Министерство науки и высшего образования Российской Федерации

Муромский институт (филиал)

федерального государственного бюджетного образовательного учреждения высшего образования

«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых» (МИ ВлГУ)

Факультет информационных технологий

Кафедра ФПМ

КУРСОВАЯ РАБОТА

По Базы данных

(наименование дисциплины)

Тема Проектирование базы данных для хранения данных социальной сети

Руководитель

Макаров К.В.

(фамилия, инициалы)

(оценка)

(подпись) (дата)

Члены комиссии

Студент ПМИ-118

(группа)

Семенов И.А.

(фамилия, инициалы)

(оценка) (Ф.И.О)

(подпись) (дата)

(подпись) (дата)

(оценка) (Ф.И.О)

(подпись) (дата)

Муром 2020

Введение

В течение последнего десятилетия (2010 – 2020 гг) в бытовой жизни человеческого общества формировалось принятие технологически сложных процессов и устройств. Развитие глобальных информационных сетей является таким процессом. Понятно, и на данный момент очевидно, что социальные сети являются неотъемлемой частью любой такой глобальной сети.

Одновременно с постепенным и быстрым развитием технологий, которые позволяют единовременно получать много информации, происходил процесс внедрения глобальных социальных сетей в реальную социальную жизнь.

Становится ясно, что использование общепринятых протоколов связи и доступа к подобным сетям часто ставит под угрозу личные данные участников сети. Исходя из такого понимания, представляется логичным стремление создать уникальный способ взаимодействия между узким кругом лиц внутри глобальной сети.

Очевидно, что в рамках курсового проектирования невозможно реализовать сколько-нибудь защищенный канал связи для подобного взаимодействия. Однако возможно создать собственную социальную сеть, и уже на личном примере понимать, какие ошибки и недочеты при работе внутри общепринятых социальных сетей стоит избегать. Таким образом, была поставлена цель курсовой работы и сформулированы задачи, позволяющие ее достичь.

Цель: создать информационную систему, реализующую сервис социальной сети.

Задачи:

1. Определение данных, которые должен хранить медиа сервис "социальная сеть"

2. Определение взаимодействия таких данных

3. Завершение проектирования рабочим прототипом

1. Анализ технического задания

1.1. Формулировка задания на курсовую работу

Задание курсовой работы – проектирование и реализация базы данных "Данные социальной сети". Настоящая пояснительная записка курсовой работы отражает выполнение следующих этапов:

1. Создание базы данных с использованием СУБД FireBird 2.5 (выбор версии предопределен заданием курсовой работы)

2. Реализация пользовательского интерфейса информационной системы в виде клиентского приложения, работающего с БД. Клиентское приложение проектируется с использованием Visual Studio 2019 (выбор версии обоснован актуальностью выпускаемых обновлений безопасности).

Для того, чтобы успешно выполнить перечисленные этапы разработки, были сформулированы следующие подзадачи:

1. Проведение анализа предметной области

2. Разработка формальных требований к хранимым в базе данных данным

3. Разработка структуры базы данных

4. Разработка клиентского приложения

1.2. Описание предметной области

Определение социального пространства имеет несколько значений. В статье [1] эти определения обобщены; по заключению автора, разновидности «социальных медиа" на данный момент могут быть определены четырьмя различными способами. Для технического задания на выполняемую курсовую работу наиболее подходящим можно считать следующее определение:

"To define "social media" for our current purposes, we synthesize definitions presented in the literature and identify the following commonalities among current social media services:

1) Social media services are (currently) Web 2.0 Internet-based applications"

Вольный перевод:

"Чтобы определить понятие "социальные медиа" для текущих целей, обобщим определения, представленные в литературе, и выявим следующие общие черты между текущими социальными медиасервисами:

1) социальные медиасервисы (в настоящее время) интернет-приложения Web 2.0"

Поскольку техническое задание на курсовой проект предполагает создание информационной системы, реализующей социальный медиасервис (сеть), определение, данное выше, позволяет точно охарактеризовать предметную область работы: социальные медиасервисы (сети), доступные для общего доступа в сети Web, реализованные с применением принципов Web 2.0.

Общепринятая тенденция развития социальных сетей предполагает некоторый базовый набор функциональности. Предполагая, что реализуемый медиасервис будет ориентирован, в первую очередь, на текстовые данные, можно составить список требований к этому сервису. В то же время, учитывая, что данный проект является работой по курсу "Базы данных", стоит выделять, прежде всего, требования к обработке и хранению данных, а не к определенному функционалу прикладного приложения.

Итак, база данных "Данные социальной сети" должна хранить данные, позволяющие обеспечить необходимые и достаточные для функционирования сети возможности. Перечислим их.

1. Создание, настройка и просмотр (получение данных) личной страницы пользователя. Просмотр предусматривает скрытие некоторых данных в соответствии с настройками приватности. Для данных процессов (не включая настройки приватности) необходимо хранить такие данные, как:

1.1. Уникальный идентификатор

1.2. Фамилия

1.3. Имя

1.4. Изображение пользователя (аватар)

1.5. Пол

1.6. Дата рождения

1.7. Страна

1.8. Город

1.9. Телефон

1.10. Информация "О себе"

1.11. Пароль (в зашифрованном виде)

Здесь и далее используется термин "Приватность" применительно к данным. По причине неочевидности контекста этого определения стоит его охарактеризовать с точки зрения задания на проект. В работе [2] рассматривается проблема защиты данных в интернете. В частности, особенно интересна следующая цитата:

"Однако, часто пользователи не имеют выбора - оставлять или нет запрашиваемые персональные данные, если они хотят пользоваться тем или иным сервисом и в то же время, регистрируясь, могут не читать или не понимать смысл норм, заложенных в соглашении об использовании программного обеспечения или онлайн сервиса."

Разрабатываемый медиасервис должен предоставлять настройки приватности как базовую возможность. В то же время, возвращаясь к приведенной выше цитате, ясно, что таких настроек не должно быть много (в этом случае их назначение станет неявным), и одновременно с этим настройки личных данных должны обеспечивать наиболее полное изменение работы с приватной информацией каждого пользователя сервиса.

2. Настройки приватности и пользовательского интерфейса. Для этих процессов необходимо хранить следующие данные:

2.1. Отображение пола (для всех пользователей, всех друзей, никому или избранным пользователям из списка друзей)

2.2. Отображение даты рождения (аналогичные варианты)

2.3. Возможность просмотра увеличенного аватара (аналогичные варианты)

2.4. Возможность просмотра списка друзей (аналогичные варианты)

2.5. Возможность просмотра списка групп (аналогичные варианты)

2.6. Возможность просмотра записей на странице (аналогичные варианты)

2.7. Возможность отправления сообщений (аналогичные варианты)

2.8. Отображение уведомлений в приложении (отображать или не отображать)

2.9. Воспроизводить звук уведомлений (воспроизводить или нет)

2.10. Отображать содержание уведомлений (отображать или скрывать)

3. Взаимодействие пользователя с другими пользователями и личной страницей (за исключением взаимодействия с группами, документами и сообщениями): добавление новых друзей (с подтверждением со стороны пользователя, которому отправлен запрос на добавление в друзья), удаление друзей (в одностороннем порядке), оценка (оценка может либо быть, либо отсутствовать) записей на страницах других пользователей, добавление записей на свою страницу, загрузка и прикрепление к создаваемой записи аудио, видео, изображения или иного документа. Для этих процессов необходимы такие данные:

3.1. Список добавленных друзей

3.2. Список друзей, которым отправлен запрос на добавление в друзья

3.3. Идентификаторы записей на странице

3.4. Текст для каждой записи на странице

3.5. Прикрепленные к каждой записи на странице документы

3.6. Количество оценок к записи на странице

3.7. Идентификаторы оцененных записей

4. Взаимодействие пользователей с группами: создание группы, настройка группы (администрирование), добавление записей в группу, загрузка и прикрепление к создаваемым записям документов, оценка записей группы, вступление в группу (не требует подтверждения), выход из группы (не требует подтверждения). Настройки приватности для системы групп не предусматриваются. Для процессов, описанных выше, нужны следующие данные:

4.1. Уникальный идентификатор

4.2. Название группы

4.3. Изображение группы (аватар)

4.4. Идентификатор пользователя, создавшего группу (только ему доступна возможность выкладывать записи на страницу группы)

4.5. Список групп, на которые подписан пользователь (для каждого пользователя)

4.6. Список пользователей, которые подписаны на группу (для каждой группы)

4.7. Идентификаторы записей на странице

4.8. Текст для каждой записи на странице

4.9. Прикрепленные к каждой записи на странице документы

4.10. Количество оценок к записи на странице

4.11. Информация "О группе"

