

Министерство образования, науки и молодежной политики Республики Коми  
ГПОУ “Сыктывкарский политехнический техникум”

## Курсовая работа

Тема: База данных для риэлторского агентства

**выполнил**

студент 4 курса

414 группы

Гилёв Иван Алексеевич

**Проверил**

Пунгин И.В.

дата проверки:

Сыктывкар, 2025

ВВЕДЕНИЕ .....	3
1.1 Актуальность темы.....	3
1.2 Цель и задачи .....	4
Задачи курсовой работы:.....	4
1.3 Практическая значимость .....	4
Анализ предметной области .....	5
2.1 Описание предметной области.....	5
2.2 Входящие и выходные данные.....	6
2.3 Ограничения.....	8
Проектирование базы данных .....	9
3.1 Концептуальная модель .....	9
3.2 Логическая структура.....	11
3.3 Физическая реализация в PostgreSQL .....	15
Реализация системы .....	18
4.1 Создание таблиц и индексов.....	18
4.2 Разработка интерфейса на Python .....	18
4.3 Тестирование системы .....	19
Безопасность и оптимизация .....	21
5.1 Защита данных.....	21
5.2 Оптимизация производительности .....	21
Заключение.....	22
6.1 Результаты работы .....	22
6.2 Перспективы развития .....	22
Список литературы.....	23
Приложения.....	24
Приложение 1.....	24

## **ВВЕДЕНИЕ**

### **1.1 Актуальность темы**

В современной экономике риэлторский бизнес является одной из наиболее динамично развивающихся отраслей услуг. Увеличение объемов сделок с недвижимостью, рост числа клиентов и объектов требуют эффективных инструментов управления информационными потоками. Традиционные методы ведения учета на бумажных носителях или в табличных редакторах становятся неэффективными, приводя к ошибкам, потерям данных и снижению производительности труда сотрудников.

Актуальность разработки специализированной информационной системы для риэлторского агентства обусловлена следующими факторами:

**Увеличение объемов данных:** Современное агентство работает с сотнями объектов недвижимости, тысячами клиентов и десятками сделок ежемесячно.

**Необходимость оперативного доступа:** Быстрый поиск подходящих объектов для клиентов и клиентов для объектов требует эффективной системы фильтрации.

**Требования к отчетности:** Владельцы бизнеса нуждаются в аналитике по продажам, эффективности агентов, финансовым показателям.

**Конкурентное преимущество:** Автоматизация процессов позволяет снизить операционные издержки и повысить качество обслуживания.

## **1.2 Цель и задачи**

Разработка полнофункциональной информационной системы для управления деятельностью риэлторского агентства, включающей базу данных на PostgreSQL и GUI-приложение на Python с использованием Tkinter.

Задачи курсовой работы:

- Провести анализ предметной области риэлторского агентства
- Разработать концептуальную (ER-диаграмму), логическую и физическую модели базы данных
- Реализовать базу данных в СУБД PostgreSQL
- Разработать GUI-приложение для работы с базой данных
- Реализовать систему безопасности и резервного копирования
- Создать документацию для пользователей и программистов

## **1.3 Практическая значимость**

Разработанная система имеет практическую ценность для:

- Личного использования
- Коммерческих систем — как основа для более сложных решений учета .

## **Анализ предметной области**

### **2.1 Описание предметной области**

Предметная область: Деятельность риэлторского агентства, специализирующегося на операциях с недвижимостью (продажа, аренда, сопровождение сделок).

Основные функции системы:

1. Управление персоналом:
  - Учет сотрудников (риэлторов)
  - Отслеживание комиссионных
  - Управление активностью агентов
2. Работа с клиентами:
  - Регистрация и классификация клиентов (покупатели/продавцы)
  - Ведение истории взаимодействий
  - Управление контактной информацией
3. Управление объектами недвижимости:
  - Каталогизация объектов (квартиры, дома, коммерческая недвижимость, земля)
    - Отслеживание статусов объектов (активен, продан, сдан в аренду)
    - Управление ценами и характеристиками
4. Организация сделок:
  - Регистрация сделок купли-продажи и аренды
  - Расчет комиссий
  - Ведение истории сделок
5. Планирование просмотров:
  - Организация встреч клиентов с объектами
  - Отслеживание результатов просмотров
  - Управление расписанием агентов

6. Предоставление услуг:

- Каталог дополнительных услуг (оценка, юридическое сопровождение)
- Учет заявок на услуги
- Расчет доходов от услуг

**2.2 Входные и выходные данные**

1. Данные сотрудников:

- ФИО, контакты, дата приема, процент комиссии, статус активности

2. Данные клиентов:

- ФИО, контакты, тип клиента, дата регистрации

3. Характеристики объектов недвижимости:

- Адрес, тип объекта, площадь, количество комнат, цена, статус
- Связи с владельцем и ответственным агентом

4. Информация о сделках:

- Связанный объект, покупатель, продавец, агент
- Дата сделки, цена, комиссия, тип сделки

5. Данные о просмотрах:

- Объект, клиент, агент, дата и время, статус, отзыв

6. Информация об услугах:

- Название, описание, стандартная цена, срок выполнения

7. Заявки на услуги:

- Клиент, услуга, агент, статус, фактическая цена, дата заявки

**Выходные данные**

1. Отчеты по сотрудникам:

- Статистика по количеству сделок
- Суммарная комиссия за период
- Список активных агентов

2. Статистика по объектам:
  - Количество объектов по типам
  - Средние цены по категориям
  - Соотношение проданных/активных объектов
3. Финансовая отчетность:
  - Общий объем сделок за период
  - Доходы от комиссий
  - Доходы от дополнительных услуг
4. Аналитические данные:
  - Эффективность агентов
  - Популярность типов недвижимости
  - Динамика сделок по месяцам
5. Операционные данные:
  - Расписание просмотров
  - Список текущих заявок
  - Активные объекты для показа

## **2.3 Ограничения**

Ограничения по типам данных:

- Тип клиента: только "buyer", "seller", "both"
- Тип недвижимости: только "apartment", "house", "commercial", "land"
- Статус объекта: только "active", "sold", "rented", "archived"
- Статус просмотра: только "scheduled", "completed", "cancelled"

Бизнес-правила:

- Комиссия агента не может превышать 100%
- Цена объекта должна быть положительной
- Дата просмотра не может быть в прошлом (для будущих просмотров)
- Email должен содержать символ "@"
- Телефон должен содержать минимум 5 цифр

Временные ограничения:

- Система работает в рамках одного часового пояса
- История хранится бессрочно, но может быть архивирована



## **Проектирование базы данных**

### **3.1 Концептуальная модель**

Концептуальная модель представляет собой диаграмму, отражающую основные сущности системы и связи между ними.

Основные сущности:

1. Сотрудник (Employee) - работник риэлторского агентства
2. Клиент (Client) - физическое или юридическое лицо, взаимодействующее с агентством
3. Объект недвижимости (Property) - недвижимость, предлагаемая к продаже/аренде
4. Сделка (Deal) - завершенная операция с недвижимостью
5. Просмотр (Viewing) - запланированный или состоявшийся показ объекта
6. Услуга (Service) - дополнительная услуга, предоставляемая агентством
7. Заявка на услугу (ServiceRequest) - запрос клиента на оказание услуги

Связи между сущностями:

1. Сотрудник - Объект недвижимости
  - Один сотрудник может отвечать за несколько объектов
  - Один объект закреплен за одним сотрудником
2. Клиент - Объект недвижимости как владелец
  - Один клиент может владеть несколькими объектами
  - Один объект принадлежит одному клиенту
3. Объект недвижимости - Сделка
  - Один объект может участвовать в нескольких сделках (последовательно)
  - Одна сделка относится к одному объекту

#### 4.Сотрудник - Сделка

- Один сотрудник может провести несколько сделок
- Одна сделка проводится одним сотрудником

#### 5.Клиент - Сделка через роли покупателя/продавца

- Один клиент, может быть, и покупателем и продавцом в разных сделках

- Одна сделка имеет одного покупателя и одного продавца

#### 6.Объект недвижимости - Просмотр

- Один объект может просматриваться несколько раз
- Один просмотр относится к одному объекту

#### 7.Клиент - Просмотр

- Один клиент может участвовать в нескольких просмотрах
- Один просмотр проводится для одного клиента

#### 8.Сотрудник - Просмотр

- Один сотрудник может сопровождать несколько просмотров
- Один просмотр сопровождается одним сотрудником

#### 9.Клиент - Заявка на услугу

- Один клиент может подать несколько заявок
- Одна заявка подается одним клиентом

#### 10.Услуга - Заявка на услугу

- Одна услуга может быть запрошена несколько раз
- Одна заявка относится к одной услуге



### 3.2 Логическая структура

Логическая структура базы данных состоит из 6 таблиц

Логическая структура базы данных состоит из 6 таблиц.

