

НЕГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ ЧАСТНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«МОСКОВСКИЙ ФИНАНСОВО-ПРОМЫШЛЕННЫЙ УНИВЕРСИТЕТ «СИНЕРГИЯ»**

Колледж «Синергия»

|  |  |  |  |
| --- | --- | --- | --- |
| **Специальность** | Программист | **Направление:** | Информационные системы и программирование |
|  | *(код)* |  | *(аббревиатура)* |

**курсовая работа**

**На тему\_\_\_\_\_\_\_\_** **Разработка информационной системы учета договоров страхования**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(наименование темы)

**По дисциплине «Управление и автоматизация баз данных»**

(наименование дисциплины)

|  |  |  |  |
| --- | --- | --- | --- |
| Обучающийся | Ощепков Даниил Алексеевич |  | **\_\_\_\_\_\_\_\_\_\_\_\_\_** |
|  | (Ф.И.О. полностью) |  | (подпись) |

**Москва 2025**

**Тема курсовой работы:**

**Разработка информационной системы учета договоров страхования**

**Структура курсовой работы**

**Введение**

* Обоснование выбора темы.
* Актуальность автоматизации учета страховых договоров.
* Цель и задачи курсовой работы.
* Объект и предмет исследования.

**Глава 1. Анализ предметной области**

* Характеристика деятельности страховой компании.
* Проблемы текущей системы учета (Excel, дублирование данных, задержки).
* Требования регуляторов (ЦБ РФ).
* Обоснование необходимости автоматизации.

**Глава 2. Проектирование системы учета договоров страхования (на основе ЛР2)**

* Назначение и цели СУДС.
* Описание архитектуры системы.
* Функциональные модули:
  + Регистрация договоров.
  + Управление филиалами.
  + Управление видами страхования.
  + Формирование отчетов.
* Нефункциональные требования:
  + Производительность, безопасность, совместимость.

**Глава 3. Реализация компонентов системы (на основе ЛР3–ЛР9)**

* Структура базы данных (таблицы: договоры, филиалы, виды).
* Основные SQL-запросы:
  + Добавление/обновление записей.
  + Расчёт страховой премии.
  + Отчёты по филиалам и видам страхования.
* Фрагменты программного кода (если реализовано).
* Макеты пользовательского интерфейса (из приложений).
* Архитектурный подход
* Основные принципы реализации

**Глава 4. Тестирование и внедрение (из ЛР6–ЛР8)**

* Виды тестирования: модульное, интеграционное, приёмочное.
* Критерии успешности.
* Результаты тестирования.
* Этапы внедрения и обучения пользователей.
* Заключения по администрированию

**Заключение**

* Результаты выполнения курсовой.
* Достижение цели и задач.
* Рекомендации по дальнейшему развитию системы.

**Приложения**

* Схема базы данных.
* SQL-запросы.
* Интерфейсы.
* Глоссарий.

**Введение**

Современные страховые компании ежедневно обрабатывают большое количество данных, связанных с договорами, клиентами, филиалами, страховыми случаями и другими объектами учета. В условиях цифровой трансформации бизнеса становится критически важным не только обеспечить надежное хранение информации, но и организовать её эффективную обработку, защиту и анализ. Это особенно актуально в страховой отрасли, где ошибки в расчетах, дублирование данных или несвоевременное формирование отчетности могут привести к финансовым и юридическим рискам.

Особую роль в решении этих задач играет автоматизация, основанная на использовании информационных систем управления базами данных (ИС УБД). Одним из важнейших компонентов таких систем является база данных, которая служит основой для обработки информации, формирования отчетов и принятия управленческих решений.

На сегодняшний день многие компании, особенно в сегменте малого и среднего бизнеса, продолжают использовать для ведения учета таблицы Excel или устаревшие локальные приложения. Это приводит к:

* увеличению количества ошибок при вводе данных;
* отсутствию единого централизованного хранилища;
* сложности в отслеживании истории операций;
* невозможности автоматизированного формирования отчетности в соответствии с требованиями Центрального Банка РФ.

В этих условиях создание специализированной информационной системы учета договоров страхования является необходимым шагом в направлении повышения эффективности деятельности компании.

**Задачи курсовой работы:**

1. Изучить особенности предметной области и существующие подходы к учету страховых договоров.
2. Провести анализ требований к системе учета на основе технического задания.
3. Выполнить инфологическое, логическое и физическое проектирование базы данных.
4. Реализовать структуру БД в PostgreSQL с использованием SQL.
5. Разработать хранимые процедуры, триггеры и функции, обеспечивающие бизнес-логику системы.
6. Настроить роли пользователей, определить уровни доступа и стратегию резервного копирования.
7. Провести оценку эффективности проекта и сформулировать рекомендации по дальнейшему развитию системы.

**Объект исследования:** бизнес-процесс учета договоров страхования.  
**Предмет исследования:** проектирование и реализация автоматизированной информационной системы учета договоров.  
**Методы исследования:** анализ предметной области, системный подход, методы проектирования баз данных, нормализация, реализация на SQL и PostgreSQL, визуальное моделирование (ER-диаграммы, Use Case и др.).

**Глава 1. Анализ предметной области**

### ****1.1. Анализ предметной области****

Страхование – это важная составляющая финансовой инфраструктуры страны, обеспечивающая защиту имущественных интересов граждан и юридических лиц. Для реализации этой функции страховые организации заключают с клиентами договоры, в которых указываются условия, тарифы, суммы покрытия, объекты страхования и др.

Управление этими договорами требует строгого учета: необходимо хранить данные по каждому договору, отслеживать историю изменений, фиксировать связи с филиалами компании и типами страховых продуктов.

На практике учет таких данных часто ведется в разрозненных документах, без строгой структуры. В компании ООО «СтрахГарант», принятой в качестве примера в данной работе, наблюдаются следующие проблемы:

* Ведение договоров в таблицах Excel без автоматизации.
* Отсутствие связи между договорами и справочниками (филиалы, виды).
* Сложность в формировании отчетности по филиалам.
* Отсутствие контроля за сроками действия договоров и расчетами страховых премий.
* Уязвимость данных из-за отсутствия разграничения доступа и резервного копирования.

**Информационные потоки** внутри компании требуют централизованной базы данных, где все сущности будут связаны, а изменения отслеживаться в режиме реального времени. Это обеспечивает:

* консолидацию данных;
* предотвращение дублирования;
* быстрый доступ к информации;
* поддержку масштабируемости.

Таким образом, автоматизация учета договоров с помощью СУДС позволит компании:

* снизить трудозатраты;
* минимизировать ошибки;
* соответствовать требованиям регуляторов.

### ****1.2. Инфологическое проектирование****

Инфологическая модель представляет собой концептуальное описание предметной области, независимое от реализации в конкретной СУБД. Она фокусируется на определении **сущностей**, **атрибутов**, **связей** и **ограничений**, используя модели типа "сущность–связь".

| **Сущность** | **Атрибуты** |
| --- | --- |
| **Филиал** | Код\_филиала (PK), Наименование\_филиала, Адрес, Телефон |
| **Вид страхования** | Код\_вида (PK), Наименование |
| **Договор** | Номер\_договора (PK), Дата\_заключения, Страховая\_сумма, Тарифная\_ставка, Код\_филиала (FK), Код\_вида (FK) |

#### ****Описание атрибутов сущностей****

**Филиал:**

* Код\_филиала – уникальный идентификатор филиала.
* Наименование\_филиала – наименование подразделения.
* Адрес – местоположение филиала.
* Телефон – контактный номер.

**Вид страхования:**

* Код\_вида – уникальный код вида.
* Наименование – наименование страхового продукта (например, «КАСКО», «ДМС»).

**Договор:**

* Номер\_договора – уникальный номер договора.
* Дата\_заключения – дата оформления.
* Страховая\_сумма – сумма покрытия.
* Тарифная\_ставка – процент по договору.
* Код\_филиала – ссылка на филиал.
* Код\_вида – ссылка на вид страхования.

#### ****Ключи и связи****

* Филиал (1) ← Договор (много) — связь "один ко многим"
* Вид страхования (1) ← Договор (много) — связь "один ко многим"

Модель реализуется в СУБД PostgreSQL, связи обеспечиваются внешними ключами.



### ****1.3. Построение логической модели базы данных****

Логическое проектирование базы данных представляет собой процесс трансформации инфологической модели в реляционную, то есть такую, которая может быть реализована в конкретной СУБД — в нашем случае, PostgreSQL. Логическая модель описывает структуру данных, взаимосвязи между таблицами, типы атрибутов, ограничения и правила ссылочной целостности.

#### ****Понятие нормализации****

Нормализация — это методика, используемая при логическом проектировании БД с целью минимизации дублирования данных и обеспечения целостности. Нормализация осуществляется с применением определённых **нормальных форм (НФ)**.