5. Взаимодействие пользователей с другими пользователями посредством отправки личных сообщений. Для этого процесса необходимы такие данные:

5.1. Сообщения (включая указатели на документы), отправленные другому пользователю

5.2. Документы, отправленные другому пользователю

6. Взаимодействие пользователя с системой документов: загрузка документа напрямую в общее хранилище документов (предусматривается открытый доступ), загрузка документа на страницу, загрузка документа в группу, загрузка документа в личную переписку с другим пользователем. Для перечисленных процессов необходимо хранить следующие данные:

6.1. Файлы документов, загруженных в общее хранилище, с указанием идентификатора загрузившего

6.2. Идентификаторы документов, загруженных конкретным пользователем в общее хранилище

6.3. Файлы документов, загруженных на страницу пользователя

6.4. Идентификаторы документов, загруженных на страницу пользователя

6.5. Файлы документов, загруженных на страницу группы

6.6. Идентификаторы документов, загруженных на страницу группы

Клиентское приложение, реализующее доступ к создаваемой базе данных, должно обеспечивать его с помощью системы страниц:

1. Личная страница

2. Настройки

3. Друзья

4. Поиск пользователей

5. Группы

6. Поиск групп

7. Загруженные (общие) документы

8. Поиск документов

На данном этапе описание данных (и процессов над ними), которые социальная сеть (в общем случае) должна хранить для функционирования медиа сервиса, заканчивается. Очевидно, что не каждый сервис, предоставляющий услуги социальной сети, хранит все перечисленные данные; точно так же сервис может хранить не описанные выше данные. Это не создает противоречия, поскольку разрабатываемая информационная система является абстрактным сервисом (не реализует конкретную социальную задачу).

Для более полного понимания процессов, происходящих над данными внутри социальной сети, анализируется движение потоков этих данных. Результат анализа представлен ниже (2.3. Движение потоков данных).

1.3. Движение потоков данных

Для обеспечения функционирования социальной сети необходим обмен данных.

Первоначально пользователи создают и настраивают свои страницы. Затем происходят несколько процессов:

1. Пользователи добавляют друг друга в друзья, обмениваясь запросами.

2. Пользователи отправляют друг другу сообщения, обмениваясь текстовой информацией и документами друг с другом.

3. Пользователи добавляют на свою страницу записи, прикрепляя или не прикрепляя к ним документы, таким образом обмениваясь ими сразу со всеми пользователями, которые могут просматривать эти записи согласно настройкам приватности пользователя, добавившего запись.

4. Пользователи ставят (либо убирают) оценки на записи на страницах других пользователей, таким образом обмениваясь информацией об одобрении соответствующих записей с остальными пользователями, которые могут видеть эти записи согласно настройкам приватности.

5. Пользователи добавляют в общее хранилище документы, таким образом обмениваясь ими со всеми.

6. Пользователи создают группы и добавляют на их стены записи, таким образом обмениваясь ими с подписанными на соответствующие группы пользователями.

7. Пользователи ставят (либо убирают) оценки на записи на страницах групп, аким образом обмениваясь информацией об одобрении соответствующих записей с остальными пользователями, которые подписаны на эти группы.

Таким образом, в результате детального рассмотрения процессов, происходящих над данными внутри социальной сети, формируется понимание логических связей между данными медиа сервиса "социальная сеть".

1.4. Обзор аналогов

1.4.1. ВКонтакте

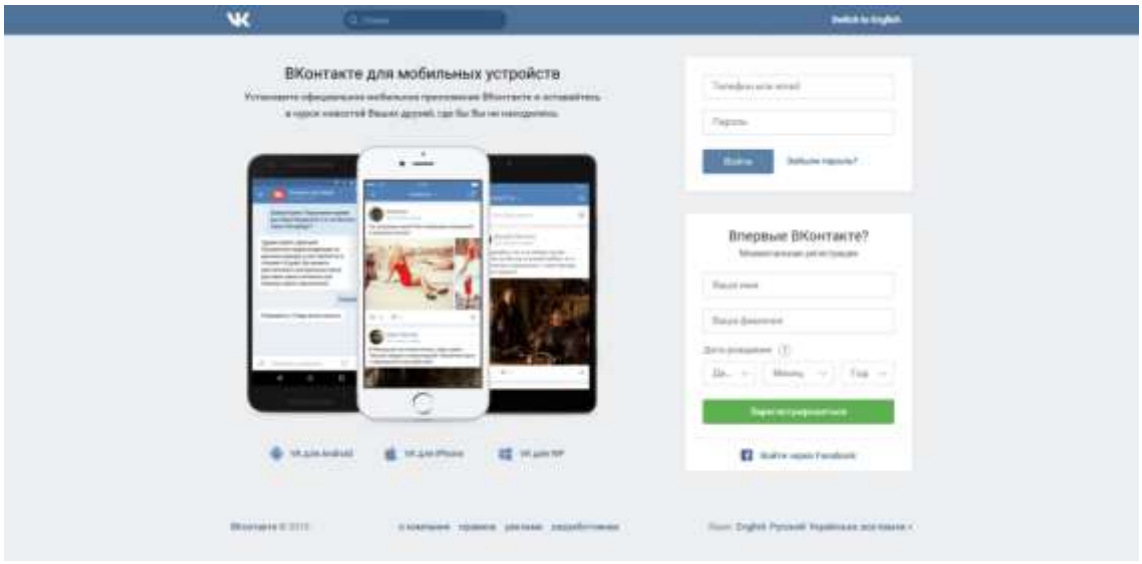

Рисунок 1 — внешний вид сайта ВКонтакте

ВКонтакте — российская социальная сеть. Сервис особенно популярен среди русскоязычных пользователей. Учитывая данные со страницы [3], в медиапространстве этой сети зарегистрировано (на момент обращения), как максимум, 630 514 887 пользователей.

ВКонтакте позволяет:

1. Создавать, настраивать и просматривать личные страницы. Поддерживается описание личной жизни вплоть до образования, карьеры и воинской службы. В качестве средства защиты от нежелательного доступа используется привязка к номеру телефона, пароль, система двухфакторной аутентификации

2. Поддерживается гибкая настройка приватности личной страницы, пользовательского интерфейса в целом и системы уведомлений в частности

3. Взаимодействовать с другими пользователями сети множеством способов: добавлять их в друзья, настраивать родственные связи, отправлять сообщения, приглашения в группы и события и аналогичные действия с другими микросервисами

4. Создавать и администрировать группы (формально подразделяемые на несколько видов по типу администрирования)

5. Ставить оценки в виде есть оценка – нет оценки ("лайки")

1.4.2. Facebook

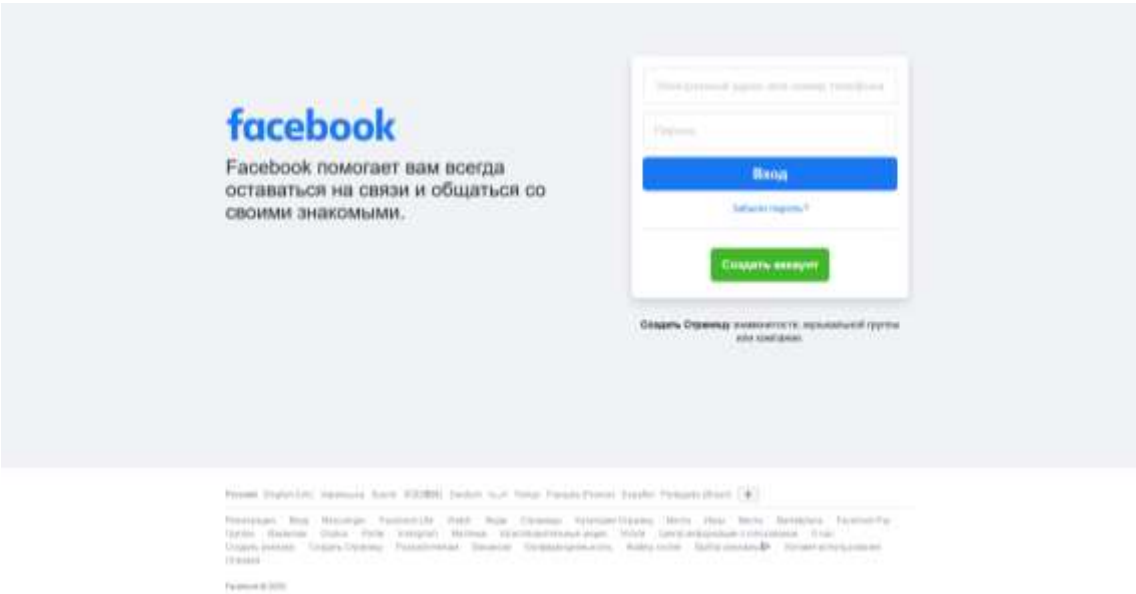

Рисунок 2 — внешний вид сайта Facebook

Facebook — крупнейшая [4] социальная сеть в мире. Была основана 4 февраля 2004 года Марком Цукербергом и его соседями по комнате во время обучения в Гарвардском университете. Начиная с сентября 2006 года сайт доступен для всех пользователей Интернета в возрасте от 13 лет, имеющих адрес электронной почты.

