Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**»

Институт управления бизнес-процессами

Кафедра «Бизнес информатика и моделирование бизнес-процессов»

**КУРСОВАЯ РАБОТА**

«Создание макета базы данных адвокатской фирмы»

тема

Руководитель \_\_\_\_\_\_\_\_ М.Н. Жукова

подпись, дата инициалы, фамилия

Студент УБ22-08Б, 432215499 \_\_\_\_\_\_\_\_ Д.И.Крашенинников

номер группы, зачетной книжки подпись, дата инициалы, фамилия

Красноярск 2024

**АННОТАЦИЯ**

Настоящий документ предназначен для описания организации макета базы данных адвокатской фирмы с распределённой территориальной структурой. В документе рассматриваются аспекты распределения данных между офисами фирмы, обеспечение целостности и доступности данных для обеспечения надежной работы системы в условиях распределённой сети.

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 2](#_Toc169982389)

[1 Введение в теорию распределённых баз данных 5](#_Toc169982390)

[1.1 Определение и классификация баз данных 5](#_Toc169982391)

[1.2. История и развитие распределённых баз данных 7](#_Toc169982392)

[1.3. Преимущества и недостатки распределённых баз данных 9](#_Toc169982393)

[2 Технологии и языки программирования для баз данных 13](#_Toc169982394)

[2.1 Язык программирования SQL для работы с базами данных 13](#_Toc169982395)

[2.2 Инструменты и платформы для разработки распределенных баз данных СУБД 14](#_Toc169982396)

[2.3 Метод моделирования данных – ER-диаграмма 16](#_Toc169982397)

[3 Описание внутримашинной информационной базы 20](#_Toc169982398)

[3.1 Логистическая структура 20](#_Toc169982399)

[3.2 Физическая структура внутримашинной информационной базы 23](#_Toc169982400)

[4. Описание внемашинной информационной базы 27](#_Toc169982401)

[4.1 Логическая и физическая структуры 27](#_Toc169982402)

[4.1.1 Представление для доступных книг в филиале 27](#_Toc169982403)

[4.1.2 Представление для списка всех читателей и их книг 29](#_Toc169982404)

[4.2 Выполнение запросов 30](#_Toc169982405)

[ЗАКЛЮЧЕНИЕ 36](#_Toc169982406)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 38](#_Toc169982407)

# 

# ВВЕДЕНИЕ

Тема данной курсовой работы посвящена созданию макета базы данных адвокатской фирмы с распределённой территориальной структурой. В современных условиях активного расширения адвокатских практик и включения филиалов в различных географических регионах возникает потребность в разработке эффективных и надежных систем управления информацией. Эти системы должны обеспечивать целостность и доступность данных, что критически важно для обеспечения высококачественных юридических услуг.

Актуальность темы обусловлена необходимостью адвокатских фирм в автоматизации процессов управления и обмена информацией между филиалами. Это позволяет улучшить оперативность обслуживания клиентов, эффективность управления делами и обеспечить точность и актуальность правовой информации. Распределённая структура базы данных также способствует минимизации рисков потери данных и обеспечению их сохранности при возможных сбоях в работе отдельных компонентов системы.

Цель данной курсовой работы заключается в разработке макета базы данных, который будет эффективно поддерживать функционирование распределённой сети адвокатских офисов. Для достижения этой цели необходимо решить следующие задачи:

Изучить современные методы и технологии, используемые для создания и поддержки распределённых баз данных.

Определить подходы к обеспечению целостности и доступности данных в распределённой системе.

Разработать логическую и физическую структуру базы данных, учитывая специфику работы адвокатских офисов.

Заполнить таблицы базы данных актуальной информацией о клиентах, делах и финансовых операциях.

Представить результаты в форме отчёта по курсовой работе.

Объектом исследования является процесс создания и функционирования базы данных адвокатской фирмы с распределённой территориальной структурой. Предметом исследования выступают методы и технологии, применяемые для разработки и поддержки распределённых баз данных в юридической сфере.

Настоящая работа направлена на совершенствование существующих информационных систем адвокатских офисов и обеспечение их более эффективного и надёжного функционирования в условиях географического разделения и децентрализации.

# Введение в теорию распределённых баз данных

## 1.1 Определение и классификация баз данных

База данных (БД) представляет собой организованную совокупность данных, которая предназначена для хранения, обработки и поиска информации. Данные в базе данных структурированы таким образом, чтобы облегчить их управление и обеспечить быстрое выполнение запросов. Базы данных используются в различных областях, таких как бизнес, наука, медицина и другие, для эффективного управления информацией.

Основными характеристиками базы данных являются:

1. Структурированность данных: Данные в базе данных организованы по определённой схеме, что позволяет легко осуществлять поиск, добавление, удаление и модификацию информации.
2. Целостность данных: База данных поддерживает правила целостности, чтобы гарантировать точность и согласованность данных.
3. Безопасность данных: Современные системы управления базами данных (СУБД) обеспечивают механизмы защиты данных от несанкционированного доступа и потери информации.
4. Поддержка многопользовательского доступа: СУБД позволяют нескольким пользователям одновременно работать с данными, обеспечивая при этом изоляцию транзакций и предотвращение конфликтов.

Базы данных можно классифицировать по различным критериям, в том числе по архитектуре и распределению данных. Рассмотрим основные типы баз данных по этим признакам. [2]

1. По архитектуре:

Централизованные базы данных: Все данные хранятся в одном месте, и доступ к ним осуществляется через центральный сервер. Примером такой системы является традиционная реляционная база данных, установленная на одном сервере.

Децентрализованные базы данных: Данные распределены между несколькими серверами или узлами, которые могут находиться в разных географических местах. Это обеспечивает большую надежность и доступность системы, так как сбой одного узла не приводит к потере всех данных.

1. По распределению данных:

Реплицированные базы данных: в таких системах копии данных хранятся на нескольких узлах. Это позволяет увеличить отказоустойчивость и улучшить производительность за счёт параллельного доступа к данным. Примером могут служить системы на основе технологии мастер-слейв репликации.

Разделённые базы данных (шардинг): Данные делятся на несколько частей, каждая из которых хранится на отдельном узле. Это позволяет эффективно распределять нагрузку и управлять большими объёмами данных. Такая архитектура часто используется в крупных веб-приложениях и облачных сервисах.

