможностей для проектирования, администрирования и анализа данных. Однако, его использование может быть ограничено из-за требовательности к ресурсам и некоторых проблем с производительностью. Несмотря на эти недостатки, MySQL Workbench остается популярным выбором среди разработчиков и администраторов баз данных благодаря своей функциональности и удобству использования. Именно поэтому мой выбор пал на данный инструмент.

2.8. Проектирование базы данных

По моим требованиям, описаным ранее, следует реализовать сущности, связанные между собой первичными и внешними ключами.

Типы медиа-файлов:

- **video -** видео-вложение
- **audio -** аудио-вложение
- **photo -** изображение
- **document -** любой другой документ

Типы сообщений:

- **from_user** - сообщение пришло от пользователя
- **from_server -** сообщение пришло от сервера (уведомление)

Сущность пользователя - таблица “users”:

- **id -** первичный ключ, содержит целое автоматически создаваемое число, идентификатор пользователя.
- **username -** строка обязательная для заполнения, содержит уникальный никнейм пользователя.
- **first_name -** строка обязательная для заполнения, содержит имя пользователя
- **last_name** - строка обязательная для заполнения, соджержит фамилию пользователя.
- **password** - строка обязательная для заполнения, содержит зашифрованый пароль пользователя.
- **date_joined** - дата и время обязательные для заполнения, поле содержит дату и время регистрации пользователя в системе.
- **date_password_changed** - дата и время обязательные для заполнения, поле содержит дату и время последней смены пароля пользователем.

Сущность медиа-файла - таблица “media”:

- **id -** первичный ключ, содержит целое автоматически созданное число, содержит индентификатор медиа-файла.
- **media_url –** строка обязательная для заполнения, содержит путь к медиа-файлу на локальном хранилище.
- **type** - строка обязательная для заполнения, содержит один из типов медиа-файлов.

Сущность вложения - таблица “attachments”:

- **id -** первичный ключ, содержит целое автоматически созданное число, содержит индентификатор вложения.
- **media** - внешний ключ к сущности медиа-файла (media.id), содержит уникальный идентификатор медиа-файла.
- **from_user** - внешний ключ к сущности пользователя (users.id), содержит уникальный индентификатор пользователя.

Сущность чата - таблица “chats”:

- **id -** первичный ключ, содержит целое автоматически созданное число, идентификатор чата.
- **name** - строка обязательная для заполнения, содержит название чата.
- **date_created** - дата и время обязательные для заполнения, поле содержит дату и время создания чата.

Сущность сообщения - таблица “messages”:

- **id** - первичный ключ, содержит целое автоматически созданное число, идентификатор сообщения.
- **from_user** - внешний ключ к сущности пользователя (users.id), содержит идентификатор пользователя.
- **chat** - внешний ключ к сущности чата (chats.id), содержит индентификатор чата.
- **message** - строка обязательная для заполнения с ограничением в 256 символов, содержит текст сообщения.
- **date_created** - дата и время обязательные для заполнения, поле содержит дату и время создания и отправки сообщения.

Сущность фотографии профиля пользователя - таблица “users_photos”:

- **id** - первичный ключ, содержит целое автоматически созданное число, идентификатор фотографии профиля пользователя.
- **user** - внешний ключ к сущности пользователя (users.id), содержит уникальный идентификатор пользователя.
- **media** - внешний ключ к сущности медиа-файла (media.id), содержит уникальный идентификатор медиа-вложения.

Связь многие ко многим пользователей и чатов - таблица “chats_users”:

- **user** - внешний ключ к сущности пользователя, содержит уникальный идентификатор пользователя.
- **chat** - внешний ключ к сущности чата, содержит уникальный идентификатор чата.

Связь многие ко многим сообщений и вложений - таблица “messages_attachments”:

- **message** - внешний ключ к сущности сообщения (messages.id), содержит уникальный идентификатор сообщения.
- **attachment** - внешний ключ к сущности вложения (attachments.id), содержит уникальный идентификатор вложения.

2.9. Реализация базы данных

После проектирования базы данных следует ее реализовать по заданным ранее условиями. Для воссоздания сущностей я использовал ERD-диаграмму, в которой определил все нужные поля с нужными типами, все связи между

сущностями - первичные и вторичные ключи, индексы на идентификаторы каждой сущности. Для создания ERD-диаграммы я использовал, как я писал ранее, IDE MySQL Workbench.