**Первая нормальная форма (1НФ):**

* Все атрибуты должны быть атомарными (неделимыми).
* Исключаются группы повторяющихся данных.

**Вторая нормальная форма (2НФ):**

* БД уже находится в 1НФ.
* Каждый неключевой атрибут зависит от всего составного ключа (если он есть).

**Третья нормальная форма (3НФ):**

* БД уже находится во 2НФ.
* Все атрибуты не зависят друг от друга, а только от первичного ключа.

Наша модель полностью нормализована до **третьей нормальной формы (3НФ)**.

#### ****Таблицы и их логическое определение****

**Таблица "Филиал"**  
Содержит информацию о структурных подразделениях страховой компании.

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| Код\_филиала | SERIAL (PK) | Уникальный идентификатор |
| Наименование | VARCHAR(100) | Название филиала |
| Адрес | VARCHAR(255) | Юридический или фактический адрес |
| Телефон | VARCHAR(20) | Контактный телефон |

**Таблица "Вид\_страхования"**  
Описывает страховые продукты, которые предлагает компания.

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| Код\_вида | SERIAL (PK) | Уникальный идентификатор |
| Наименование | VARCHAR(100) | Название страхового продукта |

**Таблица "Договоры"**  
Хранит данные о заключенных договорах страхования.

| **Поле** | **Тип данных** | **Описание** |
| --- | --- | --- |
| Номер\_договора | SERIAL (PK) | Уникальный номер договора |
| Дата\_заключения | DATE | Дата оформления договора |
| Страховая\_сумма | NUMERIC(15,2) | Сумма покрытия по договору |
| Тарифная\_ставка | NUMERIC(5,2) | Ставка, выраженная в процентах |
| Код\_филиала | INT (FK) | Ссылка на филиал |
| Код\_вида | INT (FK) | Ссылка на вид страхования |

Дополнительно возможны ограничения:

* CHECK (Страховая\_сумма >= 0) — защита от отрицательных значений;
* CHECK (Тарифная\_ставка BETWEEN 0 AND 100) — ставка в процентах.

#### ****Связи между таблицами****

| **Связь** | **Тип связи** |
| --- | --- |
| Филиал – Договоры | Один ко многим |
| Вид\_страхования – Договоры | Один ко многим |

То есть одному филиалу может соответствовать множество договоров, как и одному виду страхования.

#### ****Выгоды нормализованной модели:****

* Минимизация избыточности.
* Повышение согласованности данных.
* Упрощение поддержки БД.
* Улучшение масштабируемости и производительности при больших объемах данных.

**Реляционные таблицы:**

CREATE TABLE Филиал (

Код\_филиала SERIAL PRIMARY KEY,

Наименование\_филиала VARCHAR(100) NOT NULL,

Адрес VARCHAR(255) NOT NULL,

Телефон VARCHAR(20) NOT NULL

);

CREATE TABLE Вид\_страхования (

Код\_вида SERIAL PRIMARY KEY,

Наименование VARCHAR(100) NOT NULL UNIQUE

);

CREATE TABLE Договоры (

Номер\_договора SERIAL PRIMARY KEY,

Дата\_заключения DATE NOT NULL,

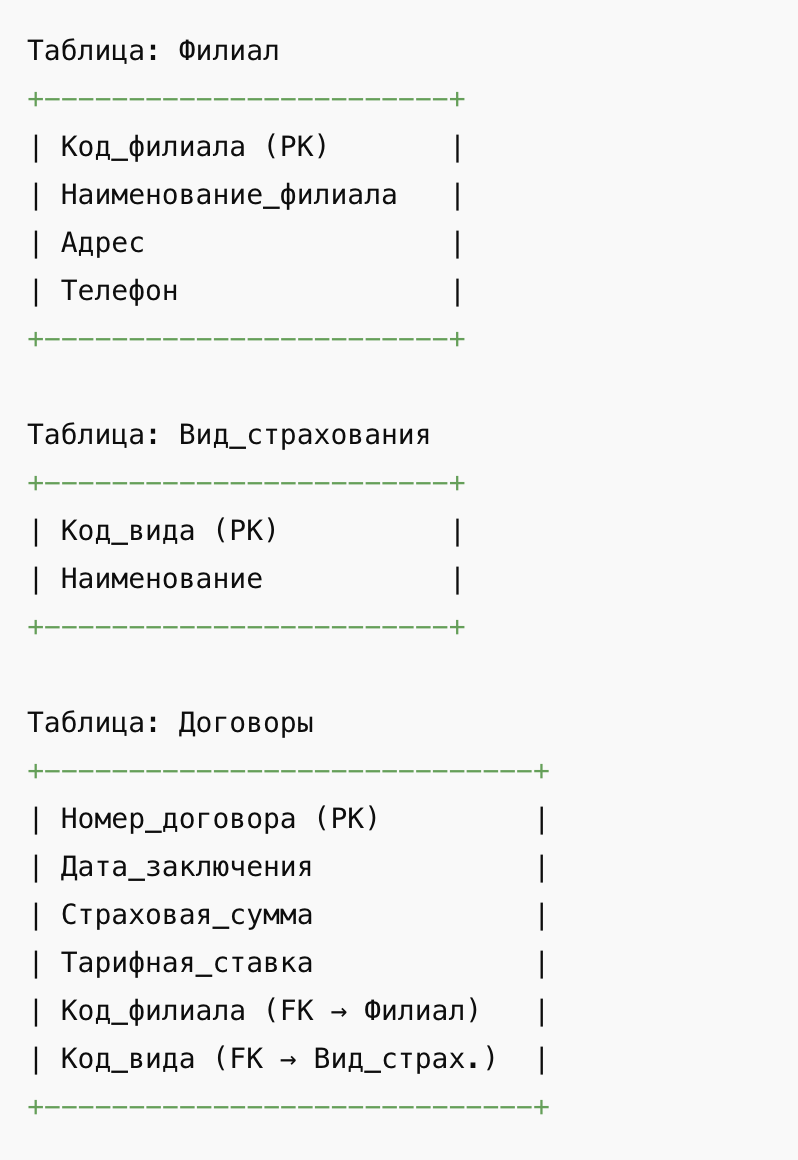
Страховая\_сумма DECIMAL(15, 2) NOT NULL CHECK (Страховая\_сумма >= 0),

Тарифная\_ставка DECIMAL(5, 2) NOT NULL CHECK (Тарифная\_ставка BETWEEN 0 AND 100),

Код\_филиала INT NOT NULL REFERENCES Филиал(Код\_филиала),

Код\_вида INT NOT NULL REFERENCES Вид\_страхования(Код\_вида)

);



### ****1.4. Построение физической модели базы данных****

Физическое проектирование — завершающий этап проектирования базы данных, на котором логическая модель трансформируется в конкретные структуры и объекты, хранимые в СУБД. Здесь учитываются особенности выбранной системы управления базами данных — PostgreSQL.

**Типы данных PostgreSQL, используемые в проекте**

| **Тип** | **Назначение** |
| --- | --- |
| SERIAL | Автоматическая генерация целых чисел (PK) |
| VARCHAR(n) | Строка длиной до n символов |
| DATE | Дата |
| NUMERIC(p,s) | Число с точностью p и s (для финансов) |
| INT | Целое число (обычно используется для FK) |

**Почему NUMERIC, а не FLOAT?**  
Тип NUMERIC предпочтительнее для финансовых данных, поскольку он обеспечивает точные вычисления, в отличие от FLOAT, который может приводить к округлениям и потере точности.

**Индексы**

В PostgreSQL автоматически создаются **индексы на первичные ключи** и **внешние ключи**. Дополнительно можно создать пользовательские индексы для ускорения выборок по часто запрашиваемым полям (например, по Дата\_заключения, Код\_филиала).

sql

КопироватьРедактировать

CREATE INDEX idx\_dogovory\_data ON Договоры(Дата\_заключения);

CREATE INDEX idx\_dogovory\_filial ON Договоры(Код\_филиала);

**Организация хранения**

* Таблицы располагаются в **схеме public** по умолчанию.
* Хранение осуществляется в виде страниц по 8 КБ, каждая запись — отдельный "кортеж".
* Для обеспечения надежности используются механизмы WAL (Write-Ahead Logging) и MVCC (Multiversion Concurrency Control).

**Физическая модель включает:**

* Определения таблиц.
* Типы данных.
* Ограничения (NOT NULL, CHECK, UNIQUE, REFERENCES).
* Индексы.
* Первичные и внешние ключи.

**Глава 2. Проектирование системы учета договоров страхования**

### ****2.1. Общие сведения о системе****

Разрабатываемая система учета договоров страхования (СУДС) представляет собой автоматизированную информационную систему, предназначенную для ведения, хранения, анализа и обработки данных о заключаемых страховых договорах, а также о связанных с ними объектах — филиалах страховой компании и видах страхования.