Гибридные базы данных: комбинируют элементы репликации и шардинга для достижения максимальной производительности и отказоустойчивости. Например, в некоторых системах могут быть реплицированы только критически важные данные, тогда как остальные данные распределены по разным узлам. [6]

Таким образом, классификация баз данных по архитектуре и распределению позволяет выбрать оптимальную модель для конкретных задач и условий эксплуатации. Централизованные базы данных подходят для небольших систем с ограниченным числом пользователей, тогда как распределённые базы данных обеспечивают высокую производительность и надёжность для крупных и географически распределённых систем.

## 1.2. История и развитие распределённых баз данных

Системы управления базами данных (СУБД) прошли долгий путь развития с момента своего появления в середине XX века. Их эволюция связана с развитием вычислительной техники, увеличением объёмов данных и потребностью в более эффективных методах их обработки и хранения.

1. Ранний период (1950-1960-е годы):

В этот период данные хранились в виде файловых систем, что затрудняло их управление и обработку. Каждое приложение имело свою собственную файловую структуру, что приводило к дублированию данных и проблемам с их согласованностью.

1. Иерархические и сетевые модели (1960-1970-е годы):

В 1960-х годах появились первые СУБД, основанные на иерархической и сетевой моделях данных. Иерархические базы данных, такие как IBM Information Management System (IMS), представляли данные в виде дерева. Сетевые модели, такие как Integrated Data Store (IDS), позволяли создавать более сложные связи между записями данных. Эти системы улучшили управление данными, но оставались сложными в использовании и изменении структуры данных.

1. Реляционные базы данных (1970-1980-е годы):

В 1970-х годах Эдгар Кодд предложил реляционную модель данных, которая стала революцией в управлении данными. Реляционные базы данных (РБД), такие как IBM System R и Oracle, позволяли управлять данными с помощью простого и мощного языка SQL (Structured Query Language). РБД обеспечивали большую гибкость и простоту использования по сравнению с иерархическими и сетевыми моделями.

1. Объектно-ориентированные базы данных (1980-1990-е годы):

В 1980-х и 1990-х годах, с развитием объектно-ориентированного программирования, появились объектно-ориентированные базы данных (ОБД). Эти СУБД, такие как ObjectStore и Versant, позволяли хранить сложные данные в виде объектов, поддерживая наследование и полиморфизм. ОБД стали популярными в областях, требующих работы с комплексными данными, таких как САПР и мультимедийные приложения.

1. Пост-реляционные и NoSQL базы данных (2000-е годы):

С началом нового тысячелетия и взрывным ростом объёмов данных и требований к их обработке появились новые типы СУБД, такие как NoSQL базы данных (например, MongoDB, Cassandra). Эти системы были разработаны для обработки больших данных и предоставляли высокую производительность и масштабируемость. NoSQL базы данных поддерживают различные модели данных, включая документо-ориентированные, графовые и колонночные модели. [1, 4]

Развитие распределённых баз данных проходило параллельно с эволюцией традиционных СУБД и связано с необходимостью обеспечения доступности и производительности данных в распределённых системах.

1. Ранние распределённые системы (1970-1980-е годы):

Первые исследования и разработки в области распределённых баз данных начались в 1970-х годах. Эти системы стремились обеспечить доступ к данным, хранящимся на нескольких узлах, и включали такие проекты, как SDD-1 и Distributed Ingres. Основными проблемами в этих системах были поддержание согласованности данных и управление транзакциями в распределённой среде.

1. Коммерческие распределённые базы данных (1990-е годы):

В 1990-х годах начали появляться коммерческие распределённые СУБД, такие как Oracle Distributed Database и IBM DB2 Distributed Edition. Эти системы обеспечивали репликацию данных и управление распределёнными транзакциями, что позволило организациям распределять данные по различным географическим локациям и улучшить отказоустойчивость и доступность данных.

1. Интернет и большие данные (2000-е годы):

С развитием интернета и ростом объёмов данных в 2000-х годах возникла необходимость в новых подходах к управлению распределёнными данными. Появились системы, такие как Google Bigtable и Amazon Dynamo, которые предложили новые модели данных и методы распределённого хранения. Эти системы стали основой для многих современных NoSQL решений.

1. Облачные базы данных и микросервисы (2010-е годы):

В 2010-х годах облачные технологии и микросервисные архитектуры привели к новому этапу развития распределённых баз данных. Облачные СУБД, такие как Amazon Aurora и Google Spanner, обеспечивали масштабируемость и гибкость, необходимые для современных приложений. Микросервисные архитектуры, с другой стороны, потребовали использования распределённых баз данных, способных работать в условиях динамических и масштабируемых систем. [4]

Таким образом, развитие распределённых баз данных отражает потребности современных информационных систем в высокопроизводительных, масштабируемых и отказоустойчивых решениях. Понимание истории и основных этапов их развития помогает лучше оценить текущие технологии и методы, а также перспективы будущих инноваций в этой области.

## 1.3. Преимущества и недостатки распределённых баз данных

Распределённые базы данных (РБД) предлагают множество преимуществ, которые делают их привлекательными для использования в современных информационных системах, особенно тех, которые требуют высокой доступности, масштабируемости и отказоустойчивости.

Масштабируемость:

Одним из главных преимуществ РБД является их способность легко масштабироваться горизонтально. Это означает, что по мере роста объёмов данных или увеличения нагрузки на систему можно добавлять новые узлы (сервера), что позволяет равномерно распределять нагрузку и улучшать производительность.

Повышенная доступность и отказоустойчивость:

Распределённая архитектура обеспечивает высокую доступность данных, так как данные хранятся на нескольких узлах. В случае отказа одного из узлов система может продолжать работать, используя данные с других узлов. Это делает систему более устойчивой к сбоям и авариям.

Географическое распределение данных:

РБД позволяют распределять данные между различными географическими регионами, что снижает задержки доступа к данным для пользователей из разных частей мира. Это особенно важно для глобальных приложений, где низкая задержка доступа к данным критически важна для обеспечения хорошего пользовательского опыта.

Гибкость и адаптивность:

Распределённые базы данных могут адаптироваться к различным требованиям приложений. Они поддерживают разные модели данных (реляционные, документо-ориентированные, графовые и т.д.), что позволяет выбрать оптимальную модель для конкретных задач и типов данных.

