

РГРТУ им. В.Ф. Уткина  
КУРСОВОЙ ПРОЕКТ

135-09.03.02

Жильцов. В.А.

2025

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  
РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ  
им. В.Ф. УТКИНА  
Кафедра автоматизированных систем управления

З А Д А Н И Е  
на курсовой проект по дисциплине  
«Программное обеспечение информационных систем»

Студент – Жильцов В. А. группа – 135

1. Тема проекта: программное обеспечение инжиниринга базы данных информационной системы
2. Срок представления к защите – 12.04.2025 г.
3. Исходные данные для проектирования.
  - 3.1. Предметная область – центр исследования строительных конструкций
  - 3.2. СУБД – PostgreSQL
  - 3.3. Инструментальные средства - BPWin, ERWin.
4. Содержание пояснительной записи курсового проекта.
  - 4.1. Задание.
  - 4.2. Теоретическая часть.
    - 4.2.1. Системный анализ и описание предметной области.
    - 4.2.2. Описание и анализ бизнес-процессов предметной области.
    - 4.2.3. Выявление, анализ и формулирование требований (бизнес-логика, правила, требования, ограничения).
    - 4.2.4. Разработка концептуальной модели.
    - 4.2.5. Логическое проектирование.
    - 4.2.6. Физическое проектирование.
  - 4.3. Практическая часть.
    - 4.3.1. Реализация бизнес-требований и ограничений.
    - 4.3.2. Реализация бизнес-логики и правил.
    - 4.3.3. Реализация политики безопасности.
  - 4.4. Экспериментальная часть.

Руководитель проекта \_\_\_\_\_ / Маркин А.В /  
(подпись) (Фамилия И.О.)

Задание принял к исполнению \_\_\_\_\_ 11.02.2025 г.  
(подпись) (дата)

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ  
УНИВЕРСИТЕТ ИМЕНИ В.Ф. УТКИНА

Кафедра Автоматизированных Систем Управления

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ ИНЖИНИРИНГА БАЗЫ ДАННЫХ  
ИНФОРМАЦИОННОЙ СИСТЕМЫ ЦЕНТРА ИССЛЕДОВАНИЙ  
СТРОИТЕЛЬНЫХ КОНСТРУКЦИЙ

Пояснительная записка

Рязань 2025

## **Аннотация**

В рамках курсовом проекте осуществляется разработка программного обеспечения инжиниринга базы данных информационной системы центра исследования строительных конструкций. В проекте выполняются следующие задачи: анализ предметной области, определение бизнес-процессов, бизнес-ограничений, бизнес-требований, создание логической и физической модели, разработка базы данных, реализация бизнес-процессов, обеспечение соблюдения бизнес-требований и ограничений, выявление и реализация политики безопасности организации. Под поставленные задачи, проектируются, программируются и тестируются хранимые функции и триггеры. В качестве системы управления базами данных используется PostgreSQL, а для администрирования базы данных применяется клиентское приложение DBeaver.

## Оглавление

<b>Введение.....</b>	<b>7</b>
<b>1 Анализ предметной области .....</b>	<b>9</b>
1.1 Описание предметной области.....	9
1.2 Определение основных бизнес-процессов предметной области.....	10
1.3 Определение бизнес-требований предметной области .....	11
1.4 Определение бизнес-ограничений и допущений предметной области	
12	
<b>2 Разработка моделей .....</b>	<b>13</b>
2.1 Разработка функциональной модели.....	13
2.2 Разработка диаграммы потоков данных .....	17
2.3 Разработка локальных концептуальных моделей .....	26
2.4.1 Определение атрибутов сущностей и их ключей .....	26
2.4.2 Выявление связей между сущностями.....	35
<b>3 Разработка глобальных моделей .....</b>	<b>38</b>
3.1 Разработка глобальной логической модели.....	38
3.2 Разработка глобальной физической модели данных .....	40
<b>4 Детализация задания.....</b>	<b>42</b>
4.1 Проектирование таблиц [12] .....	42
4.2 Проектирование хранимых функций .....	48
4.2.1 Проектирование функций .....	48
4.2.2 Проектирование хранимых процедур .....	48
4.3 Проектирование триггеров [13] .....	50
4.3.1 Проектирование триггеров бизнес-требований .....	50
4.3.2 Проектирование триггеров бизнес-ограничений.....	50
4.3.3 Проектирование триггера ограничений целостности .....	51
4.3.4 Проектирование триггера ограничения действия.....	51
4.3.5 Проектирование триггера DDL.....	51
4.3.6 Проектирование триггера БД.....	51
4.4 Разработка реализации политики безопасности .....	51