Шаги по реализации ERD-диаграммы через MySQL Workbench:

- Открыть программу MySQLWorkbench.
- Открыть раздел "Моделирование" (Modeling) в MySQL Workbench и выбрать "Новый модуль ER" (New EER Model).
- Создать необходимые по требованиям сущности с требуемыми аттрибутами
- Добавить необходимые связи между сущностями.
- Обозначить индексы и ограничения на необходимые аттрибуты.
- Сгенерировать SQL-скрипт для создания таблиц на основе EDR, используя опцию "Database" -> "Forward Engineer..." в меню.
- Выполнить SQL-скрипт в редакторе MySQL Workbench

Таким образом, все нужные сущности с нужными связями, ограничениями и индексами были успешно созданы.

2.10. Заключение второй главы

Реализация базы данных для мессенджера с использованием MySQL, моделирования ERD в MySQL Workbench оказалась ключевым этапом в разработке данного проекта. Использование ERD позволило систематизировать и структурировать все необходимые сущности, их атрибуты и связи, что обеспечило логичное и эффективное проектирование базы данных.

Основные шаги включали создание таблиц для пользователей, сообщений, чатов, вложений и других сущностей, необходимых для функционирования мессенджера. Каждая таблица была определена с учетом её роли в системе: хранение информации о пользователях, их вложений, их истории сообщений и параметров чатов.

Определение связей между таблицами позволило корректно организовать данные, например, установив связи "многие ко многим" для таблиц, отражающих чаты и их участников. Это обеспечило эффективное хранение информации и быстрый доступ к необходимым данным при выполнении запросов.

Генерация SQL скриптов из ERD в MySQL Workbench упростила процесс создания базы данных, что позволило бы быстро перейти к фазе наполнения базы данных тестовыми данными и дальнейшему тестированию функциональности. Это

также способствовало удобству исследования структуры базы данных и её документированию для будущих изменений или оптимизаций.

Таким образом, процесс реализации базы данных для мессенджера с использованием MySQL, MySQL Workbench и моделирования ERD был успешно завершен, обеспечивая надежное и эффективное хранение данных, необходимых для работы приложения.

Заключение

Разработка базы данных для мессенджера представляла собой комплексный процесс, требующий глубокого понимания теоретических основ, тщательного анализа требований и правильного выбора инструментов. Использование MySQL в сочетании с MySQL Workbench для моделирования ERD оказалось эффективным подходом, обеспечивающим структурированное хранение и быстрый доступ к данным. Проектирование и реализация базы данных позволили создать надежную основу для функционирования мессенджера, гарантируя сохранность данных и эффективное выполнение запросов.

Заключение первой главы: Теоретические основы разработки базы данных

В первой главе курсовой работы были рассмотрены основные теоретические аспекты разработки баз данных. Изучены понятие базы данных и её функции, классификация типов баз данных, а также роль нормализации и связей между таблицами для обеспечения целостности данных. Особое внимание уделено вопросам безопасности и проектированию схемы баз данных. Этот теоретический фундамент сыграл важную роль в последующих этапах разработки базы данных для мессенджера.

Заключение второй главы: Разработка базы данных для мессенджера

Во второй главе проведен анализ требований к базе данных мессенджера, что включало определение основных сущностей и их атрибутов. Были выбраны инструменты и технологии для разработки базы данных, включая MySQL и MySQL Workbench для моделирования ERD. Проектирование базы данных включало создание таблиц, определение связей между ними и установку необходимых ограничений. Реализация базы данных была выполнена с учетом всех выявленных требований и анализа, обеспечивая функциональность и производительность системы. Этот этап завершился успешно благодаря систематическому подходу к разработке и тестированию функциональности базы данных.

Таким образом, курсовая работа по разработке базы данных для мессенджера позволила глубже понять и применить теоретические знания в практической среде, а также научилась использовать современные инструменты для проектирования и реализации баз данных. Полученный опыт будет полезен для дальнейшего профессионального развития в области баз данных и информационных технологий.

Список использованной литературы

Электронные ресурсы:

1. Кафедра ИТ – it.vshp.online – [Электронный ресурс] – Режим доступа: https://it.vshp.online
2. Объектно–ориентированная база данных – Wikipedia – [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Объектно–ориентированная_база_данных
3. Функциональные базы данных – Cyclowiki – [Электронный ресурс] – Режим доступа: https://cyclowiki.org/wiki/Функциональные_базы_данных
4. MySQL – Wikipedia – [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/MySQL
5. MySQL Преимущества и недостатки – Servergate – [Электронный ресурс] – Режим доступа: https://servergate.ru/articles/mysql-preimushchestva-i-nedostatki/
6. SQL: что это, в каких базах его используют и как работать с языком программирования – Skillbox Media – [Электронный ресурс] – Режим доступа: https://skillbox.ru/media/code/chto-takoe-sql-kak-ustroen-zachem-nuzhen-i-kak-s-nim-rabotat/
7. База данных – Wikipedia – [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/База_данных
8. Иерархическая база данных - Habr - [Электронный ресурс] - Режим доступа: https://habr.com/ru/articles/771676/
9. Сетевая модель данных - Wikipedia - [Электронный ресурс] - Режим доступа: https://ru.wikipedia.org/wiki/%D0%A1%D0%B5%D1%82%D0%B5%D0%B2%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85
10. Реляционная модель данных – Wikipedia – [Электронный ресурс] – Режим доступа: https://ru.wikipedia.org/wiki/Реляционная_модель_данных
11. NoSQL – Wikipedia - [Электронный ресурс] - Режим доступа: https://ru.wikipedia.org/wiki/NoSQL
12. Графовая база данных - Wikipedia - [Электронный ресурс] - Режим доступа: https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84%D0%BE%D0%B2%D0%B0%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85
13. Resource Description Framework – Wikipedia - [Электронный ресурс] - Режим доступа: https://en.wikipedia.org/wiki/Resource_Description_Framework