Улучшенная производительность:

За счёт параллельной обработки запросов на нескольких узлах распределённые базы данных могут значительно улучшить производительность системы. Это особенно актуально для операций чтения, где запросы могут обрабатываться параллельно на разных узлах. [3]

Недостатки и ограничения

Несмотря на многочисленные преимущества, распределённые базы данных также имеют свои недостатки и ограничения, которые необходимо учитывать при их внедрении и эксплуатации.

Сложность управления:

Управление распределённой базой данных значительно сложнее по сравнению с централизованной системой. Это связано с необходимостью синхронизации данных между узлами, управлением транзакциями в распределённой среде и решением конфликтов данных.

Проблемы согласованности данных:

Обеспечение согласованности данных в распределённых системах является сложной задачей. Существуют различные модели согласованности, такие как строгая согласованность, согласованность по конечному состоянию и т.д., каждая из которых имеет свои преимущества и ограничения. Выбор подходящей модели зависит от требований конкретного приложения.

Задержки и сетевые проблемы:

В распределённых системах задержки в передаче данных между узлами и сетевые проблемы могут негативно сказываться на производительности и доступности системы. Необходимость передачи данных по сети увеличивает время выполнения операций и может привести к временным отказам в доступе к данным.

Затраты на инфраструктуру:

Развертывание и поддержка распределённой базы данных требует значительных затрат на инфраструктуру. Необходимо обеспечить наличие достаточного количества серверов, сетевого оборудования и программного обеспечения для управления системой. Также требуются квалифицированные специалисты для управления и поддержки системы.

Безопасность и управление доступом:

В распределённых системах обеспечение безопасности данных и управление доступом становятся более сложными задачами. Необходимо защитить данные на всех узлах, а также обеспечить безопасный обмен данными между узлами. Это требует дополнительных мер безопасности и контроля доступа. [3, 8]

В итоге, распределённые базы данных предоставляют значительные преимущества в плане масштабируемости, доступности и производительности, однако требуют внимательного подхода к управлению, согласованности и безопасности данных. Понимание этих преимуществ и недостатков позволяет более эффективно использовать распределённые базы данных в различных приложениях и системах.

# 2 Технологии и языки программирования для баз данных

## 2.1 Язык программирования SQL для работы с базами данных

Structured Query Language (SQL) является стандартным языком для работы с реляционными базами данных (РБД). SQL позволяет пользователям создавать, модифицировать и управлять данными, а также выполнять сложные запросы для извлечения информации из баз данных. Он был разработан в 1970-х годах компанией IBM и с тех пор стал де-факто стандартом для реляционных СУБД.

Основные возможности SQL:

* 1. Создание и управление структурами данных: с помощью SQL можно создавать таблицы, индексы, представления и другие объекты базы данных.
  2. Манипулирование данными: SQL позволяет вставлять, обновлять, удалять и извлекать данные.
  3. Управление транзакциями: SQL обеспечивает поддержку транзакций, что позволяет выполнять группы операций атомарно.
  4. Обеспечение безопасности данных: SQL предоставляет механизмы управления доступом к данным с помощью ролей и привилегий.

Реализации SQL могут немного различаться в зависимости от конкретной СУБД. Эти вариации называют диалектами SQL. Вот некоторые из наиболее популярных диалектов:

* 1. MySQL: Одна из самых популярных систем управления базами данных с открытым исходным кодом. Поддерживает собственные расширения SQL.
  2. PostgreSQL: Известен своей строгостью к стандарту SQL и поддержкой сложных запросов и транзакций.
  3. Microsoft SQL Server: Продукт компании Microsoft с собственными расширениями T-SQL (Transact-SQL).
  4. Oracle Database: поддерживает диалект PL/SQL (Procedural Language/SQL), который включает в себя дополнительные возможности для процедурного программирования.

Каждый диалект SQL предлагает уникальные возможности и оптимизации, что позволяет разработчикам выбрать наиболее подходящую СУБД в зависимости от конкретных требований приложения. [1, 7]

Таким образом, SQL базы данных предоставляют мощные возможности для структурированных данных и транзакций.

## 2.2 Инструменты и платформы для разработки распределенных баз данных СУБД

Инструменты и платформы для разработки распределённых баз данных

СУБД (Oracle, MySQL, PostgreSQL и др.)

Системы управления базами данных (СУБД) играют ключевую роль в разработке распределённых баз данных, предоставляя механизмы для хранения, управления и обработки данных. В зависимости от конкретных требований проекта, могут использоваться различные СУБД, каждая из которых имеет свои особенности и преимущества.

Oracle Database является одной из самых популярных и мощных реляционных СУБД, широко используемых в корпоративных системах. Oracle поддерживает множество функций, таких как управление большими данными, репликация и распределённые транзакции, что делает её отличным выбором для распределённых систем. Oracle Real Application Clusters (RAC) позволяет организовать кластерную архитектуру, обеспечивая высокую доступность и масштабируемость.

MySQL – это открытая реляционная СУБД, известная своей простотой использования и высокой производительностью. MySQL поддерживает мастер-мастер и мастер-слейв репликацию, что позволяет создавать распределённые конфигурации. Также MySQL предоставляет возможности для горизонтального масштабирования и балансировки нагрузки, что делает её подходящей для распределённых приложений.

PostgreSQL – это мощная и расширяемая реляционная СУБД с открытым исходным кодом. PostgreSQL поддерживает различные типы данных, сложные запросы и транзакции. Распределённая архитектура в PostgreSQL достигается с помощью расширений, таких как Citus, который позволяет масштабировать базу данных на несколько узлов, обеспечивая высокую производительность и отказоустойчивость.

Microsoft SQL Server – это корпоративная реляционная СУБД, предлагающая множество функций для управления данными, бизнес-аналитики и репликации. SQL Server поддерживает различные виды репликации, включая транзакционную и одноранговую репликацию, что позволяет создавать распределённые системы с высокой доступностью.

Платформы для распределённых систем (Hadoop, Apache Spark и др.)

Для работы с большими объёмами данных в распределённых средах разработаны специальные платформы, такие как Hadoop и Apache Spark, которые предоставляют мощные инструменты для распределённой обработки и анализа данных.

Hadoop – это фреймворк с открытым исходным кодом, который позволяет распределённо обрабатывать большие объёмы данных на кластерах из обычных серверов. Основными компонентами Hadoop являются HDFS (Hadoop Distributed File System) для распределённого хранения данных и MapReduce для распределённой обработки данных. Hadoop обеспечивает горизонтальную масштабируемость и высокую отказоустойчивость, что делает его популярным выбором для работы с большими данными.