Facebook позволяет:

1. Создавать, настраивать и просматривать личные страницы. Поддерживается добавление фотографии, статуса, описания личной жизни и т.д. Есть возможность использования двухфакторной аутентификации

2. Поддерживается гибкая настройка приватности личной страницы, пользовательского интерфейса в целом и системы уведомлений в частности

3. Взаимодействовать с другими пользователями множеством способов: "подмигивать" (создание уведомления другому пользователю для привлечения его внимания к себе), комментировать записи и т.д.

4. Создавать и администрировать группы

5. Ставить оценки в виде 5 классифицируемых реакций (удивление, смех, грусть, гнев и любовь) и стандартной ("лайк")

В результате проведенного поиска и анализа аналогов создаваемой информационной системы формулируется следующий вывод:

Описанные в разделе 2.2. Описание предметной области данные и их взаимодействие достаточно (для выполняемой курсовой работы) точно отображают логическую структуру данных этих аналогов.

Значит, проектируемая структура базы данных будет соответствовать реальным базам данных существующих социальных сетей. Что в полной мере обеспечивает выполнение поставленной на курсовую работу цели.

2. Проектирование структуры базы данных

2.1. Проектирование предварительных отношений

2.1.1. Пользователи

Для однозначного определения пользователя используется уникальный идентификатор (далее также используется синоним (сокращение от английского identifier) id).

1. Имя сущности – Пользователи

2. Краткое описание – все пользователи, зарегистрированные в социальной сети, и информация для их взаимодействия

3. Список атрибутов – id, фамилия, имя, отчество, аватар, пол, дата рождения, страна, город, телефон, о себе, пароль, отображение пола, отображение даты рождения, отображение аватара, отображение друзей, отображение групп, отображение записей, отображение сообщений, уведомления, звуки, содержимое, друзья, исходящие заявки, входящие заявки, подписки, записи, документы, оцененные записи

4. Потенциальный первичный ключ – id

Таблица 1 – Пользователи (первичное проектирование)

Имя атрибута	Краткое описание
id	Уникальный идентификатор пользователя
Фамилия	Часть имени пользователя
Имя	Часть имени пользователя
Отчество	Часть имени пользователя
Аватар	Адрес файла с картинкой-аватаром в файловой системе
Пол	Пол пользователя, 0 – мужской, 1 – женский
Дата рождения	Дата рождения пользователя
Страна	Страна проживания
Город	Город проживания
Телефон	Личный номер телефона

Таблица 1 – Пользователи (продолжение)

О себе	Произвольно заполняемая пользователем личная информация
Пароль	Ключевое слово для ограничения доступа к странице
Отображение пола	Отображать ли пол на странице, 0 – видит только владелец страницы, 1 – видят друзья, 2 – видят все
Отображение даты рождения	Отображать ли дату рождения на странице
Отображение аватара	Отображать ли аватар на странице
Отображение друзей	Отображать ли список друзей на странице
Отображение групп	Отображать ли список групп на странице
Отображение записей	Отображать ли список записей на странице
Отображение сообщений	Возможность связаться с пользователем с помощью сообщений
Уведомления	Показывать ли пользователю уведомления о входящих сообщениях, 0 – нет, 1 – да
Звуки	Воспроизводить ли звук уведомления
Содержимое	Отображать ли текст уведомления
Друзья	Список id друзей
Исходящие заявки	Исходящие заявки на добавление в друзья (id пользователей)
Входящие заявки	Входящие заявки на добавление в друзья (id пользователей)
Подписки	Список id групп, на которые подписан пользователь
Записи	Текст записей на странице пользователя
Документы	Документы, прикрепленные к определенной записи на странице
Оцененные записи	Записи пользователей и групп, которые оценил пользователь

2.1.2. Группы

Для однозначного определения группы используется уникальный идентификатор. Он не может совпадать с любым идентификатором пользователя, но кодируется также 8 цифрами.

Стоит заметить, что в этом случае необходимо устанавливать соответствие между идентификатором и сущностью, на которую он ссылается, поскольку иначе невозможно однозначно определить, какой сущности принадлежит конкретный id – пользователю или группе. Такое соответствие выражается в виде отдельного отношения – Классы id.

2.1.3. Классы id

1. Имя сущности – Классы id

2. Краткое описание – все пользователи, зарегистрированные в социальной сети, и информация для их взаимодействия

3. Список атрибутов – id, фамилия, имя, отчество, аватар, пол, дата рождения, страна, город, телефон, о себе, пароль, отображение пола, отображение даты рождения, отображение аватара, отображение друзей, отображение групп, отображение записей, отображение сообщений, уведомления, звуки, содержимое, друзья, исходящие заявки, входящие заявки, подписки, записи, документы, оцененные записи

4. Потенциальный первичный ключ – id

Таблица 2 – Классы идентификаторов (первичное проектирование)

Имя атрибута	Краткое описание
id	Уникальный идентификатор (группы или пользователя)
Класс	Класс идентификатора – пользователи (0) или группы (1)

Продолжение 2.1.2 Группы

1. Имя сущности – Группы

2. Краткое описание – все группы, зарегистрированные пользователями в социальной сети и список ее подписчиков

3. Список атрибутов – id, название, аватар, id владельца, о группе, записи, документы

4. Потенциальный первичный ключ – id

Таблица 3 – Группы (первичное проектирование)

Имя атрибута	Краткое описание
id	Уникальный идентификатор группы
Название	Имя группы
Аватар	Адрес файла с картинкой-аватаром в файловой системе
id владельца	Уникальный идентификатор пользователя
О группе	Произвольно заполняемая пользователем информация о тематике группы
Записи	Текст записей на странице группы
Документы	Документы, прикрепленные к определенной записи на странице

2.1.4. Сообщения

1. Имя сущности – Сообщения

2. Краткое описание – все сообщения, отправленные пользователями

3. Список атрибутов – id сообщения, id отправителя, id получателя, текст сообщения, документы

4. Потенциальный первичный ключ – id сообщения

Таблица 4 – Сообщения (первичное проектирование)

Имя атрибута	Краткое описание
id сообщения	Уникальный идентификатор сообщения
id отправителя	id пользователя, отправившего сообщение
id получателя	id пользователя, которому предназначено сообщение

Таблица 4 – Сообщения (продолжение)

Текст сообщения	Содержимое сообщения
Документы	Прикрепленные к сообщению документы (адрес)

2.2. Приведение базы данных к первой нормальной форме

Здесь и далее определения нормальных форм соответствуют таковым из учебно-методического пособия [5].

Определение: отношение находится в первой нормальной форме, если все его атрибуты являются простыми (имеют единственное значение).

2.2.1. Пользователи

Первая сущность (Пользователи) не удовлетворяет первой нормальной форме, поскольку атрибуты друзья, исходящие заявки, входящие заявки, подписки, записи, документы и оцененные записи порождают явную избыточность. Например, для каждой записи пользователя необходимо будет создавать отдельную запись с одинаковым id пользователя. Для избавления от явной избыточности проведем над отношением операции проекции. Получим таблицы 1-7 (см. Приложение).

2.2.2. Группы

Вторая сущность (Группы) не удовлетворяет первой нормальной форме, поскольку атрибуты записи и документы порождают явную избыточность. Например, для каждой записи группы необходимо будет создавать отдельную запись с одинаковым id группы. Для избавления от явной избыточности проведем над отношением операции проекции. Получим таблицы 8-10 (см. Приложение).

Заметим, что отношение "Записи пользователя", созданное ранее, имеет аналогичную логическую структуру. Поскольку каждый идентификатор и группы, и пользователя уникален (не существует пользователя и группы с одинаковым идентификатором), можно хранить записи группы в том же отношении. Преобразуем в соответствии с этим утверждением созданную таблицу "Записи пользователя" для хранения записей группы. Получим таблицу 9 (см. Приложение).

Аналогично преобразованию, проведенному над отношением "Записи", проводится преобразование отношения "Документы пользователя". В результате получается таблица 10 (см. Приложение).