14. Система управления базами данных - Wikipedia - [Электронный ресурс] - Режим доступа: https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0_%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D0%B1%D0%B0%D0%B7%D0%B0%D0%BC%D0%B8_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85

Приложение 1. Репозиторий GitHub

Ссылка на репозиторий:
https://github.com/MeldyTheCoder/messagnger_db

Приложение 2.

```
-- Создание роли для бэкенда
CREATE USER 'backend'@'localhost' IDENTIFIED BY 'x|vnsT25Zo';

-- Разрешение на просмотр, добавление и редактирования пользователей.
GRANT SELECT, INSERT, UPDATE ON `users` TO 'backend'@'localhost';

-- Разрешение на просмотр, добавление и удаление media-файлов.
GRANT SELECT, INSERT, DELETE ON `media` TO 'backend'@'localhost';

-- Разрешение на просмотр, добавление и удаление вложений для сообщений.
GRANT SELECT, INSERT, DELETE ON `messages_attachments` TO
'backend'@'localhost';
GRANT SELECT, INSERT, DELETE ON `attachments` TO 'backend'@'localhost';

-- Разрешение на просмотр, добавление и удаление сообщений.
GRANT SELECT, INSERT, DELETE ON `messages` TO 'backend'@'localhost';

-- Разрешение на просмотр, добавление, удаление и редактирование связей
пользователей с чатами.
GRANT ALL PRIVILEGES ON `chat_users` TO 'backend'@'localhost';

-- Разрешение на просмотр, добавление, удаление и редактирование
пользовательских фото.
GRANT ALL PRIVILEGES ON `users_photos` TO 'backend'@'localhost';

-- Разрешение на просмотр, добавление, удаление и редактирование чатов.
GRANT ALL PRIVILEGES ON `chats` TO 'backend'@'localhost';

-- Сохранение изменений
FLUSH PRIVILEGES;
```

Приложение 3.

```
DELIMITER $$
CREATE PROCEDURE sendMessage(
 IN fromUserId BIGINT,
 IN chatIdTo BIGINT,
 IN message VARCHAR(255),
 OUT messageId BIGINT
)
BEGIN
   -- Объявление переменных
   DECLARE chatExists BOOL;
   DECLARE userInChat BOOL;
   DECLARE userExists BOOL;

   -- Проверка пользователя на существование в БД
   SELECT COUNT(`users`.`id`) > 0 INTO userExists
   FROM users WHERE id = fromUserId;

   -- Проверка чата на существование в БД
   SELECT COUNT(`chats`.`id`) > 0 INTO chatExists
   FROM chats WHERE id = chatIdTo;

   -- Проверка пользователя на наличие в участниках чата
   SELECT COUNT(`users`.`id`) > 0 INTO userInChat
   FROM `users`
   JOIN `chat_users` ON `user`.`id` = `chat_users`.`id`
   WHERE (
      `chat_users`.`chat` = chatIdTo
       AND
       `chat_users`.`user` = fromUserId
   );

   -- Начало транзакции
   START TRANSACTION;

   -- Условие на существование пользователя
   -- Если пользователь не существует - отображает исключение
   IF NOT userExists THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ошибка: пользователь не
найден!';
       ROLLBACK;
   END IF;
```

```
   -- Условие на существование чата
   -- Если чат не существует - отображает исключение
   IF NOT chatExists THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ошибка: данный чат не
существует!';
       ROLLBACK;
   END IF;

   -- Условие на проверку пользователя в участниках чата
   -- Если пользователь не является участником чата - отображает исключени
   IF NOT userInChat THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ошибка: пользователь не
находится в данном чате!';
       ROLLBACK;
   END IF;

   -- Добавление данных в такблицу сообщений
   INSERT INTO messages (
      from_user,
       chat,
       message
   ) VALUES (
      fromUserId,
       chatIdTo,
       message
   );

   COMMIT;
   SELECT LAST_INSERT_ID();
END $$
DELIMITER ;
```