Apache Spark – это мощный инструмент для обработки больших данных в распределённых системах, обеспечивающий высокую производительность за счёт использования памяти (in-memory computing). Spark поддерживает различные типы операций, такие как обработка данных в реальном времени, машинное обучение и графовые вычисления. Он может интегрироваться с Hadoop, используя HDFS для хранения данных, и обеспечивает удобный API для разработки на языках Scala, Java, Python и др.

Apache Cassandra – это распределённая NoSQL база данных, разработанная для обработки больших объёмов данных с высокой доступностью и отказоустойчивостью. Cassandra использует архитектуру «мастер-мастер» и обеспечивает горизонтальное масштабирование, что делает её подходящей для использования в распределённых системах.

Google Bigtable – это высокопроизводительная, масштабируемая NoSQL база данных, используемая для работы с большими объёмами данных. Bigtable является основой для таких продуктов Google, как Google Analytics и Gmail, и поддерживает интеграцию с другими инструментами для обработки данных, такими как Apache Hadoop и Apache Spark.

Amazon DynamoDB – это облачная NoSQL база данных, разработанная для обеспечения высокой производительности и масштабируемости. DynamoDB поддерживает автоматическое масштабирование, резервное копирование и восстановление данных, что делает её отличным выбором для создания распределённых приложений в облачной среде.

Таким образом, выбор конкретных инструментов и платформ для разработки распределённых баз данных зависит от требований проекта, включая объёмы данных, требуемую производительность, отказоустойчивость и масштабируемость. Использование правильных инструментов позволяет эффективно управлять данными в распределённых системах, обеспечивая высокую доступность и производительность. [7, 8]

## 2.3 Метод моделирования данных – ER-диаграмма

Схема «сущность-связь» (также ERD или ER-диаграмма) – это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы.

ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса.

ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи – глаголов.

В системе сущность представлена в виде экземпляров. Например, экземпляры сущности «Аэропорт» – аэропорты «Домодедово», «Пулково», «Воронеж».

У сущностей есть атрибуты – характеристики, которые их описывают. Например, атрибутами сущности «Аэропорт» будут код, адрес, номер телефона. Атрибуты есть у каждого экземпляра сущности, но у них разные значения. У аэропортов «Домодедово» и «Воронеж» есть одинаковый атрибут «Адрес», но у каждого из них разное значение этого атрибута.

Собрав все сущности будущего проекта, аналитик выясняет, как они между собой, и составляет ER-модель (сокр. от entity–relationshiсвязаныp модель или модель «сущность-связь»).

В модели есть три типа связей:

1. «Один-к-одному» – один экземпляр сущности связан только с одним экземпляром другой сущности.

Например, пассажир рейса и его место в самолете.

1. «Один-ко-многим» – один экземпляр сущности связан со множеством экземпляров другой сущности.

Например, у одного пассажира может быть несколько единиц багажа, при этом каждая единица багажа может быть связана только с одним пассажиром.

1. «Многие-ко-многим» – множество экземпляров одной сущности связаны со множеством экземпляров другой сущности. Например, аэропорт обслуживает несколько авиакомпаний.

При этом каждая авиакомпания может обслуживаться в нескольких аэропортах.

Для того чтобы построить ER-диаграмму, можно использовать разные нотации. Например, Нотация Чена.

Классическая нотация, которая состоит из простых символов – прямоугольников, овалов и линий. Из-за этого нотацию часто используют для концептуальных моделей, которые презентуют заказчику. Человеку, который далёк от аналитики данных, проще разобраться в понятных диаграммах со знакомыми символами. [6]

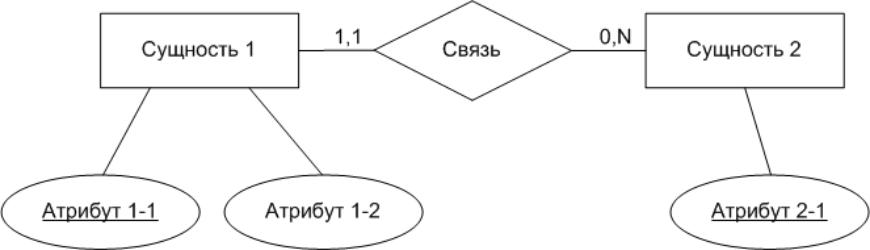


Рисунок 1 – Конструкт, по которому строится ER-диаграмма в нотации Чена

Помимо данного вида существует также популярная нотация Мартина. Её ещё называют «воронья лапка» (от англ. Crow's Foot). Она компактнее нотации Чена, поэтому её используют для построения ER-моделей логического уровня, когда нужно описать в модели все атрибуты сущностей.

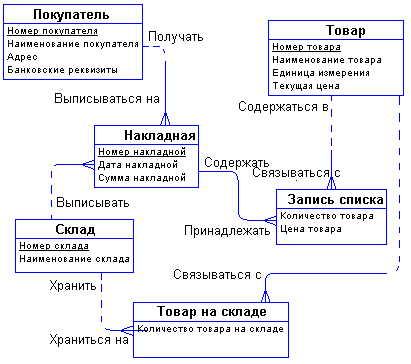


Рисунок 2 – Пример нотации Мартина

Обе нотации, Чена и Мартина, широко используются в области проектирования баз данных и информационных систем. Они помогают разработчикам и аналитикам четко представить структуру данных, их взаимосвязи и основные характеристики, что способствует более эффективному проектированию и оптимизации распределённых баз данных.

# 3 Описание внутримашинной информационной базы

## 3.1 Логистическая структура

Объектом моделирования является автоматизация процессов, протекающих в библиотеке.

В результате исследования объекта были выделены следующие типы сущностей:

1. Клиенты - сильный тип сущности, определяющий адреса, почту, ФИО, телефон.
2. Дела - сильный тип сущности, виды дел, их дата и процесс.
3. Сотрудники - сильный тип сущности, определяющий роль в команде, дату принятия, зарплату и отдел сотрудника.
4. Документы - сильный тип сущности, определяющий название, дату и информацию документа.
5. Суды - сильный тип сущности, определяющий название суда, адрес и телефон.
6. Судебные заседания - сильный тип сущности, определяет дату и время.
7. Платежи - сильный тип сущности, определяющий информацию о платежах.
8. Пользователи и роли - сильный тип сущности, определяющий всю информацию о пользователях.