2.2.3. Сообщения

Третья сущность (Сообщения) не удовлетворяет первой нормальной форме, поскольку атрибут документы порождают явную избыточность. Например, для каждого документа, прикрепленного к сообщению, необходимо будет создавать отдельную запись с одинаковым id сообщения. Для избавления от явной избыточности над отношением проводится операция проекции. В результате получаются таблицы 11 и 12 (см. Приложение).

2.3. Приведение базы данных ко второй нормальной форме

Определение: отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и каждый неключевой атрибут функционально полно зависит от первичного ключа.

Отношения Пользователи, Группы, Записи, Сообщения, Классы id остаются без изменений, поскольку их неключевые атрибуты функционально полно зависят от потенциального первичного ключа id (то есть, каждому неключевому атрибуту соответствует одно значение потенциально ключевого).

Для отношений Друзья, Исходящие заявки, Входящие заявки, Подписки, Оцененные записи, Документы, Документы сообщений справедливо, что единственные два их атрибута являются частью составных первичных ключей. Следовательно, неключевых атрибутов нет. Поэтому эти отношения удовлетворяют второй нормальной форме и также остаются без изменений.

2.4. Приведение базы данных к третьей нормальной форме

Определение: отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме и каждый неключевой атрибут нетранзитивно зависит от первичного ключа.

Ни в одном из описанный выше отношений нет транзитивных зависимостей между атрибутами.

Значит, база данных соответствует третьей нормальной форме.

2.5. Проектирование логической модели базы данных

Для визуального отображения спроектированной и нормализованной на предыдущих этапах выполнения курсовой работы структуры базы данных используется сервис app.dbdesigner.net. Выбор программы обусловлен ее быстродействием и относительной простотой получения полной (учебной) лицензии на использование. Результат проектирования сохранен в виде изображения (см. Приложение, Рисунок 1).

2.6. Проектирование физической модели базы данных

По определению из [6] (С. 96), физическая модель базы данных определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне.

Для того, чтобы преобразовать логическую модель в физическую, следует сформировать SQL-запрос (или несколько) системе управления базой данных. В случае проектируемой информационной системы, каждый SQL-запросы должны быть сформулированы на стандартном языке структурированных запросов с учетом особенностей его реализации в Firebird 2.5.

В качестве прикладного приложения на данном этапе разработки базы данных используется IBExpert Version 2020.9.21.1 Personal Edition.

3. Проектирование структуры программы и базовых алгоритмов

Возвращаясь к пункту пояснительной записки 1.2 Описание предметной области, а именно к описанию функциональности социальной сети, можно заметить, что все необходимые для функционирования приложения данные уже описываются физической структурой базы данных. Процессы, которые должны происходить над этими данными, необходимо описать как базовые алгоритмы прикладного приложения.

Для реализации любых алгоритмов взаимодействия с записями базы данных, стоит спроектировать графический пользовательский интерфейс прикладного приложения. Его структура, очевидно, должна соответствовать описанным в 1.2. данным. На рисунке 5 приведен предварительный вариант дизайна интерфейса.

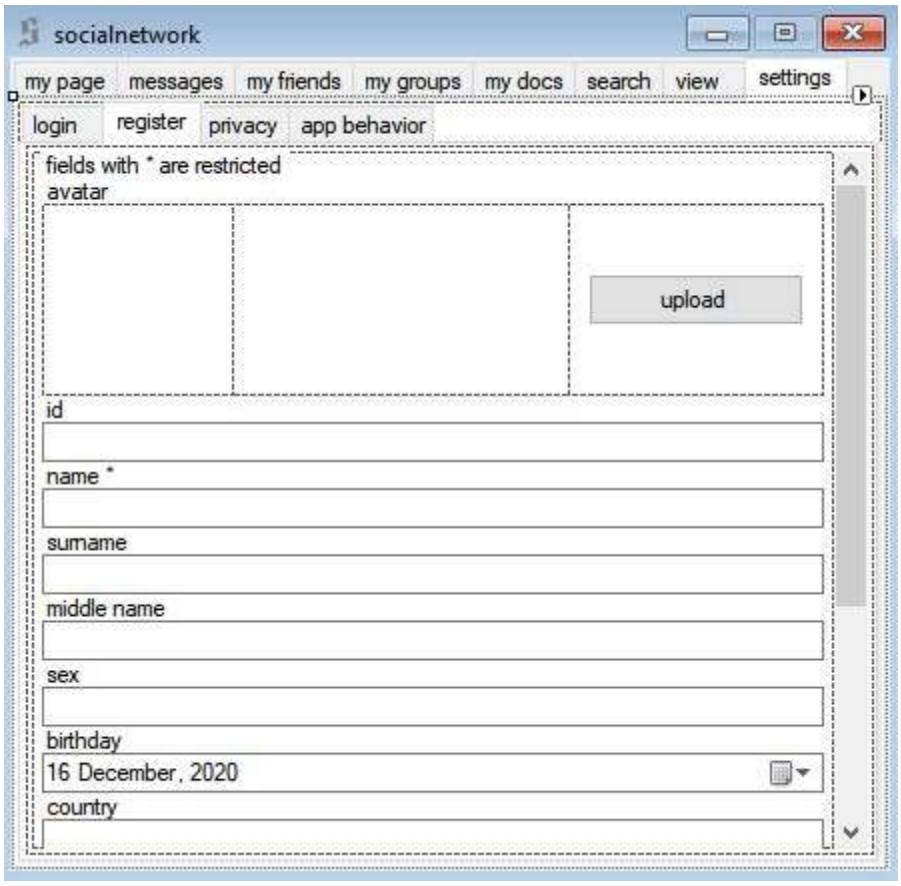

Рисунок 3 – первоначальный (тестовый) вариант интерфейса

4. Программная реализация разработанной структуры и алгоритмов

4.1. Программная реализация запросов, формирующих базу данных

В результате преобразования логической модели базы данных в запросы на языке SQL было получено два .sql файла (см. Приложение, Листинг 1, 2 и 3).

Для рассмотрения принципа работы и построения структуры этих файлов ниже приведен частичный обзор этих листингов, даны краткие комментарии.

Листинг 1 — создание таблицы users (отрывок) (отображение логической сущности Пользователи)

```
CREATE TABLE users
(
    u_id VARCHAR(8) NOT NULL,
    u_name VARCHAR(32) NOT NULL,
    u_surname VARCHAR(32),
    u_middlename VARCHAR(32),
    ...
    PRIMARY KEY(u_id)
);
```

В приведенном выше Листинге 1 можно видеть, что запрос устанавливает однозначное соответствие между атрибутами сущности Пользователи и создаваемыми полями таблицы. Например, атрибут "Имя" здесь реализован следующим образом:

Листинг 2 — атрибут u_name таблицы users

```
u_name VARCHAR(32) NOT NULL
```

Для установления связи между таблицами используются SQL-запросы следующего вида:

Листинг 3 — создание внешнего ключа subs_fk0

```
ALTER TABLE subs
ADD CONSTRAINT subs_fk0
FOREIGN KEY (u_id)
REFERENCES users (u_id);
```

Здесь устанавливается связь между таблицами, отображающими сущности, Пользователи и Подписки с помощью внешнего ключа subs_fk0 таблицы subs.

В результате выполнения приведенных выше запросов далее формируется физическое отображение отношений базы данных.

Затем с помощью программного средства IBExpert Database Designer получается физическая модель создаваемой базы данных. Результат моделирования приведен в виде иллюстрации. (см. Приложение, Рисунок 2).

4.2. Программная реализация алгоритмов

Очевидно, что некоторые структурные элементы предварительно спроектированного графического пользовательского интерфейса (то есть, все вкладки, кроме "settings") должны отображаться только при определенных условиях. Другими словами, если авторизация пользователя не произведена, вкладки, связанные с личными данными пользователя, отображаться не должны. Поэтому, как вспомогательный алгоритм, реализовывался программный интерфейс динамического (в зависимости от действий пользователя) скрытия и отображения вкладок:

Листинг 4 – метод SetMode формы Socialnetwork

```
private int SetMode(string _mode)
{
    switch(_mode)
    {
        case "login":
            tabs.Controls["settingsPage"].Controls["set-
tingsTabs"].Controls.Clear();
            tabs.Controls["settingsPage"].Controls["set-
tingsTabs"].Controls.Add(settingsLoginPage);
            tabs.Controls.Clear();
            tabs.Controls.Add(settingsPage);
            return 0;
            ...
        default:
            return -1;
    }
}
```

При создании объекта класса формы Socialnetwork инициализируется множество вкладок, которые затем добавляются на форму посредством метода SetMode. Выбор предопределенных наборов вкладок осуществляется с помощью аргумента string _mode, причем, если не существует выбираемого предопределенного набора, метод возвращает целое число -1 (индикатор ошибки).