Приложение 4.

```
DELIMITER $$
CREATE PROCEDURE deleteMessage(
 IN chatId BIGINT,
 IN messageId BIGINT,
 IN fromUserId BIGINT
)
BEGIN
   -- Объявление переменных
   DECLARE chatExists BOOL;
   DECLARE userInChat BOOL;
   DECLARE userExists BOOL;
   DECLARE messageExists BOOL;

   -- Проверка пользователя на существование в БД
   SELECT COUNT(`users`.`id`) > 0 INTO userExists
   FROM users WHERE id = fromUserId;

   -- Проверка чата на существование в БД
   SELECT COUNT(`chats`.`id`) > 0 INTO chatExists
   FROM chats WHERE id = chatIdTo;

   -- Проверка пользователя на наличие в участниках чата
   SELECT COUNT(`users`.`id`) > 0 INTO userInChat
   FROM `users`
   JOIN `chat_users` ON `user`.`id` = `chat_users`.`id`
   WHERE (
      `chat_users`.`chat` = chatIdTo
       AND
       `chat_users`.`user` = fromUserId
   );

   -- Проверка на наличие сообщения от указанного пользователя
   SELECT COUNT(`messages`.`id`) > 0 INTO messageExists
   FROM `messages`
   WHERE `from_user` = fromUserId AND `id` = messageId;

   -- Начало транзакции
   START TRANSACTION;

   -- Условие на существование пользователя
   -- Если пользователь не существует - отображает исключение
   IF NOT userExists THEN
```

```
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ошибка: пользователь не
найден!';
       ROLLBACK;
   END IF;

   -- Условие на существование чата
   -- Если чат не существует - отображает исключение
   IF NOT chatExists THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ошибка: данный чат не
существует!';
       ROLLBACK;
   END IF;

   -- Условие на проверку пользователя в участниках чата
   -- Если пользователь не является участником чата - отображает исключени
   IF NOT userInChat THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ошибка: пользователь не
находится в данном чате!';
       ROLLBACK;
   END IF;

   -- Условие на существование сообщение от указанного пользователя
   -- Если сообщение не существует - отображает исключение.
   IF NOT messageExists THEN
      SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Ошибка: сообщение от дан
отправителя не найдено!';
       ROLLBACK;
   END IF;

   -- Удаление сообщения по ID
   DELETE FROM messages WHERE id = messageId;

   COMMIT;
END $$
DELIMITER ;
```

Уважаемый пользователь!
Обращаем ваше внимание, что система Антиплагиус отвечает на вопрос, является тот или иной фрагмент текста заимствованным или нет. Ответ на вопрос, является ли заимствованный фрагмент именно плагиатом, а не законной цитатой, система оставляет на ваше усмотрение.

Отчет о проверке № 8897261

Дата выгрузки: 2024-06-20 04:04:09
Пользователь: software.dev1988@gmail.com, ID: 8897261

Отчет предоставлен сервисом «Антиплагиат»
на сайте antiplagius.ru/

Информация о документе

№ документа: 8897261
Имя исходного файла: МДК.11.01 - Курсовой проект - Грошелев Кирилл - ИСИП 3-2.docx
Размер файла: 0.09 МБ
Размер текста: 43044
Слов в тексте: 6155
Число предложений: 521

Информация об отчете

Дата: 2024-06-20 04:04:09 - Последний готовый отчет
Оценка оригинальности: 80%
Заимствования: 20%

80.68% 19.32%

Источники:

Доля в тексте	Ссылка
19.80%	https://appmaster.io/ru/blog/sovety-po-izucheniiu-proektirovanii...
18.90%	https://appmaster.io/ru/blog/osnovy-proektirovaniia-baz-dannykh
14.10%	https://cyberpedia.su/4x9bf4.html
14.00%	https://scienceforum.ru/2016/article/2016020613
13.70%	https://www.arsis.ru/blog/nosql
12.40%	https://habr.com/ru/companies/ruvds/articles/727474/
12.10%	https://habr.com/ru/articles/819971/
11.50%	https://nsportal.ru/npo-spo/informatika-i-vychislitelnaya-tekhni...
11.30%	https://business-analytics-russia.ru/vvedenie-v-proektirovanie-b...
11.00%	https://www.oracle.com/cis/database/what-is-database/
10.60%	https://thrusta.narod.ru/disc/teis.htm
10.50%	https://nuancesprog.ru/p/19722/
10.30%	https://www.roksis.ru/articles/subd-vidy-osobennosti-klassifikat...
10.00%	https://www.oracle.com/cis/database/nosql/what-is-nosql/
9.70%	https://it-vacancies.ru/blog/rabota-s-bazami-dannyx-sql-vs-nosql...
9.50%	https://metanit.com/sql/tutorial/1.1.php