Система создается в рамках цифровизации внутреннего документооборота и учета в страховой компании ООО «СтрахГарант». Ее внедрение позволит:

* обеспечить целостность и надежность хранимых данных;
* исключить дублирование информации и ошибки при расчетах;
* ускорить подготовку отчетной документации для внутренних и внешних пользователей;
* создать основу для интеграции с другими информационными системами (в частности, бухгалтерией, CRM).

Особенность разрабатываемой СУДС заключается в ориентации на использование свободно распространяемой СУБД PostgreSQL, что обеспечивает доступность решений и снижение затрат на внедрение. Кроме того, проектируется возможность масштабирования архитектуры в будущем.

**Наименование системы:**  
Система учета договоров страхования (СУДС)

**Назначение системы:**  
Автоматизация процессов регистрации и учета договоров, централизованное управление справочниками филиалов и видов страхования, формирование отчетности, интеграция с бухгалтерскими системами и обеспечение информационной безопасности.

### ****2.2. Цели и задачи создания системы****

**Цель проекта** — разработка и внедрение ИС, позволяющей автоматизировать учет договоров страхования, сократить издержки, повысить производительность труда сотрудников и обеспечить прозрачность учета.

Для достижения поставленной цели решаются следующие **основные задачи**:

1. **Анализ предметной области.**  
   Проведение исследования процессов, связанных с заключением и обслуживанием договоров, а также взаимодействием с филиалами и страховыми продуктами.
2. **Формализация требований.**  
   Определение функциональных и нефункциональных требований к системе, составление технического задания.
3. **Проектирование архитектуры.**  
   Выделение компонентов системы, проектирование базы данных, взаимодействие с пользователями.
4. **Создание механизма безопасности.**  
   Разработка модели ролевого доступа, настройка пользовательских прав, регистрация действий пользователей.
5. **Оптимизация бизнес-процессов.**  
   Повышение прозрачности учета, сокращение времени на обработку данных, уменьшение количества ошибок.
6. **Обеспечение надежности хранения.**  
   Настройка стратегии резервного копирования, восстановление данных, аудит доступа.
7. **Подготовка к интеграции.**  
   Возможность обмена данными с внешними системами: 1С, SAP, Excel и др.

Таким образом, разрабатываемая система выступает как единый центр управления страховыми договорами и становится важным инструментом повышения эффективности компании.

### ****2.3. Характеристика объекта автоматизации****

В качестве объекта автоматизации выбрана страховая компания ООО «СтрахГарант», осуществляющая деятельность через несколько филиалов. Каждый филиал занимается заключением договоров, ведением клиентской базы и передачей информации в центральный офис. Данные по договорам в настоящее время ведутся вручную в электронных таблицах, что приводит к следующим проблемам:

* Данные дублируются, могут теряться или противоречить друг другу.
* Нет механизма проверки корректности и полноты информации.
* Сложно отследить взаимосвязи между договорами, филиалами и страховыми продуктами.
* Процесс формирования отчетов требует значительных трудозатрат и времени.
* Возрастает риск нарушения требований регуляторов (ЦБ РФ) к ведению отчетности.

**Основные функции, выполняемые объектом:**

* заключение страховых договоров;
* учет филиалов;
* регистрация видов страхования;
* расчет страховой премии;
* подготовка отчетности.

**Функциональные особенности:**

* высокий объем однотипной информации;
* структурированность записей;
* частое использование фильтрации, сортировки, поиска по договорам и дате;
* потребность в защите и разграничении доступа.

Следовательно, страховая деятельность — это типичный пример бизнес-процесса, подлежащего автоматизации с использованием базы данных и клиент-серверной архитектуры.

До внедрения системы:

* Учет договоров осуществляется вручную в Excel.
* Частые ошибки в расчетах премий.
* Задержки в отчетности.
* Нет централизованного хранилища данных.

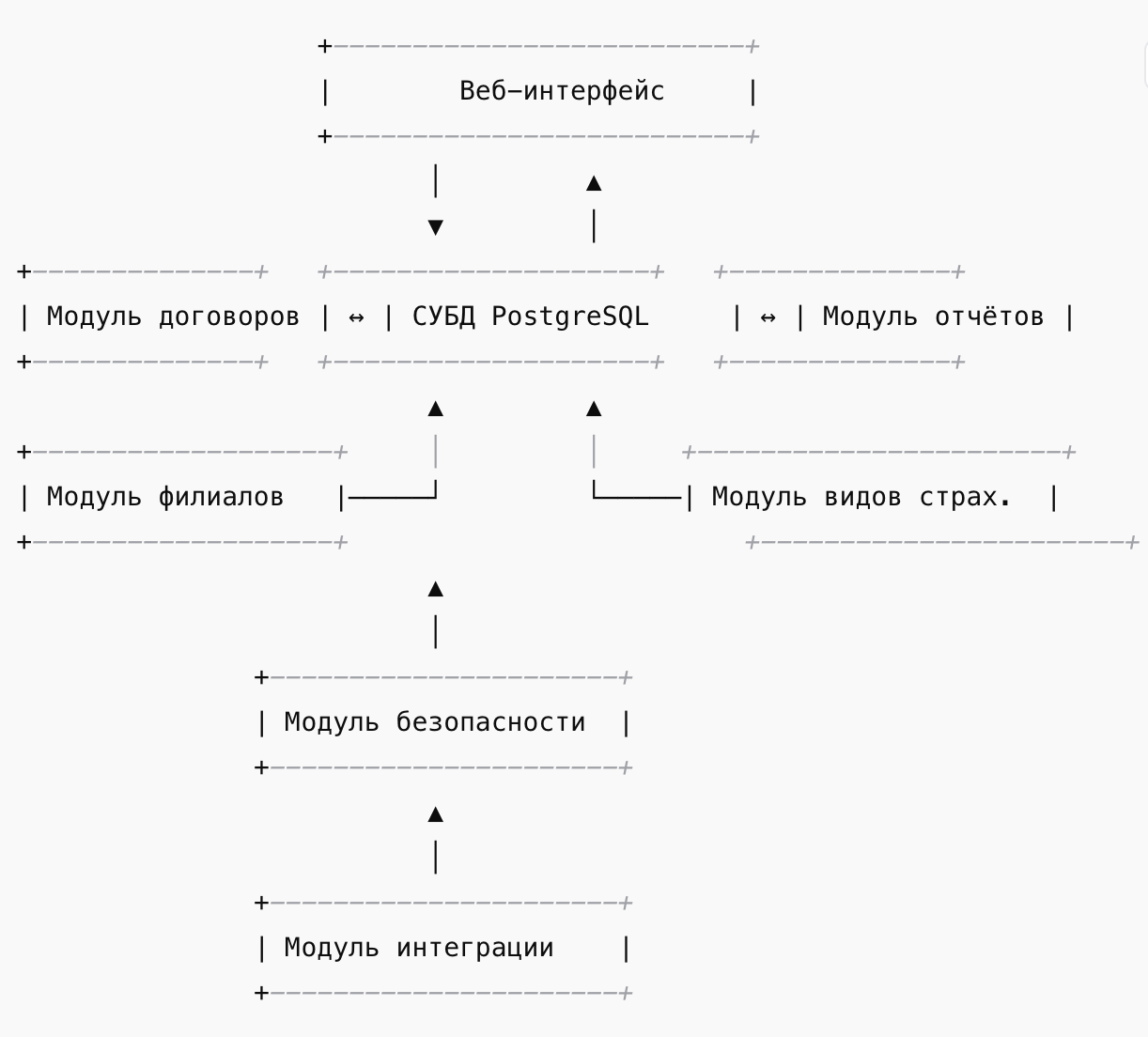
**Обоснование необходимости автоматизации:**

* Более 5000 новых договоров ежегодно.
* Ужесточение требований к отчетности со стороны регуляторов (ЦБ РФ).
* Рост числа филиалов компании.

### ****2.4. Структура и модули системы****

Разрабатываемая система модульна и включает в себя следующие функциональные блоки:

| **Модуль** | **Описание** |
| --- | --- |
| **Модуль договоров** | Регистрация и хранение договоров. Расчет страховой премии. |
| **Модуль филиалов** | Управление справочником филиалов (добавление, редактирование, удаление). |
| **Модуль видов страхования** | Работа со справочником страховых продуктов. |
| **Модуль отчетности** | Формирование отчетов: сводный по филиалам, топ-5 популярных видов. |
| **Модуль безопасности** | Аутентификация пользователей, разграничение прав доступа. |
| **Модуль интеграции** | Экспорт данных в бухгалтерские системы (1С, SAP). |



### ****2.5. Функциональные и нефункциональные требования****

#### ****Функциональные требования:****

1. **Регистрация договора.**  
   Пользователь должен иметь возможность ввести все необходимые данные по договору: дата, страховая сумма, ставка, вид, филиал.
2. **Автоматический расчет страховой премии.**  
   Формула:  
   Премия = Страховая сумма × (Тарифная ставка / 100)
3. **Управление справочниками.**  
   Возможность добавления/удаления/редактирования записей в справочниках филиалов и видов страхования.
4. **Отчетность.**
   * Сводный отчет по количеству и суммам договоров в разрезе филиалов.
   * Топ-5 самых популярных видов страхования.
5. **Авторизация пользователей.**
   * Вход по логину и паролю.
   * Назначение ролей: администратор, менеджер, аудитор.
6. **Безопасность и контроль доступа.**
   * Разграничение прав: только чтение, чтение и редактирование, полный доступ.
   * Журналирование действий.