Для обеспечения целостности данных и исключения аномалий (вставки, обновления, удаления) была проведена нормализация базы данных.

#### Первая нормальная форма (1НФ):

Требуется, чтобы все атрибуты были атомарными (неделимыми). В таблице **properties** адрес разбит на составляющие (**city**, **district**, **address**), что позволяет выполнять поиск отдельно по городу или району.

## **Вторая нормальная форма (2НФ):**

Требует, чтобы таблица находилась в 1НФ и каждый не ключевой атрибут зависел от полного первичного ключа. В нашей схеме все таблицы имеют простой первичный ключ (id), поэтому условие 2НФ выполняется автоматически.

## **Третья нормальная форма (3НФ):**

Требует отсутствия транзитивных зависимостей (когда не ключевое поле зависит от другого не ключевого поля).

Пример: изначально предполагалось хранить стоимость услуги внутри таблицы заявки. Однако цена услуги зависит от типа услуги. Поэтому был создан отдельный справочник `services`, где хранится `standard_price`. В таблицу заявок `service_requests` цена копируется (`actual_price`) только для фиксации стоимости на момент продажи, что не нарушает 3НФ, а является необходимой денормализацией для финансовой истории.

Таблица 1: `employees`

`id`

`first_name` - имя

`last_name` - фамилия

`phone` – номер телефона,

`email` - почта

`hire_date` - дата

`commission_rate` – комиссия за работу,

`is_active` – задействован ли в работе

Таблица 2: `clients`

`id`

`first_name` – имя

`last_name` - фамилия

`phone` - телефон

email - почта

client\_type – тип клиента

registration\_date – дата регистрации

### Таблица 3: properties

id

address - адресс

city - город

district - район

property\_type – желаемая недвижимость

rooms – количество комнат

total\_area - область

living\_area - жилая площадь

floor – этажи

total\_floors - общее количество этажей

price - цена

status - статус

owner\_id, - владелец

agent\_id – агент

created\_date – дата создания

description - описание

### Таблица 4: deals

id

property\_id - собственность

buyer\_id - покупатель

seller\_id, - продавец

agent\_id - риэлтор

deal\_date - дата сделки

deal\_price – цена сделки  
commission\_amount – комиссия  
deal\_type - тип сделки  
payment\_method - способ оплаты  
notes - записи

Таблица 5: viewings

id  
property\_id - собственность  
client\_id - клиент  
agent\_id - риэлтор  
viewing\_date – дата осмотра  
status – статус просмотра  
client\_feedback – отзыв клиента  
agent\_notes – отзыв риэлтора

Таблица 6: services

id  
service\_name – название услуги  
description - описание  
standard\_price - цена  
duration\_days – продолжительность дней

Таблица 7: service\_requests

id  
client\_id - клиент  
service\_id - услуга  
agent\_id - риэлтор  
request\_date - дата запроса  
completion\_date - дата выполнения  
status - статус  
actual\_price – актуальная цена  
notes - записи

### 3.3 Физическая реализация в PostgreSQL

Физическая реализация базы данных включает в себя создание таблиц, индексов и ограничений.

Таблица employees

id	Serial primary key	Первичный ключ
first_name	VARCHAR(50) NOT NULL	имя
last_name	VARCHAR(50) NOT NULL,	фамилия
phone	VARCHAR(20) UNIQUE NOT NULL	телефон
email	VARCHAR(100) UNIQUE NOT NULL	почта
hire_date	DATE NOT NULL DEFAULT CURRENT_DATE,	дата
commission_rate	DECIMAL(5,2) DEFAULT 2.5,	комиссия
is_active	BOOLEAN DEFAULT TRUE	задействован ли в работе

Таблица clients

id	Serial primary key	Первичный ключ
first_name	Varchar(50) not null	Имя
last_name	VARCHAR(50) NOT NULL	Фамилия
phone	VARCHAR(20) NOT NULL	телефон
email	VARCHAR(100)	почта
client_type	VARCHAR(10) CHECK (client_type IN ('buyer', 'seller', 'both')),	Тип клиента
registration_date	DATE DEFAULT CURRENT_DATE	Дата регистрации