Доля в тексте	Ссылка
9.30%	https://bdstudy.ru/?p=90
9.20%	https://www.itshop.ru/Obespechenie-masshtabiruemosti-dannyh-obla...
9.10%	https://systems.education/understanding-database-types
9.10%	https://habr.com/ru/articles/254773/
8.70%	https://nsportal.ru/shkola/informatika-i-ikt/library/2020/12/25/...
8.60%	https://appmaster.io/ru/blog/sistema-upravleniia-reliatsionnymi-...
8.30%	https://www.astera.com/ru/type/blog/relational-database-manageme...
8.20%	https://www.osp.ru/dbms/1995/04/13031442
7.80%	https://studfile.net/preview/21475312/page:6/
7.60%	https://bi-data.ru/blog/2024/01/17/%D0%B2%D0%B2%D0%B5%D0%B4%D0%B...
7.50%	https://mlcentre.ru/articles/722871/
7.10%	https://otvet.mail.ru/question/59655740

Информация о документе:

Частное учреждение профессионального образования "Высшая школа предпринимательства" (ЧУПО "ВШП") КУРСОВОЙ ПРОЕКТ "Разработка базы данных для мессенджера" Выполнил: студент 3-го курса специальности 09.02.07 "Информационные системы и программирование" Грошелев Кирилл Дмитриевич подпись: _____ Проверил: преподаватель дисциплины, преподаватель ЧУПО "ВШП", к.ф.н. Ткачев П.С. оценка: _____ подпись: _____ Содержание Введение 2 1. Теоритические основы разработки базы данных 5 1.1. Понятие базы данных и её основные функции 5 1.2. Типы баз данных 5 1.3. Инструменты для разработки базы данных 7 1.4. Нормализация баз данных 8 1.5. Связи между таблицами в базах данных 9 1.6. Безопасность баз данных 10 1.7 Проектирование схемы базы данных 11 1.8. Заключение первой главы 12 2. Разработка базы данных для мессенджера 13 2.1. Анализ требований 13 2.2. Типовые запросы 15 2.3. Хранимые процедуры 16 2.4. Триггеры 18 2.5. Представления 19 2.6. Пользовательские функции 19 2.7. Выбор инструментов и технологий для разработки базы данных 20 2.8. Проектирование базы данных 22 2.9. Реализация базы данных 24 2.10. Заключение второй главы 25 Заключение 26 Список использованной литературы 28 Приложение 1. Репозиторий GitHub 29 Приложение 2. 31 Приложение 3. 32 Приложение 4. 33 Введение Актуальность темы В наше время огромное количество фирм используют персональные компьютеры для сохранения и обработки любого вида информации. Эта информация содержится в базах данных. Базы данных играют важную роль в развивающемся мире технологий. Всё, с чем мы каждый день взаимодействуем в жизни, по всей видимости, зафиксировано в какой-нибудь базе. Работа с базами данных является важнейшим навыком в работе с компьютером, а специалисты данной области становятся всё более востребованными. Главные идеи нынешней информационной методики базируются на представлении, в соответствии чему информация должна быть образована в базы данных с задачей отображения динамически изменяющегося мира и удовлетворения всех потребностей в информации у пользователей. Базы данных формируются и работают под управлением специальных программных средств, называемых системами управления базами данных. В настоящее время мессенджеры стали неотъемлемой частью нашей повседневной жизни, обеспечивая обмен сообщениями, файлами, аудио- и видеозвонками. Они служат удобным проводником в общении между лицом или гуппой лиц. Для обеспечения стабильной работы и удобного пользовательского интерфейса мессенджеры используют специальные базы данных, которые хранят информацию о пользователях, сообщениях, файлов и прочих данных. Разработка базы данных для мессенджера требует тщательного планирования и оптимизации, учитывая большое количество пользователей, высокую нагрузку на серверы и необходимость обеспечения безопасности и конфиденциальности данных. В данной курсовой работе мы рассмотрим основные принципы разработки базы данных для мессенджера, ее структуру, функциональность и возможные проблемы, с которыми может столкнуться разработчик. Цель работы Цель данной курсовой работы заключается в попытке разработать базу данных для мессенджера, обеспечивающую практичное хранение данных для сообщений, пользователей, чатов и т.д Задачи Для достижения заданной мне цели, мне нужно выполнить следующие задачи: * Обозначить требования к функционалу базы данных * Выбрать инструменты и технологии для разработки * Спроектировать базу данных на основе требований к функционалу * Реализовать базу данных на основе спроектированной схемы * Определить требования к базе данных, обеспечивающие целостность системы 1. Теоритические основы разработки базы данных 1.1. Понятие базы данных и её основные функции База данных - это организованная структура, которая предназначена для хранения информации. В то время, когда происходило развитие термина баз данных, в них сохранялись исключительно информация, однако уже в наши дни