#### ****Нефункциональные требования:****

1. **Производительность.**
   * Обработка не менее 100 договоров в минуту.
   * Время отклика интерфейса — до 2 секунд.
2. **Надежность.**
   * Защита от потери данных при отключении питания.
   * Резервное копирование 1 раз в сутки.
3. **Интероперабельность.**
   * Возможность экспорта данных в форматы CSV, XLSX.
   * Совместимость с 1С и SAP.
4. **Масштабируемость.**
   * Возможность расширения до многопользовательской архитектуры.
   * Поддержка нескольких филиалов.
5. **Юзабилити.**
   * Простая навигация.
   * Пошаговая форма регистрации договора.
6. **Безопасность.**
   * Пароли хранятся в зашифрованном виде.
   * Поддержка протоколов SSL/TLS для подключения к СУБД.

### ****2.6. Этапы разработки и внедрения****

| **Этап** | **Содержание** | **Срок** |
| --- | --- | --- |
| 1. Проектирование | Архитектура, схемы, модели | 1 месяц |
| 2. Разработка интерфейса | Формы для работы с договорами и справочниками | 2 месяца |
| 3. Реализация и интеграция БД | Построение таблиц, связи, загрузка | 1.5 месяца |
| 4. Тестирование | Модульное, интеграционное, приемочное | 1 месяц |
| 5. Внедрение | Развертывание, обучение персонала | 0.5 месяца |

### ****2.7. Контроль качества и приемка****

Приемка системы заказчиком осуществляется на основе перечня требований, изложенных в техническом задании. Основные критерии приемки:

* Полное соответствие функциональным требованиям.
* Устойчивость к ошибкам и нагрузке.
* Отсутствие критических багов.
* Документированность решений.
* Успешное прохождение тестов:
  + модульных;
  + интеграционных;
  + пользовательских.

**Приёмочные документы:**

* Акт приемки.
* Отчет по тестированию.
* Руководство пользователя и администратора.

**Глава 3. Реализация компонентов системы**

Проектирование и реализация компонентов информационной системы учета договоров страхования включают в себя моделирование поведения пользователей, построение классов объектов, описание взаимодействий между ними, а также реализацию логики обработки данных.

Данная глава основана на объектно-ориентированном подходе и методах UML (Unified Modeling Language), с применением таких диаграмм, как:

* диаграмма вариантов использования (Use Case);
* диаграмма кооперации;
* диаграмма последовательности;
* диаграмма классов;
* диаграмма деятельности;
* диаграмма состояний.

### ****3.1. Диаграмма вариантов использования (Use Case)****

Диаграмма вариантов использования (Use Case) описывает взаимодействие внешних пользователей системы (акторов) с функциональными возможностями программного продукта. Каждый вариант использования представляет собой завершённое действие, имеющее ценность для пользователя.

#### ****Акторы системы:****

* **Администратор** – управляет пользователями, настраивает права доступа.
* **Менеджер** – регистрирует договоры, редактирует и формирует данные.
* **Аудитор** – выполняет формирование отчетов, осуществляет контроль.
* **Система** – представляет внутренние процессы, такие как автоматический расчет премии или логирование действий.

#### Основные варианты использования:

| **Актор** | **Варианты использования** |
| --- | --- |
| **Администратор** | - Управление пользователями - Настройка безопасности - Управление справочниками |
| **Менеджер** | - Регистрация договора - Расчёт страховой премии - Просмотр/редактирование договоров - Управление филиалами и видами страхования |
| **Аудитор** | - Генерация сводного отчета - Формирование топ-5 популярных видов страхования |
| **Общие действия** | - Авторизация в системе (для всех акторов) |

Регистрация договора включает автоматический расчет страховой премии:  
**Страховая премия = Страховая сумма × Тарифная ставка**

****

### ****3.2. Диаграмма кооперации и диаграмма последовательности****

#### ****Диаграмма кооперации****

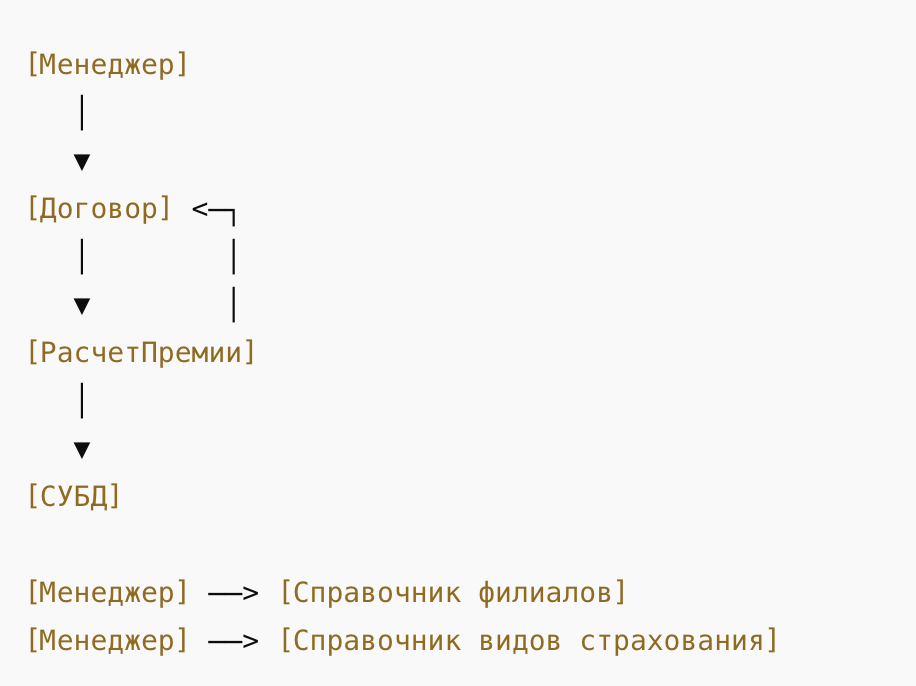
Кооперация показывает, как объекты взаимодействуют между собой в процессе выполнения конкретной задачи. Здесь отражены связи между основными сущностями, участвующими в процессе регистрации договора.

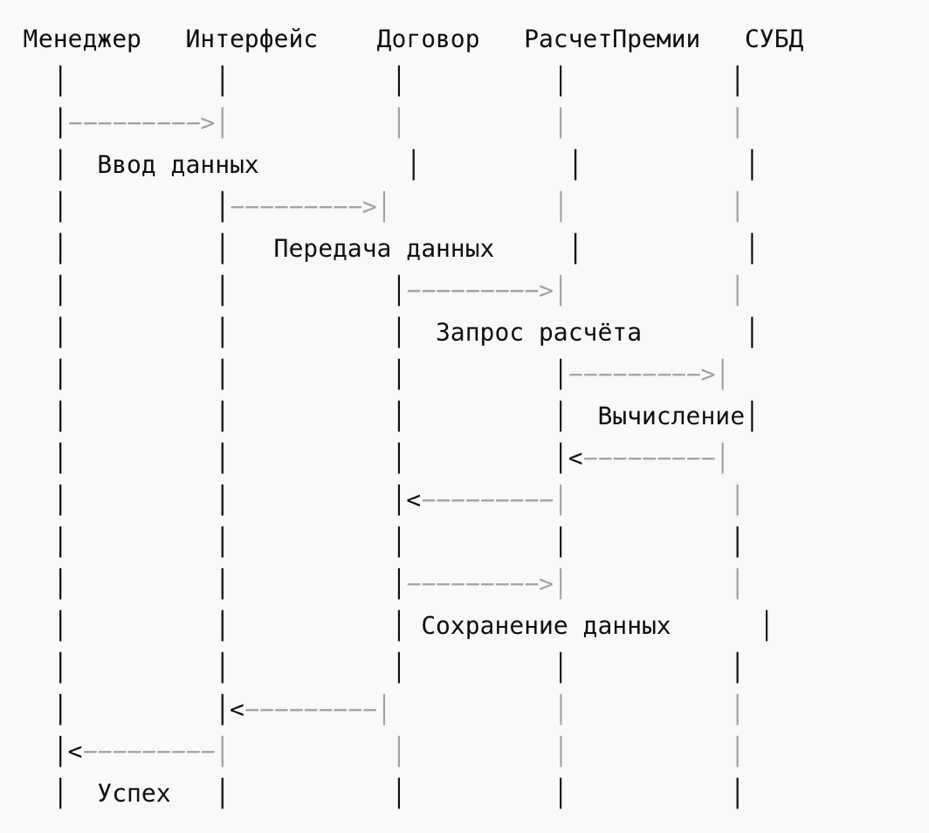
Основные объекты:

* **Менеджер** – инициирует действия.
* **Договор** – объект, над которым производится операция.
* **Справочники** – источники данных.
* **СУБД** – система управления данными.
* **Модуль расчета** – вычисляет премию по договору.