В процессе разработки дизайна был выявлен существенный недостаток тестового пользовательского интерфейса: вкладки без пиктограмм не являются инту-

итивно понятным элементом формы. По этой причине было принято решение создать набор иконок для вкладок, а также изменить расположение элементов переключения страниц с верхнего горизонтального на левое вертикальное.

В результате переосмысления способа реализации функционала прикладного приложения был спроектирован приведенный на рисунках 4-6 графический пользовательский интерфейс.

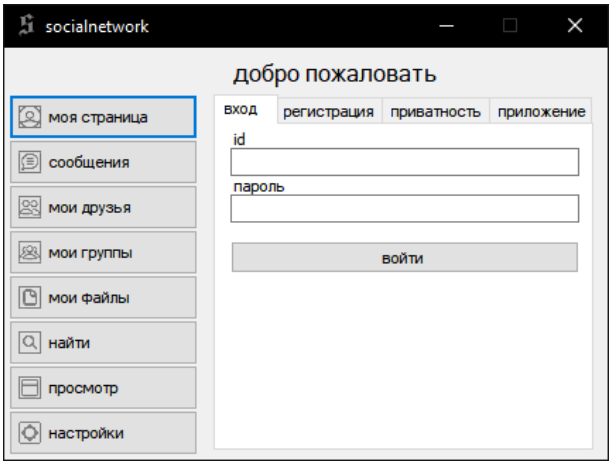

Рисунок 4 – окно приветствия пользователя (после входа в учетную запись)

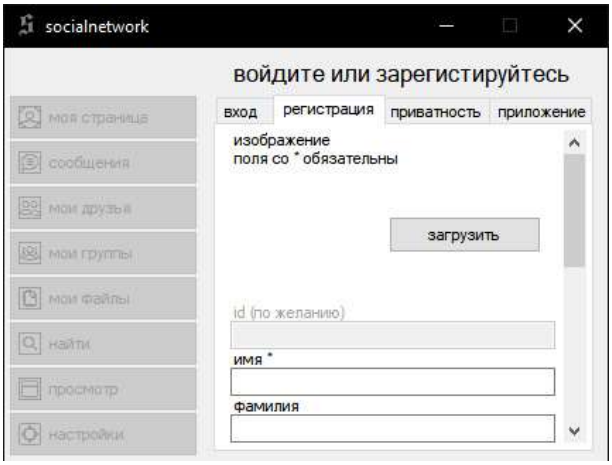

Рисунок 5 – приглашение войти или создать новую учетную запись

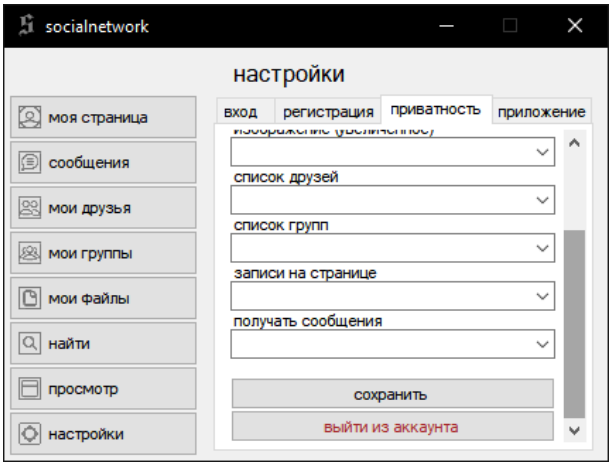

Рисунок 6 – настройки приватности

Методы, реализованные в динамически подключаемой библиотеке fbclient.dll, предоставляют низкоуровневый интерфейс доступа к базе данных. Несложно предположить, что многие запросы к базе данных будут использовать повторяющиеся инструкции. По этой причине такие инструкции инкапсулируются создаваемым классом DB.

Базовый алгоритм, реализующий взаимодействие приложения с базой данных, заключается в создании запросов при помощи созданного класса DB.

Ясно, что полная функциональность приложения требует больших временных затрат на ее имплементацию. Также понятно, что конечная задача, поставленная на проектирование (3. Завершение проектирования рабочим прототипом) не предполагает реализацию полной функциональности создаваемого сервиса.

По этим причинам рабочий прототип обеспечивает лишь базовые возможности, доступные для пользователя социальной сети:

1. Регистрация нового пользователя
2. Авторизация уже зарегистрированного пользователя
3. Настройки пользовательского интерфейса
4. Поиск зарегистрированных в сети пользователей

5. Руководство программиста

5.1. Использование Git

В качестве средства контроля версий был выбран сервис github.com, поскольку приложения, реализующие интерфейс этого сервиса, доступны на многих платформах. В частности, на ОС Windows, при использовании Visual Studio 2019, на момент разработки приложения был доступен весь необходимый функционал Git (git add, git commit, git push, git merge).

Процесс разработки базы данных, прикладного приложения и настоящей пояснительной записки доступен по адресу в интернете [7] в виде 54 коммитов, сделанных в период от 2.10.2020 до 24.12.2020.

Список исходных файлов на момент обращения по ссылке [7] приводится ниже. Пояснения к файлам и каталогам приводятся после списка. Каталоги и файлы, не включенные в этот список, являются генерируемыми средой разработки.

1. queries/*

    3.1. queries/create.sql

    3.2. queries/constraint.sql

    3.3. queries/auto-id-increment.sql

2. app/*

    1.1. app/CMakeLists.txt

    1.2. app/DB.cs

    1.3. app/Program.cs

    1.4. app/Socialnetwork.Designer.cs

    1.5. app/Socialnetwork.cs

    1.6. app/ico/32.ico

    1.7. app/ico/docs.png

    1.8. app/ico/friends.png

    1.9. app/ico/groups.png

    1.10. app/ico/messages.png

    1.11. app/ico/mypage.png

1.12. app/ico/search.png

1.13. app/ico/settings.png

1.14. app/ico/view.png

3. explanatory-note/explanatory-note.md

1. – исходные файлы проекта приложения.

В качестве средства сборки проекта используется CMake. Выбор CMake (в качестве альтернативы стандартному сборщику Visual Studio 2019) обоснован тем, что позволяет создавать платформо-независимый код, который можно загрузить в сервис github.com.

1.1. – информация о порядке сборки проекта приложения.

1.2-1.5 – исходные файлы проекта приложения.

1.6.-1.14 – элементы дизайна интерфейса приложения. 32.ico является контейнером для иконки проекта, реализованной в нескольких вариантах (различия между вариантами заключаются в битности и разрешении изображений).

2. – исходный файл настоящей пояснительной записки. Содержит только текст (без оформления).

3. – запросы к базе данных.

3.1. – создание отношений.

3.2. – установление зависимостей между отношениями.

3.3. – создание генератора уникального идентификатора.

5.2. Приложение

Функциональность приложения реализуется на одной форме.

Переключение между режимами интерфейса осуществляется с помощью модифицированного алгоритма переключения (первоначальный вариант приведен в Листинге 4 (см. 4.2. Программная реализация алгоритмов)).

Работа этого алгоритма обеспечивается возможностью скрытия элементов на форме. В частности, элемент управления Panel позволяет инкапсулировать в себе другие элементы, и при скрытии блокирует к ним доступ. Такое свойство (в соче-

тании с расположением элементов на форме свойством Dock) позволяет эффективно (по скорости) управлять доступными в текущий момент пользователю элементами.

Для того, чтобы приложение получало доступ к серверу баз данных Firebird, и в последствии имело возможность устанавливать подключение к созданной базе данных, был реализован класс-обертка над стандартными (для фреймворка .NET) методами. Его полный листинг приведен в приложении. Ниже частично рассматривается наиболее важная функциональность этого класса.

Листинг 5 – деструктор

```
~DB()
{
    connection.Close();
    connection.Dispose();
}
```

Для того, чтобы созданное в процессе работы программы подключение корректно завершалось, код, связанный с удалением объекта connection, был помещен в деструктор класса-обертки. Таким образом, при сборке мусора garbage collector'ом, соединение будет сначала закрыто, а затем удалено. Очевидно, что сборка мусора произойдет только в том случае, когда объект класса потеряет актуальность.