многие системы управления базами данных позволяет размещать в своих структурах и данные, и программный код, с помощью которого совершается связь с пользователями или с другими программно-аппаратными комплексами. При этом данные не должны противоречить друг другу, быть целостными и не избыточными. База данных создается для сохранения и непосредственного доступа к информации, содержащей сведения об искомой предметной области. Прежде всего, целью использования информации из баз данных и сложностью информационных процессов, существующих в пределах предметной области в конкретных условиях. Основными функциями базы данных являются: * Хранение данных: БД обеспечивает безопасное и надёжное хранение информации. * Обработка данных: с помощью запросов и операций можно выполнять различные действия с данными, такие как поиск, сортировка, фильтрация и т. д. * Анализ данных: базы данных позволяют проводить анализ данных, выявлять закономерности и тенденции, а также принимать обоснованные решения на основе полученной информации. Система управления базами данных - это программный механизм, предназначенный для записи, поиска, сортировки, обработки и печати информации, содержащейся в базе данных. Иными словами, система управления базами данных - механизм, регулирующий работу самой базы данных.[14] 1.2. Типы баз данных Существует множество типов баз данных, каждый из которых имеет свои особенности и преимущества в зависимости от поставленной задачи и требований проекта. В данной работе будут рассмотрены основные типы баз данных: * Реляционные базы данных (РБД) - это наиболее распространённый тип баз данных. Они основаны на реляционной модели данных и используют таблицы для хранения информации. Реляционные базы данных имеют множество преимуществ, таких как простота использования, надёжность и эффективность. Однако они также имеют некоторые недостатки, такие как сложность масштабирования и ограничения в производительности при работе с большими объёмами данных. [10] * Объектно-ориентированные базы данных (ООБД) - этот тип баз данных основан на объектно-ориентированной модели данных. В ООБД данные представлены в виде объектов, а не таблиц. Это позволяет более точно моделировать сложные предметные области и обеспечивает более высокую гибкость при разработке приложений. Однако ООБД имеют более сложную структуру и требуют более высокой квалификации разработчиков. [2] * Иерархические базы данных (ИБД) - в иерархических базах данных информация организована в виде древовидной структуры. ИБД используются для моделирования иерархических отношений между объектами, например, в системах управления проектами или в каталогах товаров. Иерархические базы данных просты в использовании и обеспечивают хорошую производительность при работе с небольшими объёмами данных, но они не подходят для сложных запросов и обработки больших объёмов данных. [8] * Сетевые базы данных (СБД) - сетевые базы данных похожи на иерархические, но в них допускается наличие нескольких родительских узлов у одного дочернего узла. СБД используются для моделирования сложных отношений между объектами и обеспечивают большую гибкость по сравнению с иерархическими базами данных. Однако СБД сложнее в реализации и имеют более низкую производительность по сравнению с реляционными базами данных. [9] * NoSQL базы данных - NoSQL базы данных представляют собой класс нереляционных баз данных, который включает в себя широкий спектр технологий, разработанных для обеспечения масштабируемости и гибкости. NoSQL базы данных не используют схемы и могут обрабатывать различные типы данных, включая документы, пары ключ-значение и графы. Они часто используются для обработки больших объёмов неструктурированных данных, таких как данные социальных сетей, журналов веб-сервера и т. д. [11] * Графовые базы данных - графовые базы данных предназначены для работы с данными, представленными в виде графа. Графовые базы данных позволяют эффективно моделировать и обрабатывать отношения между объектами. Они используются в таких областях, как социальные сети, рекомендательные системы и анализ данных.[12] * RDF (Resource Description Framework) - это стандарт представления данных, который используется для описания ресурсов в интернете. RDF позволяет описывать ресурсы с помощью троек "субъект-предикат-объект", где субъект и объект являются ресурсами, а предикат описывает отношение между ними. RDF является основой для создания семантических веб-приложений, которые используют машинное понимание данных для предоставления более интеллектуальных услуг.[13] 1.3. Инструменты для разработки базы данных Разработка баз данных включает в себя проектирование, создание, управление и оптимизацию баз данных. Для этих целей существует множество инструментов, каждый из которых предлагает определенные функции и возможности. В этом разделе будут рассмотрены основные инструменты для разработки баз данных. Инструменты для разработки баз данных делятся на следующие категории: * Системы управления базами данных (СУБД) - это программное обеспечение, предназначенное для создания, управления и модификации баз данных. Например, MySQL - распространенная реалиационная СУБД с открытым исходным кодом. * Инструменты моделирования данных: помогают разработчикам создавать и визуализировать структуры баз данных. Например, Microsoft Visio - универсальный инструмент для создания диаграмм, включая диаграммы сущность-связь (ERD). * Инструменты управления базами данных: предоставляют интерфейсы для администрирования и управления базами данных. Например, MySQL Workbench - инструмент для управления СУБД MySQL. * Инструменты разработки и интеграции: облегчают процесс написания, тестирования и отладки SQL-кода и приложений, работающих с базами данных. Например, DBeaver - многофункциональный инструмент для работы с различными СУБД, поддерживающий множество форматов данных. * Инструменты мониторинга и оптимизации: эти инструменты помогают отслеживать производительность баз данных и оптимизировать запросы. Например, Oracle AWR - инструмент для анализа производительности баз данных Oracle. Для разработки следует в лучшем случае выбрать инструмент из каждой категории, изходя из используемой СУБД и из личных предпочтений. Использование данных инструментов оптимизирует работу разработчика и сделает ее намного проще и качественнее. 1.4. Нормализация баз данных Нормализация - это процесс организации структуры базы данных таким образом, чтобы минимизировать избыточность данных и обеспечить целостность информации. Нормализация позволяет улучшить производительность базы данных, упростить её обслуживание и обеспечить более эффективное использование ресурсов. Существует несколько уровней нормализации, каждый из которых устраняет определённые виды избыточности. Чем выше уровень нормализации, тем больше таблиц требуется для представления данных, но при этом повышается целостность данных и уменьшается вероятность возникновения аномалий обновления. Уровни нормализации существуют следуюшие: * Первый уровень нормализации, или 1НФ, является