Таблица properties

id	Serial primary key	Первичный ключ
address	VARCHAR(200) NOT NULL	Адресс
city	VARCHAR(50) NOT NULL,	Город
district	VARCHAR(50),	Район
property_type	VARCHAR(20) CHECK (property_type IN ('apartment', 'house', 'commercial', 'land')),	желаемая недвижимость
rooms	INTEGER	Количество комнат
total_area	DECIMAL(10,2) NOT NULL,	область
living_area	DECIMAL(10,2)	жилая площадь
floor	INTEGER	этаж
total_floors	INTEGER	Количество этаже в доме
price	DECIMAL(12,2) NOT NULL	Цена
status	VARCHAR(20) DEFAULT 'active' CHECK (status IN ('active', 'sold', 'rented', 'archived')),	Статус
owner_id	INTEGER REFERENCES clients(id)	Внешний ключ
agent_id	INTEGER REFERENCES employees(id)	Внешний ключ
created_date	DATE DEFAULT CURRENT_DATE	Дата создания

При создании таблиц были выбраны следующие типы данных, оптимальные с точки зрения хранения и обработки:

- Integer / Serial: для первичных и внешних ключей. Занимает 4 байта, обеспечивает быструю индексацию.



- `Varchar(N)`: для строковых данных (ФИО, email). В отличие от `Char(N)`, занимает столько места, сколько реально весит строка + служебная информация.
- `Decimal(12, 2)`: для цен и комиссий. Типы с плавающей точкой (`Real`, `Double`) не подходят для денег из-за ошибок округления. `Decimal` хранит числа точно.
- `Boolean`: для флагов (`is_active`). Занимает 1 байт.
- `Date / Timestamp`: для хранения временных меток событий.

## Реализация системы

### 4.1 Создание таблиц и индексов

База данных была разработана в СУБД PostgreSQL 14.

Наполнение тестовыми данными:

```
INSERT INTO employees (first_name, last_name, phone, email,
commission_rate) VALUES
```

```
('Иван', 'Петров', '+79161234567', 'ivan@agency.ru', 3.0),
```

```
('Мария', 'Сидорова', '+79167654321', 'maria@agency.ru', 2.8);
```

```
INSERT INTO clients (first_name, last_name, phone, email, client_type)
VALUES
```

```
('Алексей', 'Смирнов', '+79161112233', 'alex@mail.ru', 'seller'),
```

```
('Елена', 'Кузнецова', '+79164445566', 'elena@gmail.com', 'buyer');
```

```
INSERT INTO services (service_name, description, standard_price)
VALUES ('Оценка недвижимости', 'Профессиональная оценка рыночной
стоимости', 5000.00),
```

```
('Юридическое сопровождение', 'Проверка документов и оформление
сделки', 15000.00);
```

### 4.2 Разработка интерфейса на Python

Пользовательский интерфейс разработан на python с использованием библиотеки tkinter для создания графического интерфейса и psycopg2 для работы с СУБД PostgreSQL.

Архитектура приложения:

#### 1. ВНЕШНИЙ УРОВЕНЬ (External Level)

- Представления для агентов
- Представления для клиентов (через API)
- Представления для аналитики
- Представления для администраторов

## 2. КОНЦЕПТУАЛЬНЫЙ УРОВЕНЬ (Conceptual Level)

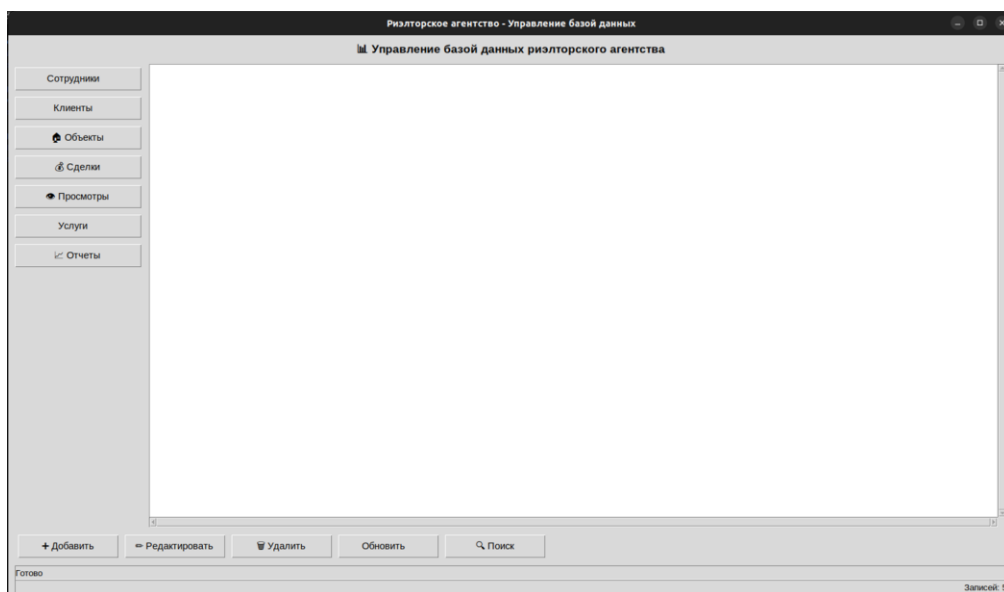
- Логическая схема базы данных
- Сущности и их отношения
- Ограничения целостности
- Бизнес-правила