5 Скрипты базы данных .....	54
5.1 Скрипты создания базы данных и таблиц базы данных .....	54
5.2 Скрипты данных .....	56
5.3 Скрипты хранимых функций и процедур .....	59
5.3.1 Скрипты селективных функций .....	59
5.3.2 Скрипты выполняемых процедур .....	64
5.4 Скрипты триггеров .....	68
5.4.1 Скрипты триггеров бизнес-правил .....	68
5.4.2 Скрипты триггеров бизнес-ограничений .....	69
5.4.3 Скрипт триггера ограничения целостности .....	70
5.4.4 Скрипт триггеров ограничения действия .....	71
5.4.5 Скрипт триггера DDL .....	71
5.4.6 Скрипт триггера БД .....	72
5.5 Скрипты реализации политики безопасности .....	72
6 Экспериментальная часть .....	74
6.1 Проверка базы данных и наличия таблиц .....	74
6.2 Проверка таблиц .....	74
6.3 Проверка работы хранимых функций и процедур .....	78
6.3.1 Проверка работы селективных функций .....	78
6.3.2 Проверка работы выполняемых процедур .....	80
6.4 Проверка работы триггеров .....	86
6.4.1 Проверка триггеров бизнес-правил .....	86
6.4.2 Проверка триггеров бизнес-ограничений .....	87
6.4.3 Проверка триггера ограничения целостности .....	88
6.4.4 Проверка триггеров ограничения действия .....	88
6.4.5 Проверка триггера DDL .....	89
6.4.6 Проверка триггера БД .....	89
6.5 Проверка реализации политики безопасности .....	90
Заключение .....	95
Библиографический список .....	96

## **Введение**

На сегодняшний день компании по инженерным взысканиям и проектированию имеют большой спектр предоставляемых услуг. Учёт заявок на обследование очень важен для компаний такого типа, поскольку он позволяет ускорить обработку заявок поданных клиентами подобных организаций [1]. Автоматизированная информационная система процесса учета поданных заявок на обследование строительных конструкций значительно снижает время обработки заявок на обследование, помогает эффективно распределять бригады на объекты обследования и снижает время процесса вычисления суммы за оказанные услуги компанией [2] [3].

Актуальность выбранной темы «Программное обеспечение инжиниринга базы данных информационной системы центра исследования строительных конструкций» основывается на том, что учет заявок для компаний такого вида является их неотъемлемой частью. Также автоматизация основных процессов по подаче заявки облегчит взаимодействие самих клиентов с компанией и позволит отслеживать результаты обследования, представленные в виде отчета об обследовании [2].

Цель работы – разработка программного обеспечения инжиниринга базы данных информационной системы центра исследований строительных конструкций, которая позволит автоматизировать процессы подачи заявки, назначения исполнительной бригады, формирование договора, работы с клиенткой базой.

Для достижения цели требуется решить следующие задачи:

- описать предметную область;
- описать и проанализировать бизнес-процессы предметной области;
- выявить, проанализировать и сформулировать требования и ограничения;

- разработать функциональную модель;
  - разработать диаграммы потоков данных;
  - разработать концептуальную модель;
  - разработать глобальную логическую модель базы данных;
  - разработать глобальную физическую модель базы данных;
  - реализовать выявленные требования с помощью инструментария СУБД PostgreSQL;
- протестировать базу данных.

# **1      Анализ предметной области**

## **1.1    Описание предметной области**

Компании, занимающиеся обследованием строительных конструкций, играют важную роль в обеспечении безопасности и надежности зданий и сооружений [4]. Эти компании проводят всестороннюю оценку состояния строительных объектов, выявляют возможные дефекты и проблемы, а также разрабатывают рекомендации по их устранению. Благодаря своей деятельности, такие компании помогают предотвратить аварии и разрушения, обеспечивая тем самым безопасность людей и долговечность строений.

Основные аспекты центров исследования строительных конструкций и материалов:

Цели и задачи: компании, занимающиеся обследованием строительных конструкций, направлены на всестороннее изучение и оценку состояния зданий и сооружений. Задачи включают в себя выявление дефектов, оценку прочности и надежности конструкций, разработку рекомендаций по их устранению, а также обеспечение безопасности эксплуатации объектов.