базовым уровнем нормализации реляционных баз данных. Он требует, чтобы все атрибуты были атомарными, то есть не содержали в себе других значений. Это означает, что каждый атрибут должен представлять собой одно значение, а не набор значений или список. * Второй уровень нормализации, или 2НФ, является следующим шагом после достижения 1НФ. Он требует выполнения 1НФ и отсутствия частичной зависимости неключевых атрибутов от первичного ключа. Частичная зависимость возникает, когда неключевой атрибут зависит только от части первичного ключа. Для устранения частичной зависимости необходимо разбить таблицу на две или более таблиц. * Третий уровень нормализации, или 3НФ, является самым строгим уровнем нормализации реляционных баз данных. Он требует выполнения 2НФ и отсутствия транзитивной зависимости неключевых атрибутов от первичного ключа. Транзитивная зависимость возникает, когда неключевой атрибут зависит от другого неключевого атрибута, который, в свою очередь, зависит от первичного ключа. * Нормальная форма Бойса-Кодда (BCNF) - для любой функциональной зависимости X → A в отношении R атрибут A должен функционально зависеть от каждого элемента в X (то есть X должен быть суперключом отношения R). * Четвёртый уровень (4NF) - отношение находится в 4NF, если оно находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от её потенциальных ключей. * Пятый уровень или нормальная форма проекции-соединения (5NF или PJ/NF) - любая зависимость соединения в ней определяется потенциальными ключами отношения (ключами исходного отношения). Также отношение R соответствует 5NF тогда и только тогда, когда каждая нетривиальная зависимость соединения в R следует из существования некоторого потенциального ключа K для R. * Шестой уровень нормализации (6NF) - также известен как Domain/Key Normal Form (DKNF) или идеальная нормальная форма. Это самый высокий уровень нормализации, который требует, чтобы каждый атрибут отношения зависел от первичного ключа этого отношения и не зависел функционально от любого другого атрибута. Другими словами, отношение находится в DKNF, когда оно уже находится в BCNF и каждый домен атрибутов полностью зависит от первичного ключа. Важно отметить, что шестой уровень нормализации редко используется на практике, так как он может привести к чрезмерной декомпозиции базы данных и усложнить её понимание и использование. Кроме того, многие системы управления базами данных не поддерживают этот уровень нормализации. 1.5. Связи между таблицами в базах данных В базах данных (БД) связи между таблицами играют ключевую роль в обеспечении целостности данных, а также в упрощении процесса обработки информации. Связь между двумя таблицами представляет собой зависимость между данными, которые хранятся в этих таблицах. Существует несколько типов связей, каждый из которых имеет свои особенности и преимущества: * Один к одному. В этом случае каждой записи в одной таблице соответствует только одна запись в другой таблице. Этот тип связи используется редко, так как он не позволяет эффективно использовать данные. * Один ко многим. Это наиболее распространённый тип связи, который используется для моделирования отношений "один ко многим". Например, один клиент может иметь несколько заказов. * Многие ко многим. Этот тип связи требует использования третьей таблицы, которая будет служить связующим звеном между двумя другими таблицами. Например, многие студенты могут изучать несколько предметов, и каждый предмет может изучаться многими студентами. * Самосвязь. Этот тип связи не требует использования отдельных таблиц для реализации. При таком типе связи внешний ключ определенной таблицы ссылается на запись с этой же таблицы. Например, сущность пользователя хранит в себе внешний ключ сущности другого пригласившего на сайт пользователя, Стоит отметить, что связи используются только в реалиационных БД. Другие типы БД не поддерживают данный функционал. 1.6. Безопасность баз данных Безопасность базы данных (БД) - это совокупность мер и стратегий, направленных на защиту данных от несанкционированного доступа, утечек, повреждений и других угроз. В современных условиях обеспечения безопасности информации защита баз данных становится критически важной задачей. Основные аспекты безопасности базы данных: * Аутентификация - это процесс проверки подлинности пользователя или системы, пытающихся получить доступ к базе данных. * Авторизация - это процесс определения прав и привилегий пользователя после аутентификации. * Ролевое управление доступом (RBAC): Предоставляет доступ к данным на основе ролей, назначенных пользователям. * Политики доступа (ACL): детализируют права доступа на уровне таблиц, строк или столбцов. * Шифрование: защищает данные, делая их нечитаемыми для неавторизованных пользователей. Возможен вариант как шифровки данных, хранящиеся на диске, с использованием алгоритмов AES или RSA, так и шифровки данных, передаваемых между клиентом и сервером, с использованием SSL/TLS. * Управление уязвимостями: регулярное обновление и исправление программного обеспечения базы данных для устранения известных уязвимостей. Важно своевременно устанавливать обновления безопасности. * Мониторинг и аудит: Использование систем обнаружения вторжений (IDS) и ведение журналов аудита для выявления и анализа подозрительной активности. * Бэкапы и восстановление данных: резервное копирование данных помогает восстановить базу данных в случае утраты или повреждения данных. * Управление сеансами и время жизни паролей: контроль сеансов для отключения нежелательных пользователей. * Контроль доступа: оргранничение физического и логического доступа к серверам баз данных. Применение этих мер позволяет значительно уменьшить риски утечки данных и обеспечить защиту информации в базе данных. 1.7 Проектирование схемы базы данных Проектирование схемы базы данных представляет собой процесс создания структуры базы данных, включающий определение таблиц, столбцов, связей, индексов, ограничений и т.д. Проектирование схемы базы данных включает следующие шаги: * Определение требований к базе данных: анализ требований, определение целей и задач, которые должна решать база данных. Изучение потребностей пользователей и процессов, которые могут быть автоматизированы с помощью базы данных. * Определение сущностей и атрибутов: создание таблиц (сущностей), создание необходимых по требованиям полей, в которых будет храниться нужная информация. * Выбор СУБД: выбор подходящей СУБД для проектирования на основе выбранного типа базы данных. * Определение связей между сущностями: Определение первичных и внешних ключей для обеспечения целостности и согласованности данных в базе данных. * Создание диаграммы (ERD): визуально отображает сущности и их связи. Это помогает лучше понять структуру данных и связи между ними. * Нормализация данных: организации данных в таблицы для уменьшения избыточности и обеспечения целостности данных * Определение первичных и внешних ключей: Первичные ключи уникально идентифицируют записи в таблице. Внешние ключи устанавливают