## 3. ВНУТРЕННИЙ УРОВЕНЬ (Internal Level)

- Физическое хранение данных
- Индексы и структуры доступа
- Стратегия кэширования
- Стратегия резервного копирования

Основные компоненты интерфейса:

- Главное окно:
- Адаптивный интерфейс (подстраивается под разрешение экрана)
- Вкладки для каждой сущности
- Панель управления с кнопками CRUD
- Статусная строка с информацией

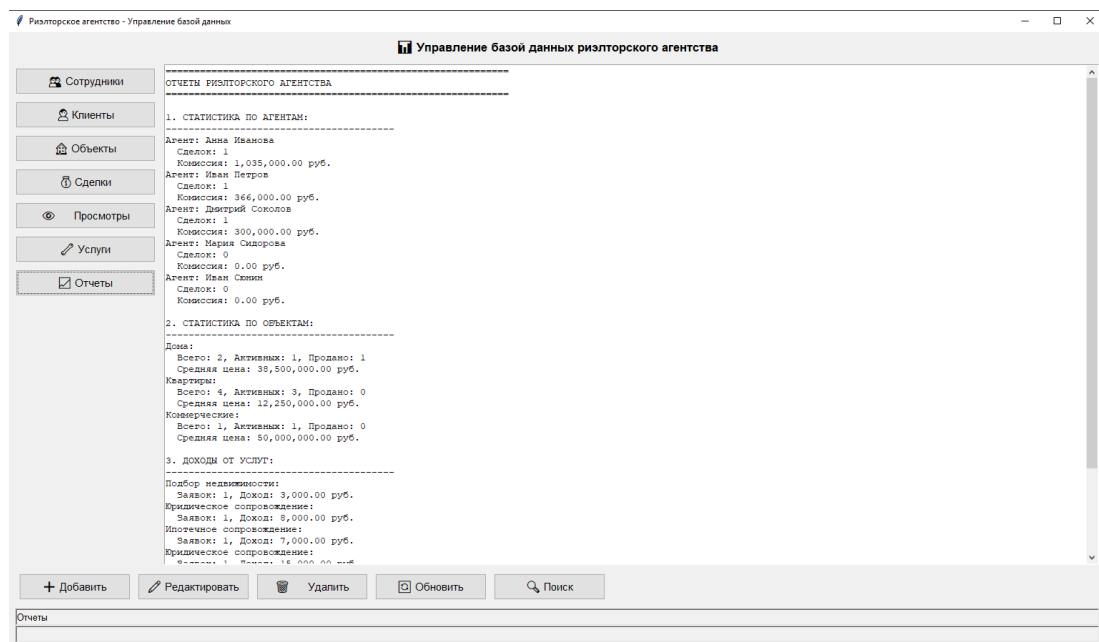


Диалоговые окна:

- Единообразный дизайн для всех форм
- Валидация ввода данных
- Подсказки и сообщения об ошибках
- Автозаполнение для связанных данных

Отчеты:

- Статистика по агентам
- Аналитика по объектам
- Финансовые отчеты
- Экспорт в текстовый формат



## 4.3 Тестирование системы

Виды тестирования:

### 1. Функциональное тестирование

Добавление агента с корректными данными

Добавление объекта с некорректными данными

Попытка добавить клиента с некорректным рейтингом

Поиск объекта по названию

## 2. Тестирование пользовательского интерфейса

Проверка на работоспособность всех кнопок

Проверка на корректное отображение ошибок

Обнаруженные и исправленные ошибки:

1. Проблема: при добавлении осмотра у него не проверялась дата.