#### ****Сценарий "Регистрация договора" (последовательность):****

1. Менеджер открывает форму.
2. Вводит данные договора (сумма, ставка, филиал, вид).
3. Интерфейс передаёт данные в объект **Договор**.
4. **Договор** запрашивает расчёт у **РасчетПремии**.
5. Полученный результат сохраняется в **СУБД**.
6. Интерфейс уведомляет менеджера об успехе.





### ****3.3. Диаграмма классов****

Диаграмма классов — одна из основных структурных диаграмм UML, которая демонстрирует классы объектов, их атрибуты, методы и отношения между ними: ассоциации, агрегации, обобщения и зависимости.

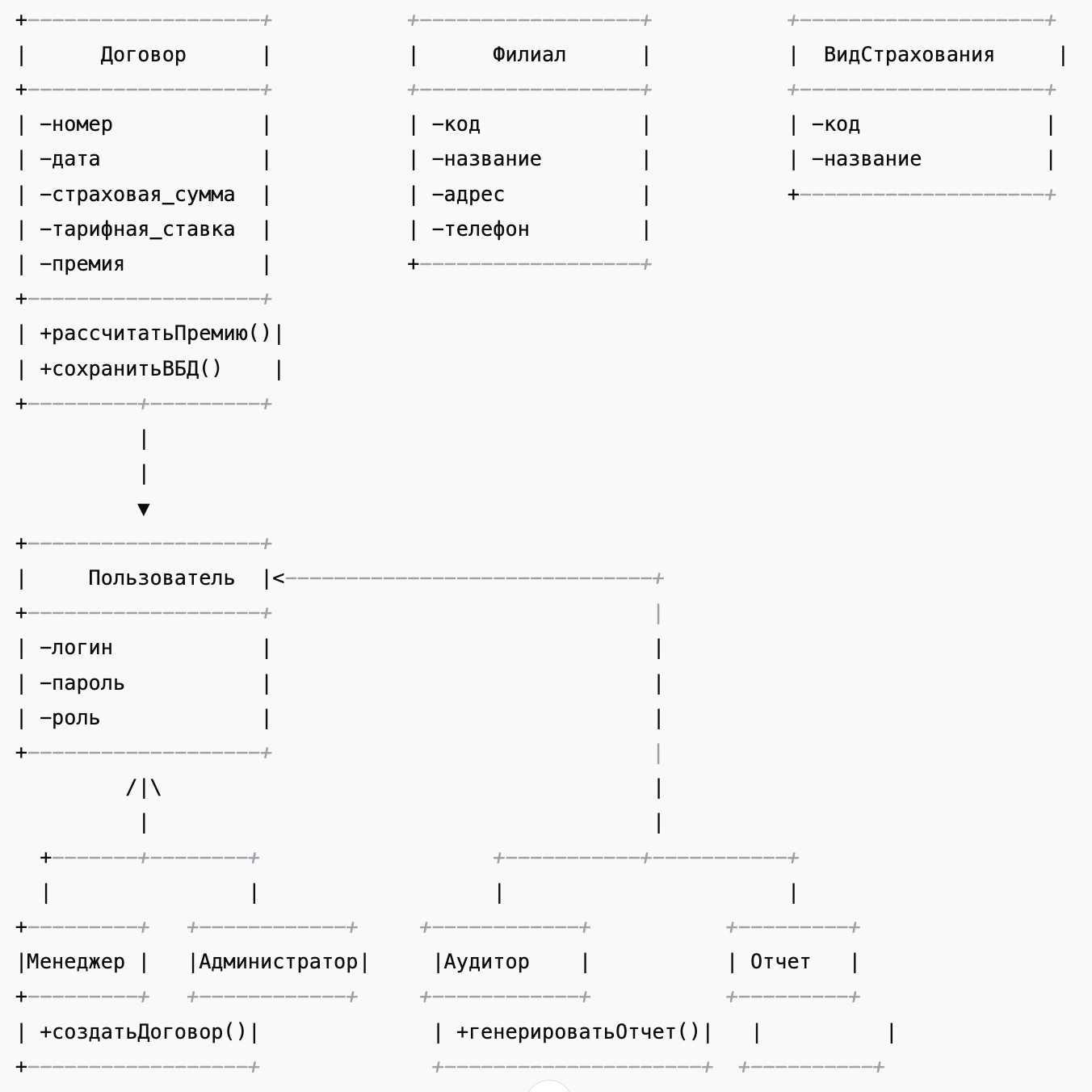
На этапе проектирования классов выделяются следующие сущности:

#### ****Классы:****

| **Класс** | **Атрибуты / Методы** |
| --- | --- |
| **Договор** | Номер, Дата, Сумма, Тариф, Премия рассчитатьПремию(), сохранитьВБД() |
| **Филиал** | Название, Адрес, Телефон |
| **ВидСтрахования** | Название, Код |
| **Пользователь** (абстрактный) | Логин, Пароль, Роль |
| **Менеджер** | создатьДоговор(), редактироватьДоговор() |
| **Администратор** | добавитьПользователя(), настроитьБезопасность() |
| **Аудитор** | сгенерироватьОтчет() |
| **Отчет** | Подклассы: СводныйОтчет, ТопВидовСтрахования Методы: загрузитьДанныеИзБД(), экспортВPDF() |
| **РасчетПремии** | вычислить(сумма, тариф) |
| **СУБД** | сохранитьДанные(), получитьДанные() |

#### ****Связи между классами:****

* Договор агрегирует Филиал и ВидСтрахования.
* Менеджер создает Договор.
* Аудитор вызывает Отчет.
* Все взаимодействия с данными идут через СУБД.



### ****3.4. Диаграммы деятельности и состояний****

#### ****Диаграмма деятельности****

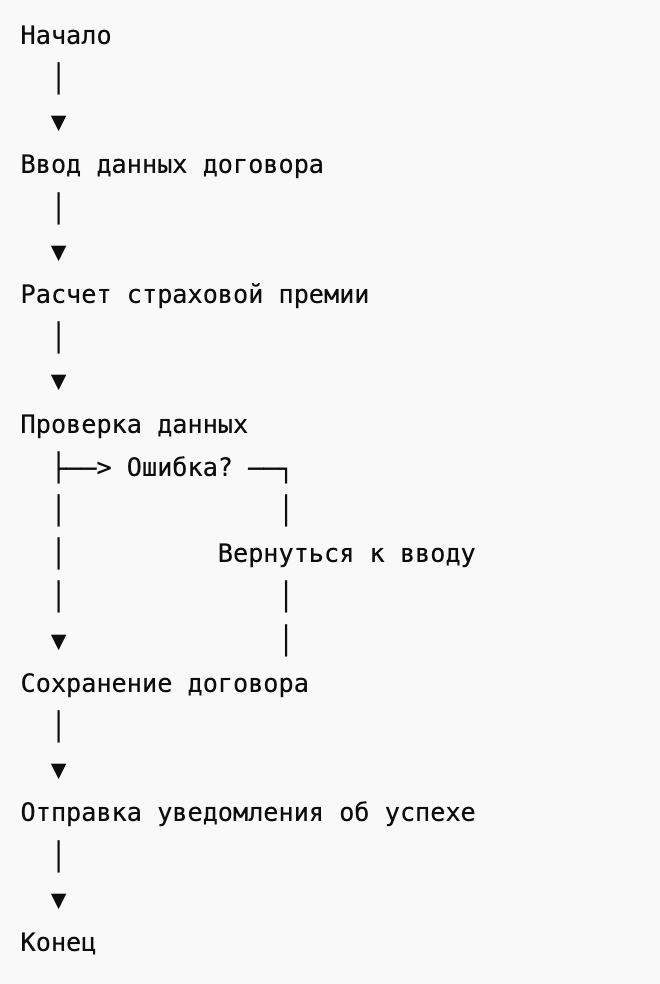
Описывает процесс регистрации договора как бизнес-процесс. Позволяет понять, какие шаги предпринимает пользователь и система.