Листинг 6 – вставка данных

```
public string GetIdInsert(string _table, List<string>
_attributes, List<string> _values, string _returning)
{
    string o = "";
    string sqlCommand = GetCommand(_table,
    _attributes, _values);
    sqlCommand += " returning " + _returning + ";";
    FbCommand command = new FbCommand(sqlCommand,
    connection);
    FbTransaction transaction =
    connection.BeginTransaction();
    command.Transaction = transaction;
    FbDataReader reader = command.ExecuteReader();
    while(reader.Read())
    {
        o += reader.GetInt32(0).ToString();
    }
    transaction.Dispose();
    command.Dispose();
```

Листинг 6 (продолжение)

```
        return o;
    }
```

Для того, чтобы данные с формы можно было направить в базу данных, реализуются методы GetInsert и GetIdInsert. Их функциональность идентична, за исключением того, что GetIdInsert возвращает создаваемый генератором базы данных уникальный идентификатор.

Стоит отметить, что строка-запрос формируется отдельным методом – GetCommand. На вход ему подается название отношения, список атрибутов для вставки и значения этих атрибутов. Возвращаемым значением является строка запроса на языке SQL (Firebird).

В качестве примера вставки приводится обработчик события нажатия на кнопку завершения регистрации нового пользователя.

Листинг 7 – пример вставки данных

```
private void settingsRegisterConfirmButton_Click(object
sender, EventArgs e)
{
    List<string> classAtributes = new List<string> { "ID",
"CLASS" };
    List<string> classValues = new List<string> { "NULL",
"u" };
    string id = scDb.GetIdInsert("ID_CLASSES",
    classAtributes, classValues, "ID");
    string s;
    if (settingsRegisterSexCheck.Checked)
    {
        s = "m";
    }
    else
    {
        s = "f";
    }
    List<string> usersAttributes = new List<string>
    {
        "U_ID",
        "U_NAME",
        "U_SURNAME",
        "U_MIDDLENAME",
...
```

Таким образом, очевидно, что класс-обертка DB сокращает объем повторяющегося кода.

5.3. База данных

Разработка базы данных заключалась в нормализации первоначальных отношений. Результат разработки приведен в виде набора SQL-запросов, распределенных по файлам create.sql (запросы CREATE TABLE), constraint.sql (запросы ALTER TABLE ADD CONSTRAINT), auto-id-increment.sql (запросы GENERATE GENERATOR, SET GENERATOR, CREATE TRIGGER, if-then).

Используемые при именовании полей и отношений сокращения описаны в начале файла create.sql. Например, под полем ud_birthday отношения users подразумевается, что ud = user displaying option; значит, ud_birthday = user displaying option birthday.

6. Руководство пользователя

При использовании приложения сверху на форме отображаются краткие описания доступных в данных момент действий.

Например, при запуске приложения будет отображаться приглашение зарегистрироваться или войти в уже существующую учетную запись (рисунок 7).

Для начала использования системы необходимо:

1. Запустить приложение

2. Перейти на вкладку "регистрация"

3. Ввести личные данные, отмеченные знаком "*"

4. Данные, не отмеченные специальным символом, заполняются по желанию

5. Нажать кнопку "зарегистрироваться"

В случае, если учетная запись уже существует, необходимо:

1. Запустить приложение

2. Перейти на вкладку "вход"

3. Ввести личный идентификатор, указанный вручную или выданный при регистрации

4. Ввести пароль, указанный при регистрации

5. Нажать кнопку "войти"

Результат регистрации и попытки входа будет отображены в заголовке-подсказке.

Для того, чтобы найти пользователя, необходимо:

1. Выбрать режим "найти"

2. Перейти на вкладку "пользователи"

3. Ввести поисковый запрос

4. Нажать кнопку "поиск"

7. Результаты тестирования созданного программного продукта

Тестирование реализованной функциональности ограничивается следующим набором тестов:

1. Создание пользователя

2. Вход в учетную запись пользователя

3. Поиск пользователя

4. Выход из учетной записи пользователя

В результате пробного тестирования были выявлены следующие недостатки системы:

1. Личный идентификатор пользователя создается неочевидным образом. При регистрации он может быть указан вручную, и в этом случае проблема не возникает. Но если идентификатор создается генератором идентификаторов базы данных, то пользователь (легко предположить) не запомнит набор из восьми знаков. И в будущем этот пользователь не будет иметь возможность восстановить его в случае утери.

2. Слишком большое количество записей, которые отображаются на странице регистрации нового пользователя. Понятно, что процесс регистрации (для облегчения понимания интерфейса) должен состоять из двух этапов: введение основных данных и, по желанию, дополнительных. В реализованном же варианте интерфейса оба этих этапа объединены в один. Причем различие между обязательными и необязательными данными заключается в символе "*" после описания обязательных данных. Ясно, что пользователю не очевидно, что все поля заполнять не обязательно.

Таким образом, созданный программный продукт выполняет поставленную на курсовую работу цель, но не предоставляет эффективный графический интерфейс пользователя.

Заключение

В начале курсового проектирования была поставлена цель: создать информационную систему, реализующую сервис социальной сети. Задачи, на которые была разбита эта цель, можно считать выполненными:

1. Задача определения данных, которые хранятся сервисами социальных сетей, подробно рассмотрена и решена в разделе 1. Теперь представляется возможным сформулировать следующий факт:

Данные, хранимые внутри социальной сети, не должны быть легко доступны, поскольку могут (проще или сложнее) быть скомпрометированы.

2. Взаимодействие между данными, которые в простейшем случае хранит социальная сеть, также подробно рассмотрены и описаны. Ясно, что обычные (стандартные) средства систем управления базами данных позволяют описывать и реализовывать движение данных внутри социальной сети, но не обеспечивают безопасность этих данных автоматически.

Другими словами, после завершения курсовой работы очевидно, что ответственность за конфиденциальность хранимых социальной сетью данных полностью лежит на разработчике такой сети.

3. Курсовая работа завершена, поскольку создан прототип приложения, реализующий доступ к базе данных социальной сети.

Список использованной литературы

1. J.A. Obar, S. Wildman Social media definition and the governance challenge: An introduction to the special issue // Telecommunications policy. 2015. №39(9). С. 745-750.

2. В.И. Турканова Приватность в интернете и распространение личных данных подростками // Научные ведомости. Серия Философия. Социология. Право. 2010. №20(91). С. 292-295.

3. Каталог пользователей // vk.com URL: https://vk.com/catalog.php (дата обращения: 23.12.2020).

4. Facebook Reports Second Quarter 2020 Results // fb.com URL: https://investor.fb.com/investor-news/press-release-details/2020/Facebook-Reports-Second-Quarter-2020-Results/default.aspx (дата обращения: 23.12.2020).

5. А.А. Захаров, Р.А. Симаков Базы данных. Муром: Изд.-полиграфический центр МИ ВлГУ, 2008.

6. А.А. Фомин Базы данных. Практикум. Часть 2. Муром: Изд.-полиграфический центр МИ ВлГУ, 2016.

7. Database for a social network on FireBird 2.5 // github.com URL: https://github.com/semenovi/socialnetwork-db (дата обращения: 25.12.2020)

Приложение

Таблица 1 – Пользователи (приведение к первой нормальной форме)

Имя атрибута	Краткое описание
id	Уникальный идентификатор пользователя
Фамилия	Часть имени пользователя
Имя	Часть имени пользователя
Отчество	Часть имени пользователя
Аватар	Адрес файла с картинкой-аватаром в файловой системе
Пол	Пол пользователя, 0 - мужской, 1 - женский
Дата рождения	Дата рождения пользователя
Страна	Страна проживания
Город	Город проживания
Телефон	Личный номер телефона
О себе	Произвольно заполняемая пользователем личная информация
Пароль	Ключевое слово для ограничения доступа к странице
Отображение пола	Отображать ли пол на странице, 0 - видит только владелец страницы, 1 - видят друзья, 2 - видят все
Отображение ДР	Отображать ли дату рождения на странице
Отображение аватара	Отображать ли аватар на странице
Отображение друзей	Отображать ли список друзей на странице
Отображение групп	Отображать ли список групп на странице
Отображение записей	Отображать ли список записей на странице

Таблица 2 – Пользователи (продолжение)

Отображение сообщений	Возможность связаться с пользователем с помощью сообщений
Уведомления	Показывать ли пользователю уведомления о входящих сообщениях, 0 – нет, 1 – да
Звуки	Воспроизводить ли звук уведомления
Содержимое	Отображать ли текст уведомления

Таблица 3 – Друзья (нормализация Пользователи)

Имя атрибута	Краткое описание
id пользователя	id пользователя
id друга	Один друг из списка друзей

Таблица 4 – Исходящие заявки (нормализация Пользователи)

Имя атрибута	Краткое описание
id пользователя	id пользователя
id получателя	Исходящая заявка на добавление в друзья (id пользователя, которому отправлена заявка данным пользователем)