Решение: Добавлена проверка на то, что дата просмотра не может быть раньше, чем нынешнее время

2. Проблема: при нажатии таблицы отчёты не переключалось на другие таблицы

Решение: `def edit_employee_dialog(self, dialog, record_id, values):`  
`pass.`

## Безопасность и оптимизация

### 5.1 Защита данных

Ограничение доступа на уровне базы данных:

- Создание представлений для ограничения доступа к данным

#### Дашборд агента

```
CREATE VIEW v_agent_dashboard AS
SELECT
    e.id as agent_id,
    e.first_name || ' ' || e.last_name as agent_name,
    e.commission_rate,
    COUNT(DISTINCT CASE WHEN p.status = 'active' THEN p.id END) as active_properties,
    COUNT(DISTINCT CASE WHEN v.status = 'scheduled' AND DATE(v.viewing_date) = CURRENT_DATE THEN v.id END) as todays_viewings,
    COUNT(DISTINCT CASE WHEN d.deal_date >= DATE_TRUNC('month', CURRENT_DATE) THEN d.id END) as monthly_deals,
    COALESCE(SUM(CASE WHEN d.deal_date >= DATE_TRUNC('month', CURRENT_DATE) THEN d.commission_amount END), 0) as monthly_commission,
    COUNT(DISTINCT CASE WHEN sr.status IN ('new', 'in_progress') THEN sr.id END)
as pending_services
FROM employees e
LEFT JOIN properties p ON e.id = p.agent_id
LEFT JOIN viewings v ON e.id = v.agent_id
LEFT JOIN deals d ON e.id = d.agent_id
LEFT JOIN service_requests sr ON e.id = sr.agent_id
WHERE e.is_active = TRUE
GROUP BY e.id, e.first_name, e.last_name, e.commission_rate;
```

#### Статистика по объектам недвижимости

```
CREATE VIEW v_property_statistics AS
SELECT
    p.city,
    p.property_type,
    COUNT(p.id) as total_properties,
    COUNT(CASE WHEN p.status = 'active' THEN p.id END) as active_properties,
    COUNT(CASE WHEN p.status = 'sold' THEN p.id END) as sold_properties,
    COUNT(CASE WHEN p.status = 'rented' THEN p.id END) as rented_properties,
    ROUND(AVG(p.price), 2) as avg_price,
    MIN(p.price) as min_price,
    MAX(p.price) as max_price,
    ROUND(AVG(p.total_area), 2) as avg_area,
```

```
ROUND(AVG(p.price / NULLIF(p.total_area, 0)), 2) as avg_price_per_sqm,
AVG(EXTRACT(DAY FROM CURRENT_DATE - p.created_date)) as avg_days_on_market
FROM properties p
GROUP BY p.city, p.property_type;
```

## Мониторинг производительности запросов

```
CREATE VIEW v_query_performance_monitoring AS
SELECT
query,
calls,
total_exec_time,
mean_exec_time,
rows,
DATE_TRUNC('hour', query_start) as hour_period
FROM pg_stat_statements -- Требуется расширение pg_stat_statements
WHERE query_start >= CURRENT_DATE - INTERVAL '7 days'
ORDER BY total_exec_time DESC
LIMIT 100;
```

- Использование ролевой модели

*-- Администратор базы данных (операционный доступ)*

```
CREATE ROLE db_admin WITH
NOSUPERUSER
CREATEDB
CREATEROLE
LOGIN
PASSWORD 'admin_secure_pass'
VALID UNTIL 'infinity';
```

*-- Роль риэлтора/агента*

```
CREATE ROLE agent_role WITH
NOLOGIN
NOINHERIT;
```

**Резервное копирование** — процесс, целью которого является обеспечить возможность восстановления данных при сбое (аппаратном, человеческой ошибке и пр.) с минимальными потерями.

Сначала был воспроизведён полный бэкап всей базы данных:  
pg\_dump -U gillob gillob > gillobdb\_backup.sql

Затем были созданы бэкапы отдельных таблиц: `pg_dump -t reservation elesov > /tmp/reservationhostel.dump`; `pg_dump -t schedule_of_cleaning_rooms elesov > /tmp/schedule_of_cleaning_rooms hostel.dump`;

Также была создана копия python-файла, отвечавшего за графический пользовательский интерфейс.