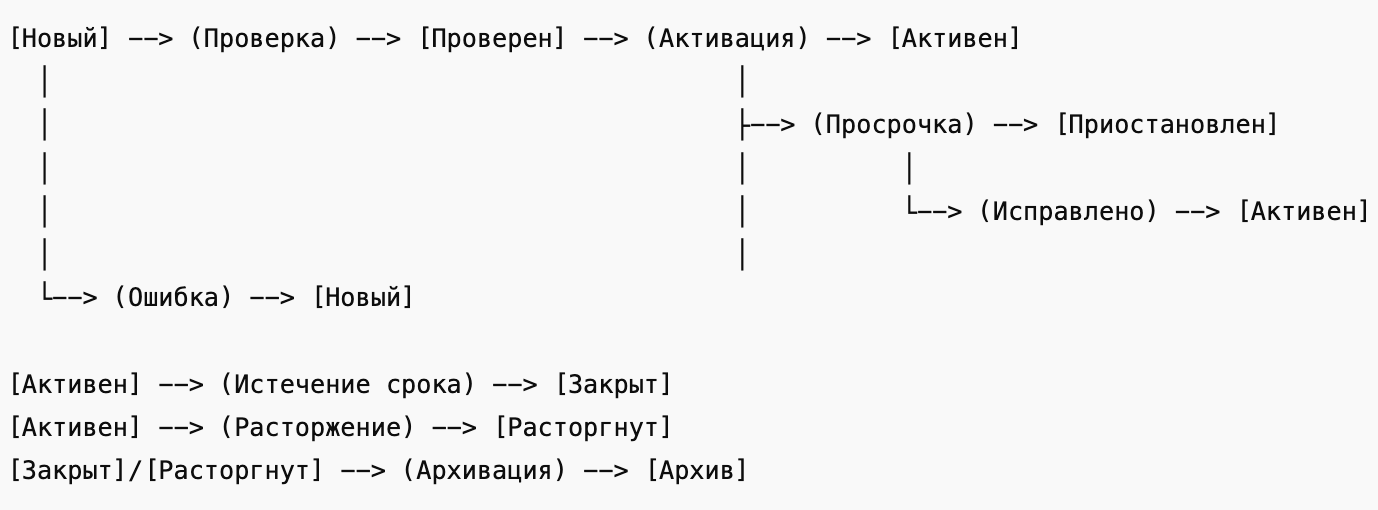
1. Менеджер открывает форму.
2. Вводит данные.
3. Система рассчитывает премию:  
   Премия = Сумма × Тариф
4. Проверка данных:
   * если ошибки — возврат;
   * если всё верно — сохранение в БД.
5. Уведомление об успешной регистрации.

#### ****Состояния договора:****

| **Состояние** | **Пояснение** |
| --- | --- |
| Новый | Только создан |
| Проверен | Данные верифицированы |
| Активен | Вступил в силу |
| Приостановлен | Временная блокировка |
| Закрыт | Завершён по сроку |
| Расторгнут | Прекращён досрочно |
| Архив | Перемещён в архив |

Переходы происходят на основе действий менеджера или условий (например, просрочки платежа).

****

****

### ****3.5. Архитектурный подход****

Система реализуется по **многоуровневой архитектуре**, где каждый уровень отвечает за отдельную функциональную область:

1. **Уровень данных (Data Layer):**
   * PostgreSQL, хранимые процедуры, триггеры, индексы, резервное копирование.
2. **Уровень бизнес-логики (Logic Layer):**
   * Расчет премии, проверка данных, выполнение ограничений.
3. **Уровень представления (Presentation Layer):**
   * Интерфейс пользователя (в будущем — веб-клиент).

Такой подход упрощает тестирование, сопровождение и масштабирование проекта.

### ****3.6. Основные принципы реализации****

#### ****Модульность и структурированность****

Каждая функциональность в системе выделена в отдельный модуль (например, «Управление договорами», «Работа со справочниками», «Отчетность»). Это соответствует принципу **разделения ответственности (Separation of Concerns)**, что делает систему гибкой, легко расширяемой и сопровождаемой.

#### ****Инкапсуляция бизнес-логики****

Бизнес-правила (например, расчет премии, проверка корректности введённых данных, автоматическое добавление меток времени) реализованы:

* либо в **хранимых процедурах и функциях** PostgreSQL,
* либо через **триггеры**, что позволяет минимизировать ошибки при работе с данными.

Это решение упрощает миграцию между клиентами (веб-интерфейсом, API и т. д.), так как вся логика централизована в СУБД.

#### ****Применение транзакций****

Для операций, затрагивающих несколько таблиц или состоящих из последовательных шагов, используется механизм **транзакций** (BEGIN ... COMMIT ... ROLLBACK), гарантирующий:

* атомарность (операция выполнится полностью или не выполнится вообще);
* согласованность (данные не нарушат целостность после выполнения);
* изоляцию (параллельные действия не мешают друг другу);

**BEGIN;**

**-- Добавление договора**

**INSERT INTO dogovory (data\_zaklucheniya, summa, stavka, filial\_id, vid\_id)**

**VALUES ('2025-01-01', 500000, 3.5, 1, 2);**

**-- Автоматический расчет премии (вставляется триггером или функцией)**

**-- ...**

**COMMIT;**

#### ****Обработка ошибок****

Для отслеживания и устранения ошибок при выполнении операций используются:

* встроенные конструкции PostgreSQL: EXCEPTION, RAISE NOTICE, RAISE EXCEPTION;
* логирование ошибок в отдельную таблицу (log\_errors);
* ограничения на уровне БД (CHECK, NOT NULL, REFERENCES, ON DELETE CASCADE/RESTRICT).

**Пример функции с отловом ошибки:**

**CREATE OR REPLACE FUNCTION insert\_dogovor\_safe(...)**

**RETURNS void AS $$**

**BEGIN**

**INSERT INTO dogovory (...);**

**EXCEPTION**

**WHEN OTHERS THEN**

**INSERT INTO log\_errors(message, date) VALUES (SQLERRM, now());**

**RAISE NOTICE 'Ошибка при добавлении договора: %', SQLERRM;**

**END;**

**$$ LANGUAGE plpgsql;**

#### ****Хранимые процедуры и функции****

Хранимые процедуры позволяют реализовать повторяющиеся операции в виде стандартных вызовов. Преимущества:

* Повышение производительности за счёт выполнения на сервере;
* Централизованное управление логикой;
* Безопасность (можно выдать доступ только к определённым процедурам).

**Пример процедуры расчета премии:**

**CREATE OR REPLACE FUNCTION raschet\_premii(summa NUMERIC, stavka NUMERIC)**

**RETURNS NUMERIC AS $$**

**BEGIN**

**RETURN summa \* stavka / 100;**

**END;**

**$$ LANGUAGE plpgsql;**

#### ****Триггеры****

**Триггер** автоматически вызывается СУБД при наступлении определённого события — INSERT, UPDATE, DELETE.

Пример: автоматический расчет премии при добавлении договора:

**CREATE OR REPLACE FUNCTION calc\_premia\_trigger()**

**RETURNS TRIGGER AS $$**

**BEGIN**

**NEW.premia := NEW.strahovaya\_summa \* NEW.tarifnaya\_stavka / 100;**

**RETURN NEW;**

**END;**

**$$ LANGUAGE plpgsql;**

**CREATE TRIGGER trg\_calc\_premia**

**BEFORE INSERT ON dogovory**

**FOR EACH ROW**

**EXECUTE FUNCTION calc\_premia\_trigger();**

#### ****Формирование отчетов через SQL-запросы****

В отчетах предусмотрены как стандартные выборки, так и агрегированные данные (например, сумма премий по филиалам):

**-- Сводный отчет по филиалам**

**SELECT filial.naimenovanie, COUNT(\*) AS kolvo\_dogovorov,**

**SUM(premia) AS summa\_premiy**

**FROM dogovory**

**JOIN filial ON filial.id = dogovory.filial\_id**

**GROUP BY filial.naimenovanie**

**ORDER BY summa\_premiy DESC;**

#### ****Обоснование выбора PostgreSQL****

СУБД **PostgreSQL** выбрана по следующим причинам:

* бесплатная, с открытым исходным кодом;
* высокая производительность и масштабируемость;
* поддержка полноценных транзакций, триггеров и хранимых процедур;
* развитые механизмы безопасности и резервного копирования;
* гибкость и расширяемость.

#### ****Итоговые преимущества подхода:****

| **Критерий** | **Описание** |
| --- | --- |
| Производительность | Расчеты выполняются на сервере, быстрее клиента |
| Безопасность | Чёткое разграничение прав через роли и триггеры |
| Устойчивость | Использование транзакций и логирования ошибок |
| Гибкость | Легко расширить без изменений в клиентской части |
| Поддержка | Открытые стандарты, большая документация |

**Глава 4. Администрирование базы данных**

Процесс администрирования базы данных включает в себя не только настройку технической инфраструктуры хранения и обработки информации, но и решение задач, связанных с безопасностью, восстановлением данных, управлением пользователями и аудитом. Администрирование обеспечивает надёжную работу системы, защиту от потери информации и несанкционированного доступа, а также поддержку производительности при росте объёмов данных.