Таблица 5 – Входящие заявки (нормализация Пользователи)

Имя атрибута	Краткое описание
id пользователя	id пользователя
id получателя	Входящая заявка на добавление в друзья (id пользователя, который отправил заявку данному пользователю)

Таблица 6 – Подписки (нормализация Пользователи)

Имя атрибута	Краткое описание
id пользователя	id пользователя
id группы	id группы, на которую подписан пользователь

Таблица 7 – Записи пользователя (нормализация Пользователи)

Имя атрибута	Краткое описание
id пользователя	id пользователя
id записи	Выложенная на странице пользователя запись

Таблица 8 – Документы пользователя (нормализация Пользователи)

Имя атрибута	Краткое описание
id записи	id выложенной на странице пользователя записи
Документ	Документ, прикрепленные к записи на странице пользователя

Таблица 9 – Оцененные записи (нормализация Пользователи)

Имя атрибута	Краткое описание
id пользователя	id пользователя
id записи	Выложенная на странице пользователя или группы запись

Таблица 10 – Группы (приведение к первой нормальной форме)

Имя атрибута	Краткое описание
id	Уникальный идентификатор группы
Название	Имя группы
Аватар	Адрес файла с картинкой-аватаром в файловой системе
id владельца	Уникальный идентификатор пользователя
О группе	Произвольно заполняемая пользователем информация о тематике группы

Таблица 11 – Записи (нормализация Группы)

Имя атрибута	Краткое описание
id	id пользователя или группы
id записи	Выложенная на странице пользователя или группы запись

Таблица 12 – Документы (нормализация Группы)

Имя атрибута	Краткое описание
id записи	id выложенной на странице пользователя или группы записи
Документ	Документ, прикрепленные к записи на стене пользователя или группы

Таблица 13 – Сообщения (приведение к первой нормальной форме)

Имя атрибута	Краткое описание
id сообщения	Уникальный идентификатор сообщения
id отправителя	id пользователя, отправившего сообщение

Таблица 14 – Сообщения (продолжение)

id получателя	id пользователя, которому предназначено сообщение
Текст сообщения	Содержимое сообщения

Таблица 15 – Документы (нормализация Сообщения)

Имя атрибута	Краткое описание
id сообщения	id отправленного сообщения
Документ	Документ, прикрепленные к сообщению