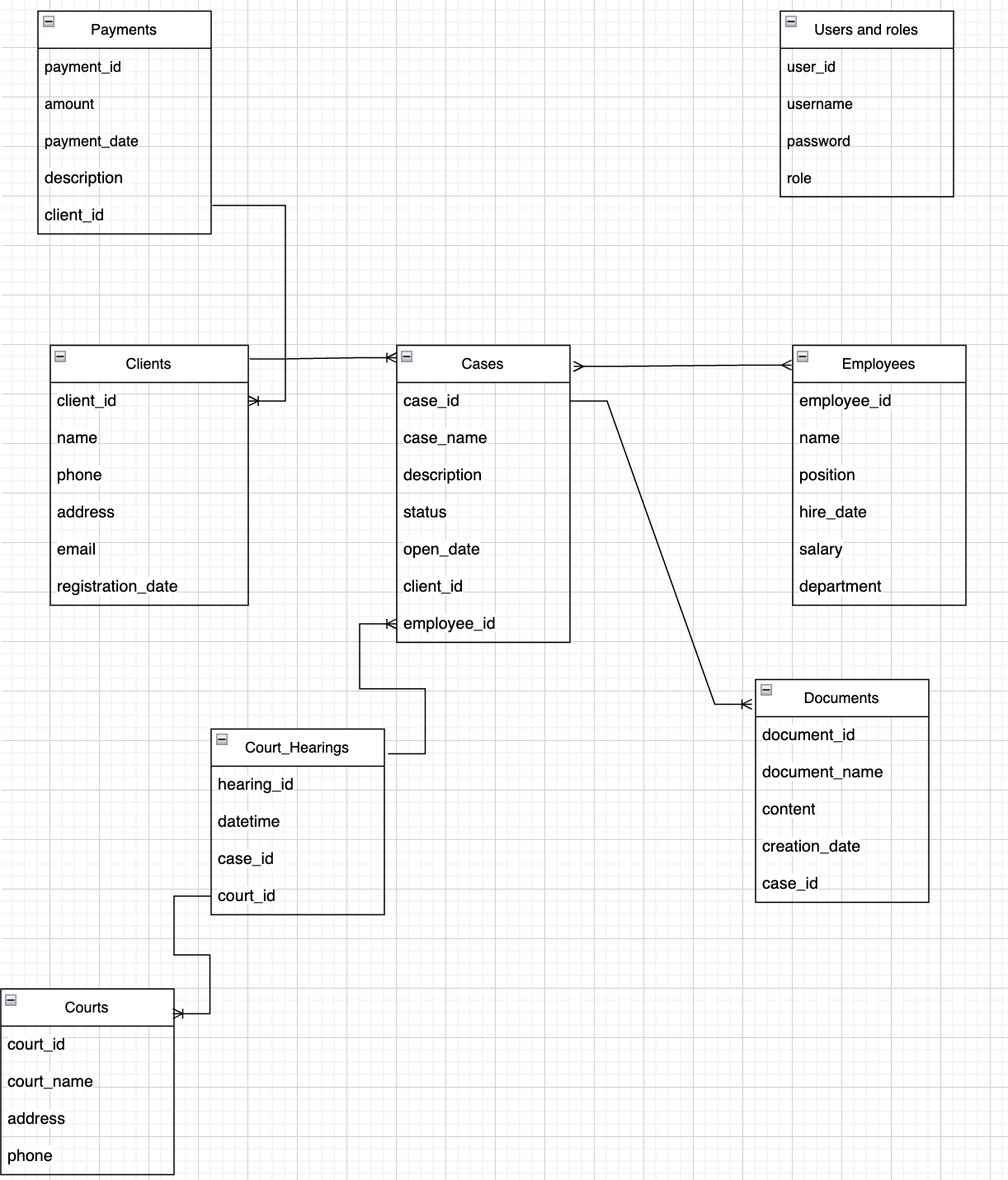


Рисунок 3 – ER-диаграмма

1. Клиенты (Clients):

- Id клиента: Уникальный идентификатор клиента.

- Имя: ФИО клиента.

- Контактный телефон: телефон клиента.

- Адрес: Адрес клиента.

- Эл. почта: электронная почта клиента.

- Дата регистрации: дата регистрации пользователя.

2. Дела (Cases):

- Case\_id: Уникальный идентификатор дела.

- Название дела: Название дела клиента.

- Описание: Информация дела.

- Статус: Статус готовности дела.

- Дата открытия: Дата открытия дела.

- client\_id: внешний ключ к таблице Clients.

- employee\_id: внешний ключ к таблице Employees.

3. Сотрудники (Employees):

-employee\_id: Уникальный идентификатор сотрудника.

- Имя: ФИО сотрудника.

- Должность: Должность сотрудника.

- Дата найма: Число найма сотрудника.

- Зарплата: Зарплата сотрудника.

- Отдел: Отдел работы сотрудника.

4. Документы (Documents):

- document\_id: Уникальный идентификатор документа.

- Название документа: Название данного документа.

- Содержание: Информация документа.

- Дата создания: Число создания документа.

- Case\_id: Внешний ключ к таблице Cases.

5. Суды (Courts):

- Court\_id: Уникальный номер суда.

- Название суда: Полное название суда.

- Адрес: Адрес суд

- Телефон: Контактный телефон суда.

6. Судебные заседания (Court Hearings):

- hearing\_id: Уникальный идентификатор судебного заседания.

- Дата и время: Дата и время судебного заседания.

- case\_id: внешний ключ к таблице Cases.

- court\_id: внешний ключ к таблице Courts.

7. Платежи (Payments):

- payment\_id: Уникальный идентификатор платежа.

- Сумма: Сумма платежа.

- Дата платежа: Дата платежа.

- Описание: Описание платежа.

- client\_id: внешний ключ к таблице Clients.

8. Пользователи и роли (Users and Roles):

- user\_id: Уникальный идентификатор пользователя.

- Логин: Логин пользователя.

- Пароль: Пароль пользователя.

- Роль: Роль пользователя.