Методы обследования: обследование строительных конструкций базируется на использовании современных технологий и инструментов. Применяются как визуальные методы осмотра, так и инструментальные методы диагностики, включая ультразвуковое, радиографическое и магнитное обследование. Комплексный подход позволяет детально изучить состояние конструкции, выявить скрытые дефекты и провести точную оценку.

Целью курсового проекта является инжиниринг базы данных информационной системы центра исследований строительных конструкций, которая обеспечит автоматизацию технических процессов, повысит уровень удобства для клиентов, упростит работу сотрудников и увеличит безопасность данных.

База данных будет использоваться:

1. Со стороны клиента, чтобы:

- клиент мог ознакомиться с предоставляемыми услугами;
- оформить заявку на обследование требуемой конструкции;
- добавлять требуемые ему услуги для обследования;
- просматривать отчет об обследовании;
- просматривать и утверждать договор.

2. Сотрудниками центра исследования для того, чтобы:

- обрабатывать заявки клиентов на обследование;
- контролировать прибыль с заказа, организации или

исполнительных бригад;

- вести отчетность по активным заявкам.

3. Администратором системы, который:

- контролирует техническую работу системы;
- управляет пользователями и сотрудниками;
- Отменяет транзакции.

В системе установим 3 уровня доступа:

- 1) для клиента (ограниченный) – просмотр информации о услугах, подача заявок, просмотр отчетов по своим заявкам;
- 2) для сотрудника (расширенный) – доступ к заявкам клиентов, ведение учета заявок, добавление отчетов в базу;
- 3) для администратора – управление данными, настройка системы, контроль пользователей и безопасности.

## 1.2 Определение основных бизнес-процессов предметной области

Основными бизнес-процессами центра исследования строительных конструкций являются:

- 1) прием заявок на обследование строительных конструкций;
- 2) ведение учетной базы клиентов;

- 3) формирование бригад занимающихся инженерными взысканиями и проектированием;
- 4) отслеживание время исполнения заявок клиента;
- 5) отслеживание прибыльности организаций, на которых происходили обследования;
- 6) отслеживание прибыльности исполнительных бригад;
- 7) формирование стоимости выполняемого обследования;
- 8) отслеживание информации об заявки на обследование.

Процесс обследования строительных конструкций начинается с поступления заявки от клиента, который является доверенным лицом какой-либо организации. Информация о заявке, клиенте, организации, а также адресе, на котором расположен объект обследования заносится в базу данных центра исследования строительных конструкций.

После получения заявки на обследование назначается бригада, которая будет выполнять требуемые работы на объекте. Когда обследование завершено сотрудник, состоящий в бригаде, оформляет отчет об обследовании строительных конструкций. После составления отчета клиент оплачивает заказ и получает отчет о проведенном обследовании.

### **1.3 Определение бизнес-требований предметной области**

В качестве бизнес-требований выступает необходимость автоматизировать:

- 1) формирование объекта обследования;
- 2) учет наличия нескольких видов услуг в одной заявки;
- 3) назначение бригады на заявку;
- 4) учет проделанных работ;
- 5) учет статуса заявки;
- 6) расчет стоимости каждой предоставляемой услуги;
- 7) расчет стоимости заявки на обследование;

## **1.4 Определение бизнес-ограничений и допущений предметной области**

Выявлены следующие бизнес-ограничения, которые накладываются на предметную область:

- 1) у заказчика может быть только один номер телефона;
- 2) у заказчика и сотрудника может быть только одна учетная запись;
- 3) при оформлении заявки не устанавливаются точные сроки готовности отчета;
- 4) все сотрудники, входящие в состав бригад, имеют должную квалификацию для обследования строительных конструкций
- 5) отчет не может быть добавлен если статус заявки не выполнена;
- 6) договор не может быть сформирован если статус заявки не выполнена и отчет об обследовании не был добавлен;
- 7) дата начала обследования не может быть раньше даты подачи заявки;
- 8) дата окончания обследования не может быть раньше даты начала обследования;
- 9) отчет не может быть удален если связанный договор был подтвержден клиентом;
- 10) отчет должен быть сформирован сотрудником, который состоит в исполнительной бригаде, которая выполняла заявку.

## 2 Разработка моделей

### 2.1 Разработка функциональной модели

Программное обеспечение AllFusion Process Modeler (BPwin) [5] представляет собой мощный инструмент для моделирования, анализа, описания и оптимизации бизнес-процессов. Он имеет удобный и понятный интерфейс, который позволяет разрабатывать сложные модели с минимальными усилиями. BPwin поддерживает три основные методологии моделирования: функциональное моделирование (IDEF0) — для верхнеуровневого анализа [6]; описание бизнес-процессов (IDEF3) — для работы с потоками [6] и диаграммами потоков данных (DFD) [7].