связи между таблицами и обеспечивают целостность данных. Тем самым, данными методами мы обеспечиваем связи межу таблицами. * Создание физической схемы базы данных: реализация логической схемы в конкретной СУБД. Она включает в себя создание таблиц, индексов, представлений и других объектов базы данных. При проектировании схемы базы данных необходимо учитывать требования к производительности, безопасности и надежности. Важно обеспечить, чтобы база данных была масштабируемой и могла обрабатывать увеличивающиеся объемы данных без потери производительности. 1.8. Заключение первой главы В заключение первой главы можно сделать вывод о важности теоретических основ разработки базы данных и её значении для эффективного управления и обслуживания клиентов. Основные принципы и рекомендации по разработке базы данных включают: * Анализ по требованиям к функционалу базы данных. * Выбор типа базы данных * Выбор инструментов для разработки базы данных по техническим требованиям и личным предпочтениям. * Определние нормальной формы базы данных * Проектирование схемы базы данных с учетом требований к производительности, безопасности и надежности. Эти принципы и рекомендации являются основой для успешной разработки и внедрения базы данных, которая будет эффективно поддерживать деятельность бара "Кооператив" и обеспечивать высокое качество обслуживания клиентов. 2. Разработка базы данных для мессенджера 2.1. Анализ требований Проектирование базы данных для мессенджера требует тщательного анализа функциональных и нефункциональных требований системы. В данном параграфе я попытаюсь изложить требования к структуре базы данных мессенджера, которые охватывают к