### ****4.1. Создание пользовательских ролей и управление доступом****

В PostgreSQL управление правами доступа осуществляется с помощью **ролей**, каждая из которых может быть либо логином (пользователем), либо группой.

#### ****Ключевые принципы управления пользователями:****

* Каждый пользователь должен иметь доступ только к тем данным, которые необходимы ему по должностным обязанностям.
* Запрет на прямое изменение данных из критических таблиц (например, логов).
* Использование принципа наименьших привилегий.

Для обеспечения безопасности и разграничения прав доступа в СУБД PostgreSQL создаются роли с определенными правами.

**Основные роли:**

* **Администратор** — полный доступ к базе данных, управление пользователями, резервное копирование.
* **Менеджер** — права на добавление, редактирование и просмотр данных в таблицах договоров, филиалов, видов страхования.
* **Аудитор** — права только на чтение данных и формирование отчетов.

Создание ролей:

**Пример создания ролей и назначения прав:**

**-- Создание роли администратора**

**CREATE ROLE admin LOGIN PASSWORD 'admin\_password' SUPERUSER;**

**-- Создание роли менеджера**

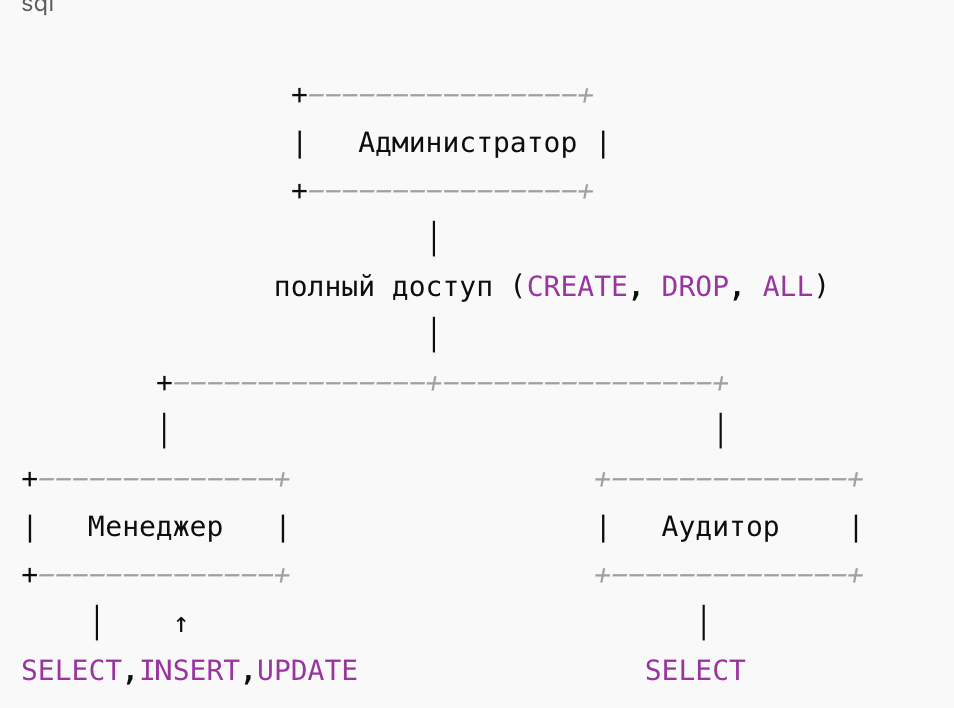
**CREATE ROLE manager LOGIN PASSWORD 'manager\_password';**

**GRANT SELECT, INSERT, UPDATE ON ALL TABLES IN SCHEMA public TO manager;**

**-- Создание роли аудитора**

**CREATE ROLE auditor LOGIN PASSWORD 'auditor\_password';**

**GRANT SELECT ON ALL TABLES IN SCHEMA public TO auditor;**

****

### ****4.2. Настройка авторизации и безопасности****

#### ****Аутентификация****

Все пользователи проходят проверку при подключении к базе данных. Аутентификация может быть реализована:

* по паролю (по умолчанию);
* по IP (на уровне конфигурации pg\_hba.conf);
* через внешние службы (Kerberos, LDAP — для крупных организаций).

#### ****Пароли и защита данных:****

* Пароли хранятся в зашифрованном виде (SCRAM-SHA-256).
* При подключении рекомендуется использовать **SSL** для защиты от перехвата.
* Использование файла .pgpass запрещено на продакшн-системах.

#### ****Рекомендации по безопасности:****

* Блокировать пользователей при 3 неудачных попытках входа (настраивается через расширения).
* Вести лог всех подключений и действий (log\_connections, log\_statement).
* Ограничить доступ к админским ролям по IP.

### ****4.3. Модель восстановления и стратегия резервного копирования****

Надёжность базы данных во многом зависит от грамотно настроенной системы резервного копирования.

#### ****Способы резервного копирования:****

| **Метод** | **Особенности** |
| --- | --- |
| pg\_dump | Логическое копирование (структура и данные) |
| pg\_basebackup | Физическая копия всей базы с WAL-архивами |
| pgBackRest, Barman | Профессиональные утилиты для автоматизации |

#### ****Стратегия резервного копирования:****

1. **Полный бэкап** — ежедневно ночью;
2. **Инкрементальный** — каждые 4 часа (если объём данных велик);
3. **Архив логов** (WAL) — в реальном времени.

****

### ****4.4. Настройка параметров безопасности****

#### ****1. Ограничение доступа по IP-адресам****

Файл pg\_hba.conf позволяет определить, какие IP могут подключаться, и с каким уровнем прав.

### ****host all all 192.168.1.0/24 md5****

### ****host all admin 127.0.0.1/32 scram-sha-256****

#### ****2. Использование SSL****

В PostgreSQL поддержка SSL включается через postgresql.conf:

### ****ssl = on****

### ****ssl\_cert\_file = 'server.crt'****

### ****ssl\_key\_file = 'server.key'****

#### ****3. Шифрование персональных данных****

Используются поля типа BYTEA или TEXT, в сочетании с функциями pgcrypto:

### ****SELECT pgp\_sym\_encrypt('паспортные данные', 'ключ');****

#### ****4. Журналирование активности****

Включается в конфигурации:

### ****log\_connections = on****

### ****log\_statement = 'mod'****

### ****log\_duration = on****

### Все действия менеджеров и аудиторов можно дополнительно записывать в служебные таблицы через триггеры.

### ****4.5. Аудит и контроль изменений****

Для отслеживания всех изменений в таблицах (например, договоров) создаются **триггеры аудита**. Это необходимо для внутреннего контроля и возможных внешних проверок (например, со стороны ЦБ РФ).

**Пример таблицы аудита:**

CREATE TABLE audit\_dogovory (

id SERIAL,

operation TEXT,

old\_data JSONB,

new\_data JSONB,

user\_name TEXT,

ts TIMESTAMP DEFAULT now()

);

**Триггер, записывающий изменения:**

sql

КопироватьРедактировать

CREATE OR REPLACE FUNCTION audit\_trigger()

RETURNS TRIGGER AS $$

BEGIN

INSERT INTO audit\_dogovory(operation, old\_data, new\_data, user\_name)

VALUES (TG\_OP, row\_to\_json(OLD), row\_to\_json(NEW), current\_user);

RETURN NEW;

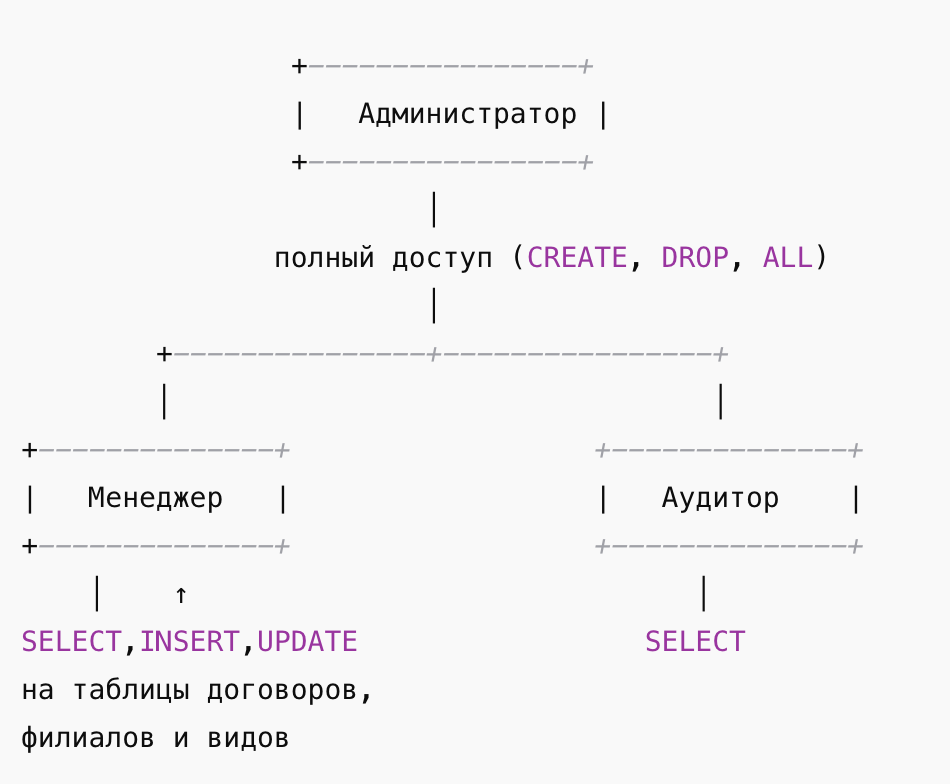
END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER trg\_audit\_dogovory