## 3.2 Физическая структура внутримашинной информационной базы

Физическая структура внутримашинной информационной базы является ключевым аспектом в современных информационных технологиях. Она определяет организацию данных на уровне аппаратного обеспечения и влияет на эффективность работы информационной системы в целом.

Рассмотрим данную структуру в рамках разрабатываемой базы данных.

1. Таблица Cases (Дела).



Рисунок 4 – Описание таблицы «Cases»

1. Таблица Clients (Клиенты).



Рисунок 5 – Описание таблицы «Clients»

1. Таблица Employees (Сотрудники).

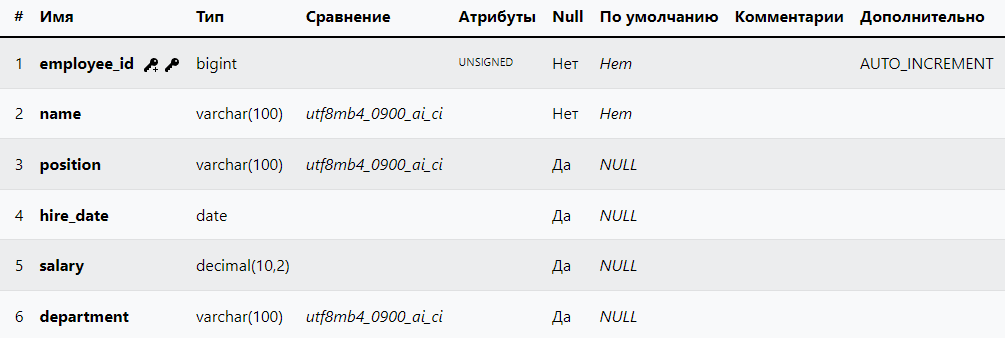


Рисунок 6 – Описание таблицы «Employees»

1. Таблица Documents (Документы)



Рисунок 7 – Описание таблицы «Documents»

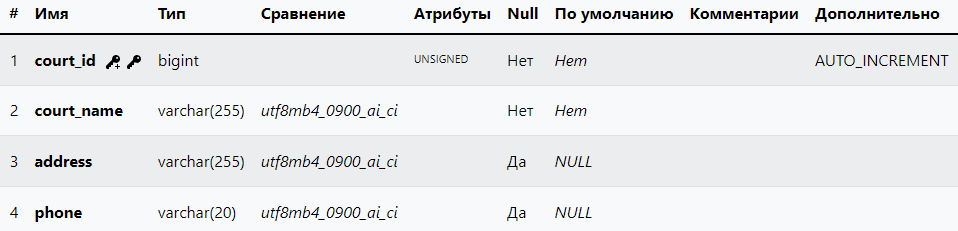
1. Таблицы Courts (Суды).

Рисунок 8 – Описание таблицы «Courts»

1. Таблица Court Hearings (Судебные заседания).



Рисунок 9 – Описание таблицы «Court Hearings»

1. Таблица Payments (Платежи).



Рисунок 10 – Описание таблицы «Payments»

1. Таблицы Users and Roles (Пользователи и роли)

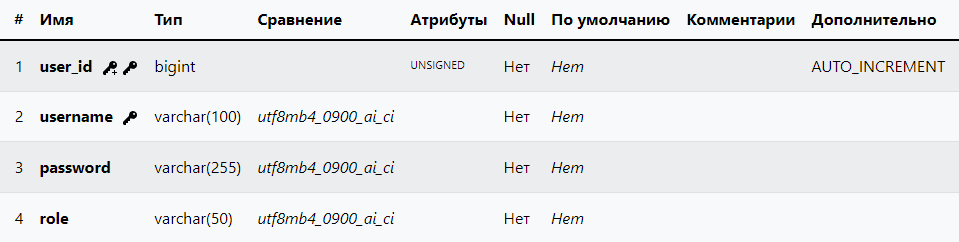


Рисунок 11 – Описание таблицы «Users and Roles»

Информационная база библиотеки поставляется с уже созданной и заполненной базой данных. В случае необходимости изменение и добавление информации в базу данных производится через интерфейс клиентской подсистемы. Все процедуры обслуживания СУБД, включая диагностику работоспособности и резервное копирование данных, должны выполняться в соответствии с регламентом эксплуатации и администрирования СУБД.

# 4. Описание внемашинной информационной базы

## 4.1 Логическая и физическая структуры

В случае необходимости изменение и добавление номенклатуры параметров, определяющих работу библиотеки, в базу данных производится через интерфейс клиентской подсистемы.

По результатам работы базы данных адвокатской фирмы предусмотрено формирование следующих отчетов (представлений) с использованием информационной базы:

1. Представление доступных дел для обработки адвокатами.
2. Представление списка всех клиентов и их текущих дел.
3. Представление информации о клиентах и их контактных данных.

Формирование отчетов осуществляется через интерфейс специализированной подсистемы, с возможностью экспорта данных.

### 4.1.1 Представление доступных дел для обработки адвокатами

Отчет содержит таблицу со следующей информацией:

1. Id платежа;
2. Сумма;
3. Число оплаты;
4. Описание;
5. Имя клиента.

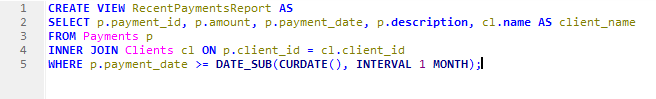


Рисунок 12 – Код представления 1

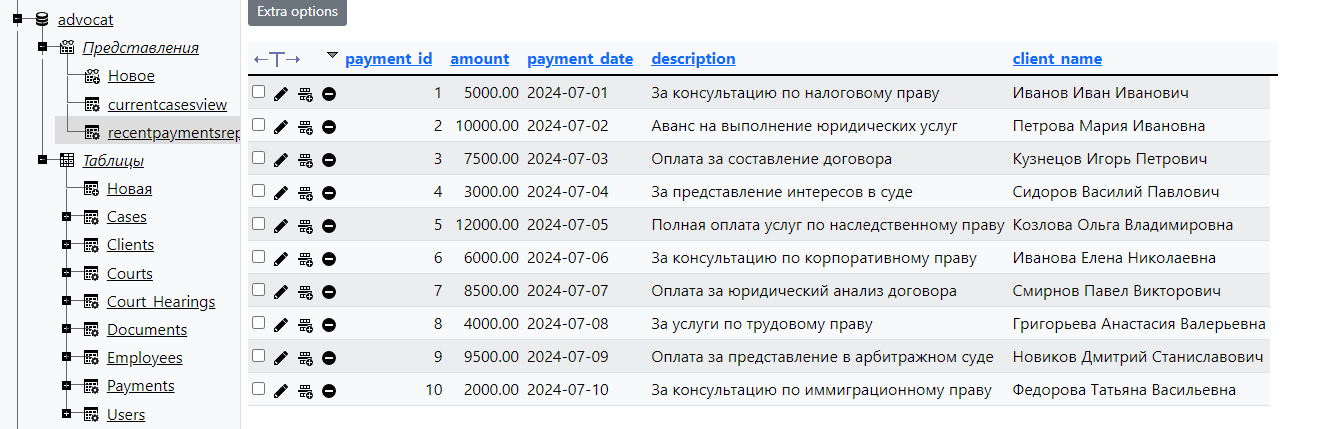


Рисунок 13 – Доступные дела

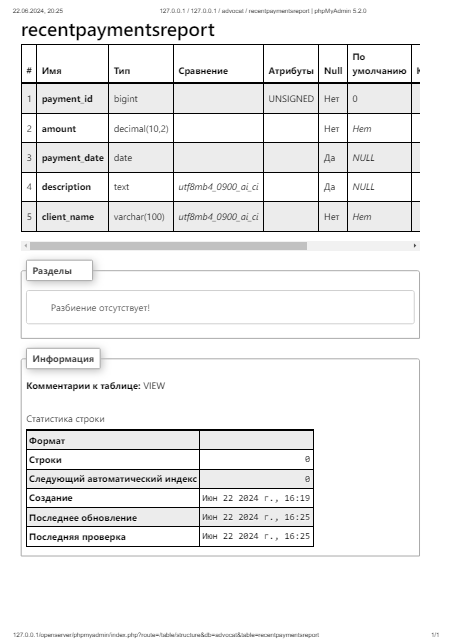


Рисунок 14 – Печатная форма для представления 1

### 4.1.2 Представление списка всех клиентов и их текущих дел

Отчет содержит таблицу со следующей информацией:

1) Id дела;

2) Название дела;

3) Описание дела;

4) Статус дела;

5) Дата дела;

6) Имя клиента;

7) Телефон.

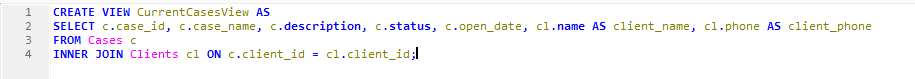


Рисунок 15 – Код представления 2



Рисунок 16 – Список всех клиентов и их дел

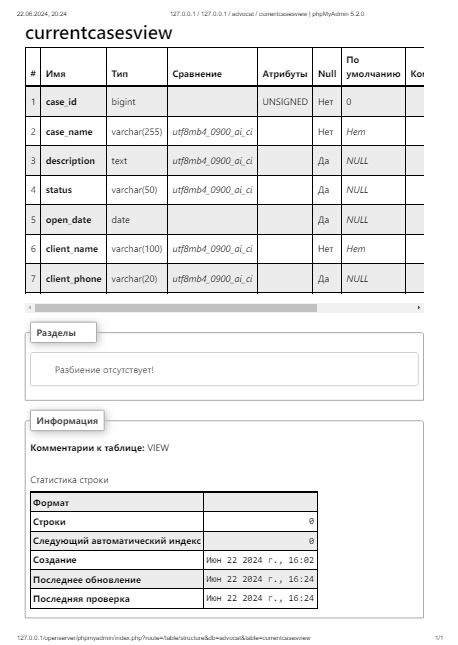


Рисунок 17 – Печатная форма для представления 2

## 4.2 Выполнение запросов

В данном разделе буду применены различные виды запросов к базе данных. Все задачи выполнены в SQL менеджере.

1. Получим информацию о клиентах.



Рисунок 18 – Информация о клиентах

1. Добавим новое дело.



Рисунок 19 – Добавление нового дела, в таблицу Case

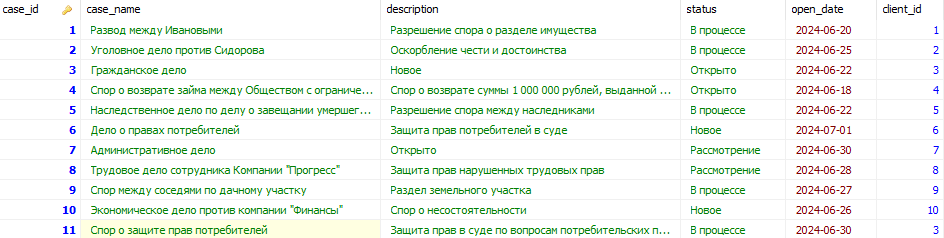


Рисунок 20 – Результат

1. Удалим документ.



Рисунок 21 – Удаление документа

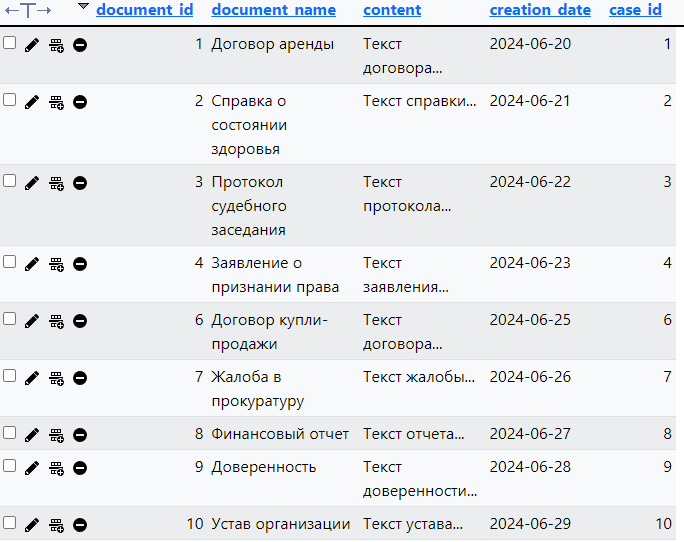


Рисунок 22 – Результат

1. Изменение статуса дела



Рисунок 23 – Изменение статуса дела

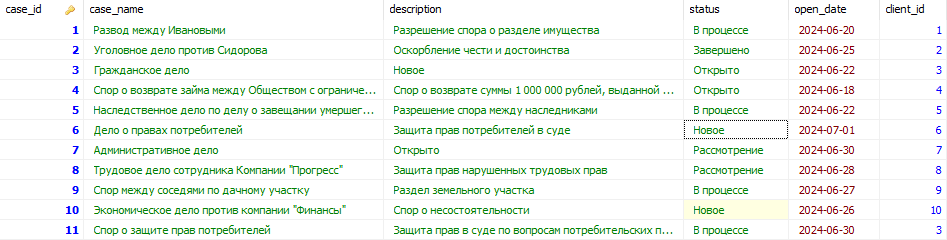


Рисунок 23 – Результат

1. Комбинированный запрос 1 - вычисление суммарной суммы платежей для каждого клиента:

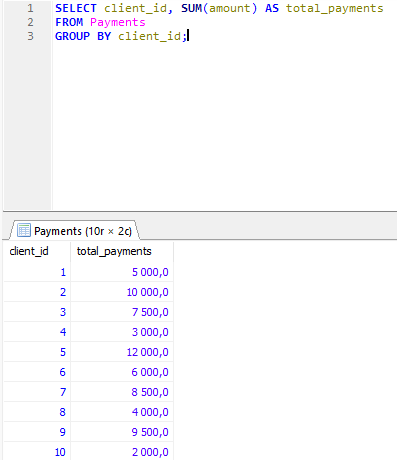


Рисунок 24 – Результат

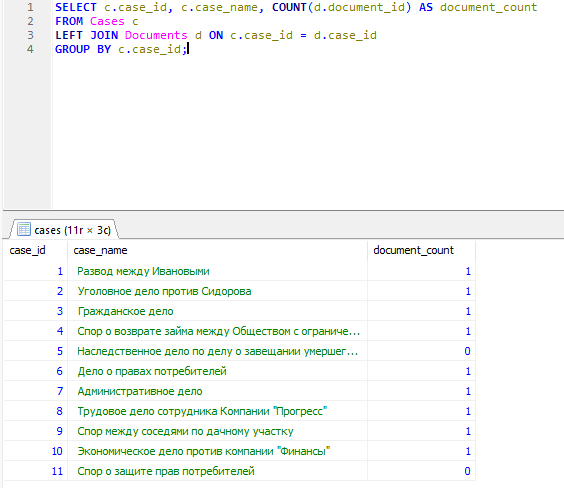
1. Комбинированный запрос 2 - подсчет количества документов по каждому делу: 

Рисунок 25 – Результат

7) Комбинированный запрос 3 - определение средней зарплаты сотрудников в каждом отделе:

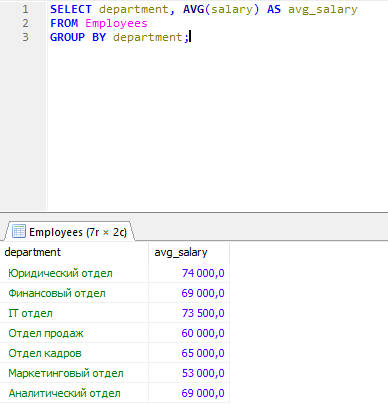


Рисунок 26 – Результат

В результате создания представлений получилось создать три таблицы с нужными данными, которые в последствии могут быть распечатаны. Также после выполнения запросов на добавление, удаление, изменение изначальная база данных претерпела изменения.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы достигнута основная цель - разработка базы данных для адвокатской фирмы с учётом распределённой территориальной структуры. Все поставленные задачи были успешно выполнены, что подтверждает достижение ключевых целей проекта.

В процессе работы были изучены современные методы и технологии, используемые при создании распределённых баз данных. Были определены подходы к обеспечению целостности данных в распределённой среде. Разработана как логическая, так и физическая структура базы данных, учитывающая специфику распределённой системы. Созданы таблицы, отражающие ключевые сущности адвокатской фирмы, такие как клиенты, дела, сотрудники, документы и платежи. Архитектура базы данных была приведена в соответствие с нормальными формами, что обеспечило её целостность и надёжность.

Таблицы базы данных были заполнены тестовой информацией, что позволило проверить корректность структуры и функционирование базы данных. Были добавлены данные о клиентах, делах, сотрудниках, документах и платежах.

Результаты работы представлены в форме отчета по курсовой работе, где подробно описаны процесс проектирования и реализации базы данных, а также её основные характеристики и особенности.

Таким образом, выполнение курсовой работы позволило достичь всех поставленных целей. Разработанный макет базы данных соответствует требованиям адвокатской фирмы, обеспечивая целостность и доступность данных. Это способствует улучшению работы существующих систем управления информацией и повышению эффективности в условиях распределённой структуры, что подтверждает актуальность и практическую значимость данного проекта.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. "Database Design for Law Firms" // Heuristech URL: https://www.heuristech.com/database-design-for-law-firms/ (дата обращения: 20.06.2024).
2. "Database Design Best Practices for Law Firms" // PracticePanther URL: https://www.practicepanther.com/database-design-best-practices-for-law-firms/ (дата обращения: 20.06.2024).
3. "Database Management for Law Firms" // ALA (Association of Legal Administrators) URL: https://www.alanet.org/career/certification/study-resources/database-management-for-law-firms (дата обращения: 20.06.2024).
4. "Designing Law Firm Databases" // Legal Technology URL: https://www.legaltechnology.com/latest-news/designing-law-firm-databases/ (дата обращения: 20.06.2024).
5. "Law Office Management: Database Management" // American Bar Association URL: https://www.americanbar.org/groups/departments\_offices/legal\_technology\_resources/resources/charts\_fyis/database-management/ (дата обращения: 20.06.2024).
6. "Law Firm Database Design: What Every Attorney Should Know" // Ruby Law URL: https://www.rubylaw.com/law-firm-database-design-what-every-attorney-should-know/ (дата обращения: 20.06.2024).
7. "Law Firm Database Systems: An Overview" // Thomson Reuters URL: https://legal.thomsonreuters.com/en/insights/articles/law-firm-database-systems-an-overview (дата обращения: 20.06.2024).
8. "Managing Law Firm Data Effectively" // Legal Support Network URL: https://www.legalsupportnetwork.co.uk/practice-management/best-practice/managing-law-firm-data-effectively (дата обращения: 20.06.2024).