## **5.2 Оптимизация производительности**

Оптимизация индексов:

- Создание составных индексов для часто используемых комбинаций полей

```
CREATE INDEX idx_properties_city ON properties(city);
```

```
CREATE INDEX idx_properties_price ON properties(price);
```

```
CREATE INDEX idx_properties_status ON properties(status);
```

```
CREATE INDEX idx_deals_date ON deals(deal_date);
```

```
CREATE INDEX idx_deals_agent ON deals(agent_id);
```

```
CREATE INDEX idx_viewings_date ON viewings(viewing_date);
```



## **Заключение**

### **6.1 Результаты работы**

В ходе курсовой работы была спроектирована и реализована реляционная база данных для риэлторского агентства. Были выполнены следующие этапы:

- Проведен анализ предметной области, определены ключевые сущности и бизнес-процессы.
- Построена инфологическая (ER) и логическая модель данных, проведена нормализация отношений.
- Разработана физическая структура БД в СУБД PostgreSQL с учетом требований к производительности (индексы) и целостности данных (ограничения, внешние ключи).
- Созданы основные таблицы, заполнены тестовыми данными, написаны ключевые SQL-запросы для получения бизнес-отчетности.
- Предложена архитектура системы безопасности (роли, привилегии) и стратегия резервного копирования.
- Описана концепция REST API для будущей интеграции с веб-интерфейсом.

Результат: Созданная база данных является полноценным ядром информационной системы, способной эффективно автоматизировать учетные и аналитические задачи риэлторского агентства, повысить управляемость бизнес-процессами и качество обслуживания клиентов

### **6.2 Перспективы развития**

Данная система представляет собой хорошую основу для дальнейшего развития.

#### **1. Расширение функциональности:**

- Добавление рекомендаций на основе объектов, которые смотрит пользователь

- Добавление рецензий и обзоров

#### **2. Улучшение интерфейса:**

- Разработка веб версии приложения
- Поддержка нескольких языков

## **Список литературы**

1. PostgreSQL 15: Official Documentation. - URL: <https://www.postgresql.org/docs/15/>
2. Tkinter Documentation. - URL: <https://docs.python.org/3/library/tkinter.html>
3. ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем.
4. Робинсон С., Серов С. PostgreSQL: Основы языка SQL. - СПб.: БХВ-Петербург, 2020.

# Приложения

## Приложение 1

### 1.Фрагмент кода приложения

```
1 import tkinter as tk
2 from tkinter import ttk, messagebox
3
4 import fields
5 import psycopg2
6 from datetime import datetime
7
8 import row
9
10
11 class RealEstateApp:
12     def __init__(self, root):
13         self.root = root
14         self.root.title("Риэлторское агентство - Управление базой данных")
15         self.root.geometry("1200x700")
16
17         # Получаем размеры экрана
18         screen_width = self.root.winfo_screenwidth()
19         screen_height = self.root.winfo_screenheight()
20
21         # Устанавливаем размеры окна в зависимости от разрешения экрана
22         if screen_width >= 1920 and screen_height >= 1080: # Full HD и выше
23             window_width = int(screen_width * 0.8)
24             window_height = int(screen_height * 0.8)
25         elif screen_width >= 1366 and screen_height >= 768: # HD
26             window_width = int(screen_width * 0.85)
27             window_height = int(screen_height * 0.85)
28         else: # Низкое разрешение
29             window_width = int(screen_width * 0.9)
30             window_height = int(screen_height * 0.9)
31
32         # Центрируем окно
33         x = (screen_width - window_width) // 2
34         y = (screen_height - window_height) // 2
35         self.root.geometry(f"{window_width}x{window_height}+{x}+{y}")
36
37         # Минимальный размер окна
38         self.root.minsize(800, 600)
39
40         # Параметры подключения к БД
```

**Добавить запись в viewings**

Объект:\*

Клиент:\*

Агент:

Дата и время просмотра \*

Дата:  Выбранная дата: 2025-12-15

Время: 15 : 21

Время в прошлом!

Статус:\*

Отзыв клиента:

## 2.Добавление записи в таблицу viewings

**Редактировать запись в services**

**Редактирование услуги**

Название услуги:\*

Описание:

Цена:\*

Срок (дней):

## 3.Редактирование услуг в таблице services