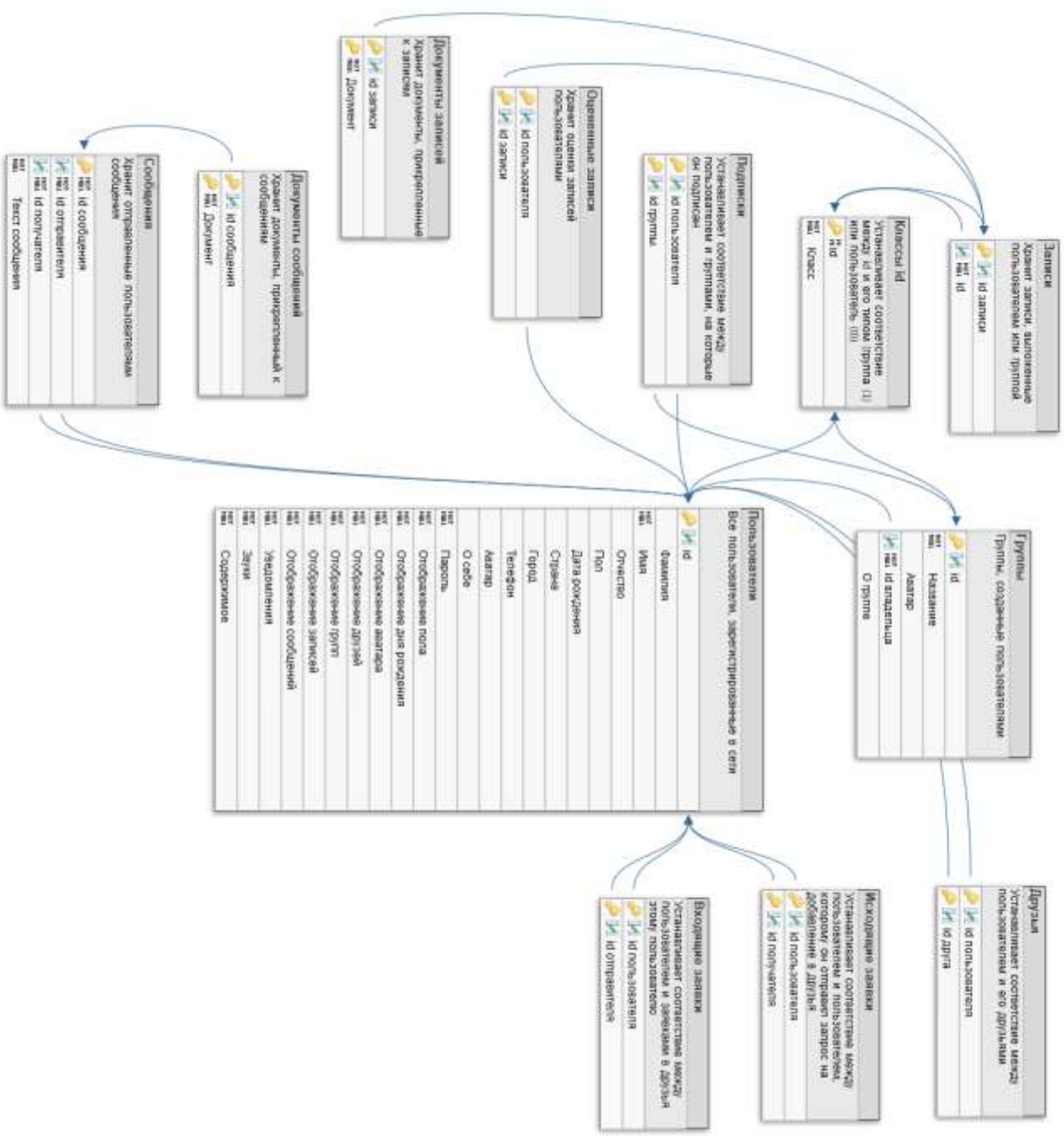

Рисунок 1 – проектирование логической структуры базы данных

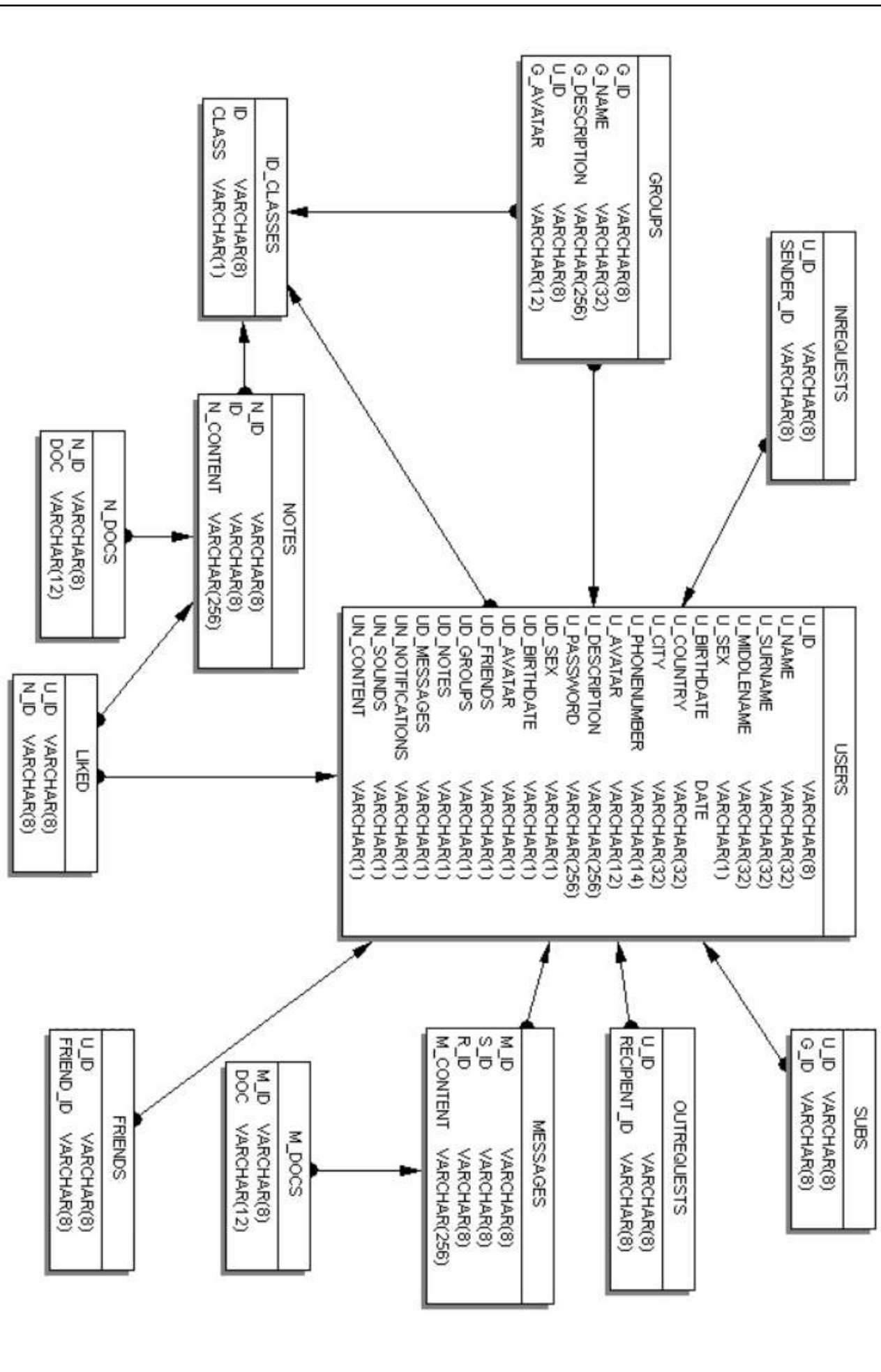

Рисунок 2 – проектирование физической структуры базы данных

Листинг 1 – create.sql

```
-- abbreviation used:
-- u - user
-- g - group
-- n - note
-- m - message
-- s - sender
-- r - recipient
-- ud - displaying option for the user
-- un - notification option for the user

CREATE TABLE users
(
    u_id VARCHAR(8) NOT NULL,
    u_name VARCHAR(32) NOT NULL,
    u_surname VARCHAR(32),
    u_middlename VARCHAR(32),
    u_sex VARCHAR(1),
    u_birthdate DATE,
    u_country VARCHAR(32),
    u_city VARCHAR(32),
    u_phonenumber VARCHAR(14),
    u_avatar VARCHAR(12),
    u_description VARCHAR(256),
    u_password VARCHAR(256) NOT NULL,
    --displaying options
    ud_sex VARCHAR(1) NOT NULL,
    ud_birthdate VARCHAR(1) NOT NULL,
    ud_avatar VARCHAR(1) NOT NULL,
    ud_friends VARCHAR(1) NOT NULL,
    ud_groups VARCHAR(1) NOT NULL,
    ud_notes VARCHAR(1) NOT NULL,
    ud_messages VARCHAR(1) NOT NULL,
    --notifications settings
    un_notifications VARCHAR(1) NOT NULL,
    un_sounds VARCHAR(1) NOT NULL,
    un_content VARCHAR(1) NOT NULL,
    PRIMARY KEY(u_id)
);
CREATE TABLE friends
(
    u_id VARCHAR(8) NOT NULL,
    friend_id VARCHAR(8) NOT NULL,
    PRIMARY KEY(u_id, friend_id)
);
CREATE TABLE outrequests
(
    u_id VARCHAR(8) NOT NULL,
    recipient_id VARCHAR(8) NOT NULL,
    PRIMARY KEY(u_id, recipient_id)
);
CREATE TABLE inrequests
(
```

Листинг 1 (продолжение)

```
    u_id VARCHAR(8) NOT NULL,
    sender_id VARCHAR(8) NOT NULL,
    PRIMARY KEY(u_id, sender_id)
);
CREATE TABLE id_classes
(
    id VARCHAR(8) NOT NULL,
    class VARCHAR(1) NOT NULL,
    PRIMARY KEY(id)
);
CREATE TABLE groups
(
    g_id VARCHAR(8) NOT NULL,
    g_name VARCHAR(32) NOT NULL,
    g_description VARCHAR(256),
    u_id VARCHAR(8) NOT NULL,
    g_avatar VARCHAR(12),
    PRIMARY KEY(g_id)
);
CREATE TABLE notes
(
    n_id VARCHAR(8) NOT NULL,
    id VARCHAR(8) NOT NULL,
    n_content VARCHAR(256),
    PRIMARY KEY(n_id)
);
CREATE TABLE n_docs
(
    n_id VARCHAR(8) NOT NULL,
    doc VARCHAR(12) NOT NULL,
    PRIMARY KEY(n_id, doc)
);
CREATE TABLE subs
(
    u_id VARCHAR(8) NOT NULL,
    g_id VARCHAR(8) NOT NULL,
    PRIMARY KEY(u_id, g_id)
);
CREATE TABLE liked
(
    u_id VARCHAR(8) NOT NULL,
    n_id VARCHAR(8) NOT NULL,
    PRIMARY KEY(u_id, n_id)
);
CREATE TABLE messages
(
    m_id VARCHAR(8) NOT NULL,
    s_id VARCHAR(8) NOT NULL,
    r_id VARCHAR(8) NOT NULL,
    m_content VARCHAR(256) NOT NULL,
    PRIMARY KEY(m_id)
);
```

Листинг 1 (продолжение)

```
CREATE TABLE m_docs
(
    m_id VARCHAR(8) NOT NULL,
    doc VARCHAR(12) NOT NULL,
    PRIMARY KEY(m_id, doc)
);
```

Листинг 2 – constraint.sql

```
ALTER TABLE users
ADD CONSTRAINT users_fk0
FOREIGN KEY (u_id)
REFERENCES id_classes (id);

ALTER TABLE friends
ADD CONSTRAINT friends_fk0
FOREIGN KEY (u_id)
REFERENCES users (u_id);

ALTER TABLE friends
ADD CONSTRAINT friends_fk1
FOREIGN KEY (friend_id)
REFERENCES users (u_id);

ALTER TABLE outrequests
ADD CONSTRAINT outrequests_fk0
FOREIGN KEY (u_id)
REFERENCES users (u_id);

ALTER TABLE outrequests
ADD CONSTRAINT outrequests_fk1
FOREIGN KEY (recipient_id)
REFERENCES users (u_id);

ALTER TABLE inrequests
ADD CONSTRAINT inrequests_fk0
FOREIGN KEY (u_id)
REFERENCES users (u_id);

ALTER TABLE inrequests
ADD CONSTRAINT inrequests_fk1
FOREIGN KEY (sender_id)
REFERENCES users (u_id);

ALTER TABLE groups
ADD CONSTRAINT groups_fk0
FOREIGN KEY (g_id)
REFERENCES id_classes (id);

ALTER TABLE groups
ADD CONSTRAINT groups_fk1
FOREIGN KEY (u_id)
```

Листинг 2 (продолжение)

```
REFERENCES users (u_id);

ALTER TABLE notes
ADD CONSTRAINT notes_fk0
FOREIGN KEY (n_id)
REFERENCES id_classes (id);

ALTER TABLE notes
ADD CONSTRAINT notes_fk1
FOREIGN KEY (id)
REFERENCES id_clases (id);

ALTER TABLE subs
ADD CONSTRAINT subs_fk0
FOREIGN KEY (u_id)
REFERENCES users (u_id);

ALTER TABLE subs
ADD CONSTRAINT subs_fk0
FOREIGN KEY (g_id)
REFERENCES groups (g_id);

ALTER TABLE liked
ADD CONSTRAINT liked_fk0
FOREIGN KEY (u_id)
REFERENCES users (u_id);

ALTER TABLE liked
ADD CONSTRAINT liked_fk1
FOREIGN KEY (n_id)
REFERENCES notes (n_id);

ALTER TABLE n_docs
ADD CONSTRAINT n_docs_fk0
FOREIGN KEY (n_id)
REFERENCES notes (n_id);

ALTER TABLE messages
ADD CONSTRAINT messages_fk0
FOREIGN KEY (s_id)
REFERENCES users (u_id);

ALTER TABLE messages
ADD CONSTRAINT messages_fk1
FOREIGN KEY (r_id)
REFERENCES users (u_id);

ALTER TABLE m_docs
ADD CONSTRAINT m_docs_fk0
FOREIGN KEY (m_id)
REFERENCES messages (m_id);
```

Листинг 3 – auto-id-increment.sql

```
CREATE GENERATOR GEN_USERS_U_ID;
SET GENERATOR GEN_USERS_U_ID TO 0;
CREATE TRIGGER USERS_BI_TR FOR USERS
ACTIVE BEFORE INSERT POSITION 0
AS
BEGIN
    if (NEW.U_ID is NULL) then NEW.U_ID = GEN_ID(GEN_US-
ERS_U_ID, 1);
END
```