Методология IDEF0 основывается на таких ключевых компонентах, как:

- входы (Input) — данные или объекты, которые изменяются или используются в процессе;
- выходы (Output) — конечные результаты выполнения процесса;
- управления (Control) — правила, стандарты или нормативные акты, определяющие выполнение процесса;
- механизмы (Mechanism) — ресурсы, задействованные в процессе (персонал, оборудование, системы).

Создание модели начинается с описания системы в целом и её взаимодействия с внешней средой. Входы, выходы, управления и механизмы формируют графическое представление, которое наглядно отображает работу системы. Функциональная модель используется для анализа текущих бизнес-процессов компании, помогает выявить слабые стороны, определить области для улучшения и оптимизировать деятельность, повышая её структуру и эффективность.

Модель IDEF0 представляет собой набор диаграмм. На первом этапе диаграмма является самым общим и абстрактным представлением всей системы, которое можно затем детализировать в более конкретные схемы.

Каждая детализированная диаграмма представляет собой развернутую версию одного из блоков более общей диаграммы. Диаграмма, содержащая общее описание, называется родительской, а более детализированная — дочерней. Процесс моделирования начинается с создания контекстной диаграммы, включающей контекстный блок.

Контекстная диаграмма представлена на рисунке 1.



Рисунок 1 – Контекстная диаграмма

Диаграмма представляет из себя единственный блок, который показывает основную задачу моделируемой системы: принять заказ на обследование строительных конструкций. Блок представлен с граничными потоками, которые отображают взаимодействие окружающей среды с системой.

Слева блока находится информация, которая подлежит обработке: заказчик, передаваемый для принятия заказа на обследование строительных

конструкций. С правой блока находится результат, выдающий моделируемая система – это технический отчет по обследованию конструкций.

Сверху в блок поступают данные и нормативные документы, регулирующие действия центра обследования строительных конструкций: справочник услуг, а также государственные правила обследования конструкций. Снизу в блок поступают сотрудники центра исследований, выступающие механизмом, обеспечивающим выполнение операций.

После создания контекстной диаграммы, которая представляет собой описание контекста системы, проводится ее декомпозиция. Декомпозиция или функциональная декомпозиция представляется из себя разбиение концептуальной модели на несколько подсистем, которые описываются в том же синтаксисе что и система в целом. После чего эти подсистемы декомпозицииются до достижения нужного уровня подробности. В результате такого разбиения каждый бизнес – процесс, реализуемый в системе, изображается на отдельной диаграмме декомпозиции. Диаграмма декомпозиции предназначена для более детализированного отображения работы и функционирования системы.

Диаграмма декомпозиции контекстной диаграммы представлена на рисунке 2.