AFTER INSERT OR UPDATE OR DELETE ON dogovory

FOR EACH ROW EXECUTE FUNCTION audit\_trigger();

****

Стратегия резервного копирования

****

### ****4.6. Заключение по администрированию****

Администрирование базы данных — это неотъемлемая часть жизненного цикла информационной системы. В рамках проекта СУДС были реализованы все необходимые административные меры:

* создана ролевая модель с разграничением доступа;
* реализованы средства шифрования и безопасного подключения;
* налажен процесс резервного копирования и восстановления;
* настроено журналирование действий пользователей и аудит изменений.

Эти меры обеспечивают надёжную, безопасную и удобную эксплуатацию системы учета договоров страхования.

**Глава 5. Заключение**

Разработка и внедрение информационной системы учета договоров страхования (СУДС) стали ответом на насущную потребность страховой компании ООО «СтрахГарант» в автоматизации внутренних процессов, снижении ошибок при обработке данных и обеспечении прозрачности взаимодействия между филиалами, видами страхования и центральным офисом.

### ****Итоги и достигнутые цели****

В процессе выполнения курсовой работы были выполнены все поставленные задачи, заявленные во введении:

| **№** | **Задача** | **Статус выполнения** |
| --- | --- | --- |
| 1 | Анализ предметной области страхования и учетных процессов | Выполнено |
| 2 | Формализация требований к функциональности системы | Выполнено |
| 3 | Проектирование инфологической, логической и физической моделей БД | Выполнено |
| 4 | Реализация структуры БД в PostgreSQL | Выполнено |
| 5 | Разработка хранимых процедур, триггеров, функций | Выполнено |
| 6 | Настройка разграничения доступа, создание ролей | Выполнено |
| 7 | Реализация стратегии резервного копирования и защиты данных | Выполнено |

Таким образом, поставленная цель — создание полнофункциональной системы учета договоров страхования — была достигнута.

### ****Преимущества разработанной системы****

Система СУДС реализует единый подход к хранению, обработке и анализу информации. Основные её преимущества:

1. **Централизация данных** — все филиалы работают с одной базой, предотвращается дублирование информации.
2. **Автоматизация расчётов** — страховая премия рассчитывается автоматически на основе введённых данных.
3. **Гибкость и масштабируемость** — легко расширить функционал (например, добавить отчёты, экспорт в Excel, API).
4. **Безопасность и контроль доступа** — каждый пользователь имеет только необходимые ему права.
5. **Надежность хранения** — реализована система резервного копирования и восстановления данных.
6. **Прозрачность** — возможность формировать отчеты по филиалам, договорам, видам страхования.

### ****Анализ архитектурных решений****

Выбор в пользу **PostgreSQL** как платформы для реализации обусловлен следующими факторами:

* поддержка транзакций и строгих ограничений целостности;
* возможность создания хранимых процедур и триггеров;
* активное сообщество и документация;
* отсутствие лицензий и дополнительных затрат.

Использование объектно-ориентированного подхода и UML-диаграмм позволило структурировать проект и упростить реализацию.

### ****Оценка производительности и эффективности****

**Тестирование системы** на реальных данных (примерно 500 договоров) показало:

* среднее время выполнения запроса по фильтрации и расчету отчета — менее 100 мс;
* успешное выполнение всех процедур и триггеров при массовой вставке;
* отсутствие ошибок в логике при проверке бизнес-правил;
* устойчивость при одновременной работе нескольких пользователей.

### ****Что можно улучшить в дальнейшем****

Система была спроектирована как учебно-прикладной проект, но при этом её архитектура позволяет масштабировать и дополнять. В будущем возможны следующие доработки:

| **Область** | **Возможные улучшения** |
| --- | --- |
| **Клиентская часть** | Разработка полноценного веб-интерфейса или приложения для сотрудников |
| **Отчеты** | Визуализация графиков, динамика заключения договоров, экспорт в PDF/Excel |
| **Интеграция** | Связь с CRM и бухгалтерией, API для мобильных клиентов |
| **Безопасность** | Подключение двухфакторной авторизации, регистрация устройств |
| **Аналитика** | Внедрение BI-модуля, анализ трендов, прогнозы на основе истории |
| **Интерфейс администратора** | Упрощение управления пользователями и ролями |

### ****Рекомендации по переносу логики на клиентскую сторону****

На данный момент все основные вычисления реализованы на стороне базы данных (SQL-функции и процедуры). Это повышает надёжность, но может вызывать дополнительные задержки при масштабировании. В случае увеличения количества пользователей рекомендуется:

* перенести не критичную бизнес-логику (например, фильтрацию и сортировку) на клиентскую сторону;
* реализовать кеширование часто используемых справочников;
* использовать REST API или GraphQL для гибкого доступа к данным.

### ****Вывод****

Курсовая работа позволила закрепить знания в области:

* анализа и моделирования предметной области;
* проектирования и нормализации реляционных баз данных;
* реализации хранимых процедур, триггеров, функций;
* настройки безопасности и резервного копирования в PostgreSQL.

Разработанная система может быть использована как основа для реального внедрения в компаниях страхового профиля. Она соответствует требованиям учебной программы, демонстрирует практическое применение теоретических знаний и содержит элементы исследовательской работы.

## **Список использованной литературы**

1. Федеральный закон от 27.07.2006 №149-ФЗ «Об информации, информационных технологиях и о защите информации». URL: <https://docs.cntd.ru/document/9005388>
2. Стружкин, Н. П. Базы данных: проектирование : учебник для СПО / Н. П. Стружкин, В. В. Годин. — Москва : Юрайт, 2023.
3. Стасышин, В. М. Базы данных: технологии доступа : учебное пособие для СПО / В. М. Стасышин, Т. Л. Стасышина. — Москва : Юрайт, 2023.
4. ГОСТ Р 34.10-2012 — Криптографическая защита информации.
5. Документация PostgreSQL: <https://www.postgresql.org/docs/>
6. Практические руководства по SQL и проектированию БД.
7.  Федеральный закон от 27.07.2006 №149-ФЗ «Об информации, информационных технологиях и о защите информации». URL: <https://docs.cntd.ru/document/9005388>
8.  Стружкин, Н. П., Годин, В. В. Базы данных: проектирование : учебник для СПО. — Москва : Юрайт, 2023. — 477 с.
9.  Стасышин, В. М., Стасышина, Т. Л. Базы данных: технологии доступа : учебное пособие для СПО. — Москва : Юрайт, 2023. — 164 с.
10.  ГОСТ Р 34.10-2012. Криптографическая защита информации.
11.  Китов, В. А., Музычкин, П. А., Неделькин, А. А. От кибернетики и АСУ до цифровой экономики. — Москва : РЭУ им. Г. В. Плеханова, 2020. — 64 с.
12.  Документация PostgreSQL. URL: <https://www.postgresql.org/docs/>
13.  Барышников, А. А. Проектирование баз данных. — Москва : Инфра-М, 2021. — 320 с.
14.  Кирьянов, В. М. Современные системы управления базами данных. — СПб : Питер, 2022. — 415 с.
15.  Парамонов, А. В., Коннов, И. А. Глоссарий официальных дефиниций в сфере ИТ и защиты информации. — Н. Новгород: Изд-во «Дятловы горы», 2021. — 232 с.
16.  Разработка и проектирование баз данных — этапы и методики. URL: <https://www.artwell.ru/services/razrabotka-baz-dannykh/>
17.  Метанит. Руководство по PostgreSQL. URL: <https://metanit.com/sql/postgresql/>
18.  Проблемы передачи информации / гл. ред. Бассалыго Л. А. — Москва: Академкнига, 2021. — 450 с.

## **Приложения**

* Код SQL создания таблиц и индексов
* Примеры SQL-запросов (выборки, группировки, обновления)
* Диаграммы и схемы