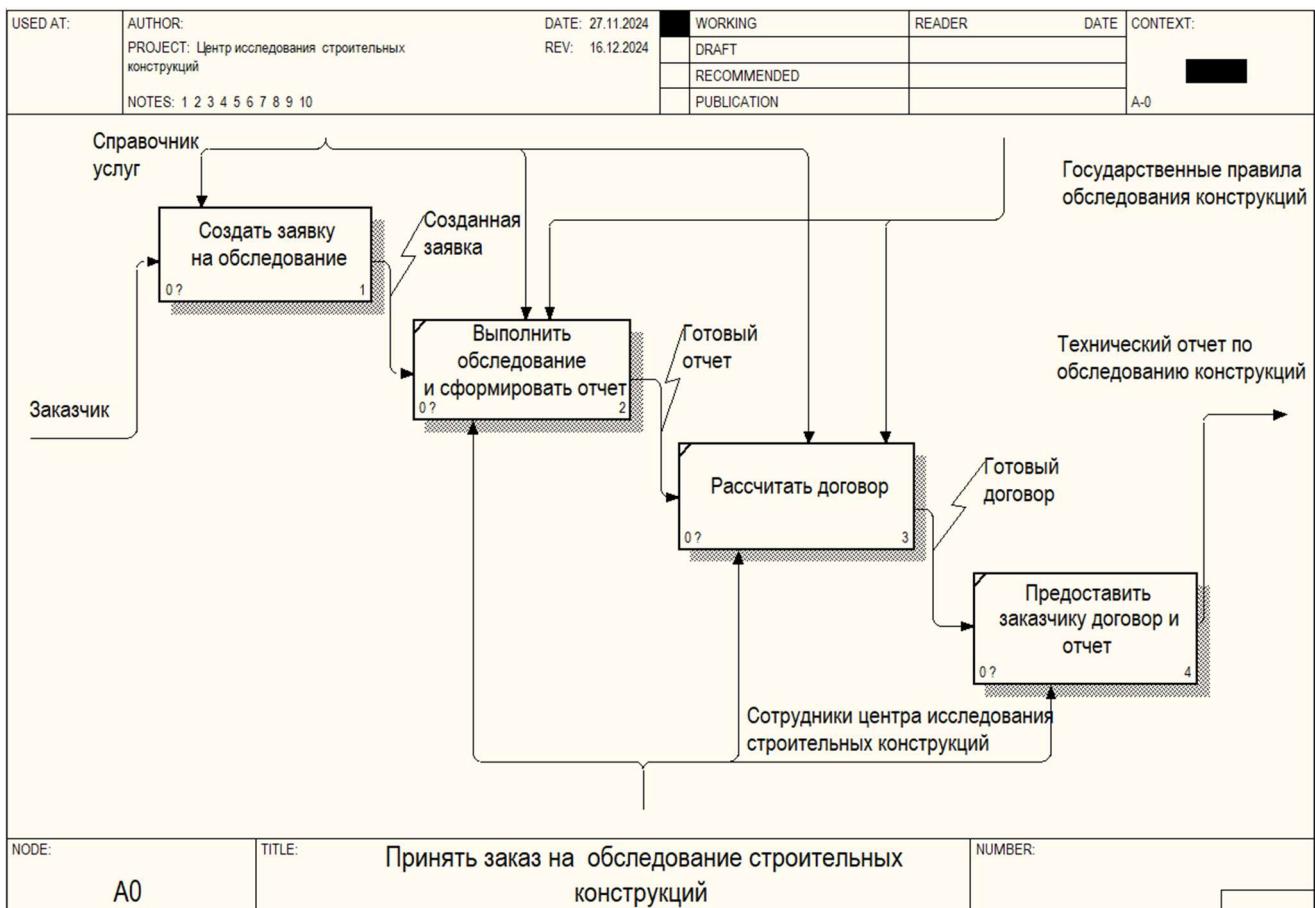


Рисунок 2 – Декомпозиция моделируемой системы

При анализе предметной области были выявлены основные этапы работы, которые необходимо выполнить для принятия заявки на обследование строительных конструкций, что бы в результате заказчик получил технический отчет по обследованию конструкций. Описание каждого этапа представлено ниже.

Для принятия заказа на обследование строительных конструкций необходимо чтобы заказчик создал заявку на обследование. Для данной процедуры заказчику необходимо выбрать требуемые ему услуги.

Процессом выполнения обследования и формировании отчета занимаются сотрудники центра исследования согласно с государственными правилами обследования конструкций. После проведения обследования сотрудники должны составить отчет по проведенному обследованию.

В процессе расчета договора необходимо рассчитать сумму, которую должен будет оплатить заказчик, а также сформировать договор.

После того как договор и отчет сформированы необходимо что бы заказчик подтвердил договор заказчиком необходимо передать отчет заказчику.

## 2.2 Разработка диаграммы потоков данных

DFD (Data Flow Diagram), или диаграмма потоков данных, представляет собой инструмент визуального моделирования, предназначенный для анализа, разработки и документирования информационных систем. Она наглядно демонстрирует, как данные перемещаются внутри системы, какие процессы их обрабатывают, где происходит их хранение и какие внешние сущности взаимодействуют с системой. Такой подход помогает разработчикам, аналитикам и другим участникам проекта глубже понять устройство и логику системы, а также обнаружить потенциальные слабые места или избыточные процессы [7].

Диаграммы потоков данных состоят из 4 основных элементов [8]:

- процессы — это действия, которые преобразуют входные данные в выходные. Процессы обозначаются с помощью прямоугольников с закруглёнными углами;
- потоки данных – это стрелки, которые показывают, как данные передаются между процессами, хранилищами данных и внешними сущностями;
- хранилища данных – это таблицы, в которых происходит хранение данных. Они обозначаются открытым прямоугольником;
- внешние сущности — это объекты, которые находятся за пределами моделируемой системы, но взаимодействуют с ней, предоставляя данные или получая результаты. Внешние сущности изображаются прямоугольниками.

На рисунке 3 показана декомпозиция второго уровня блока «Создать заявку на обследование».

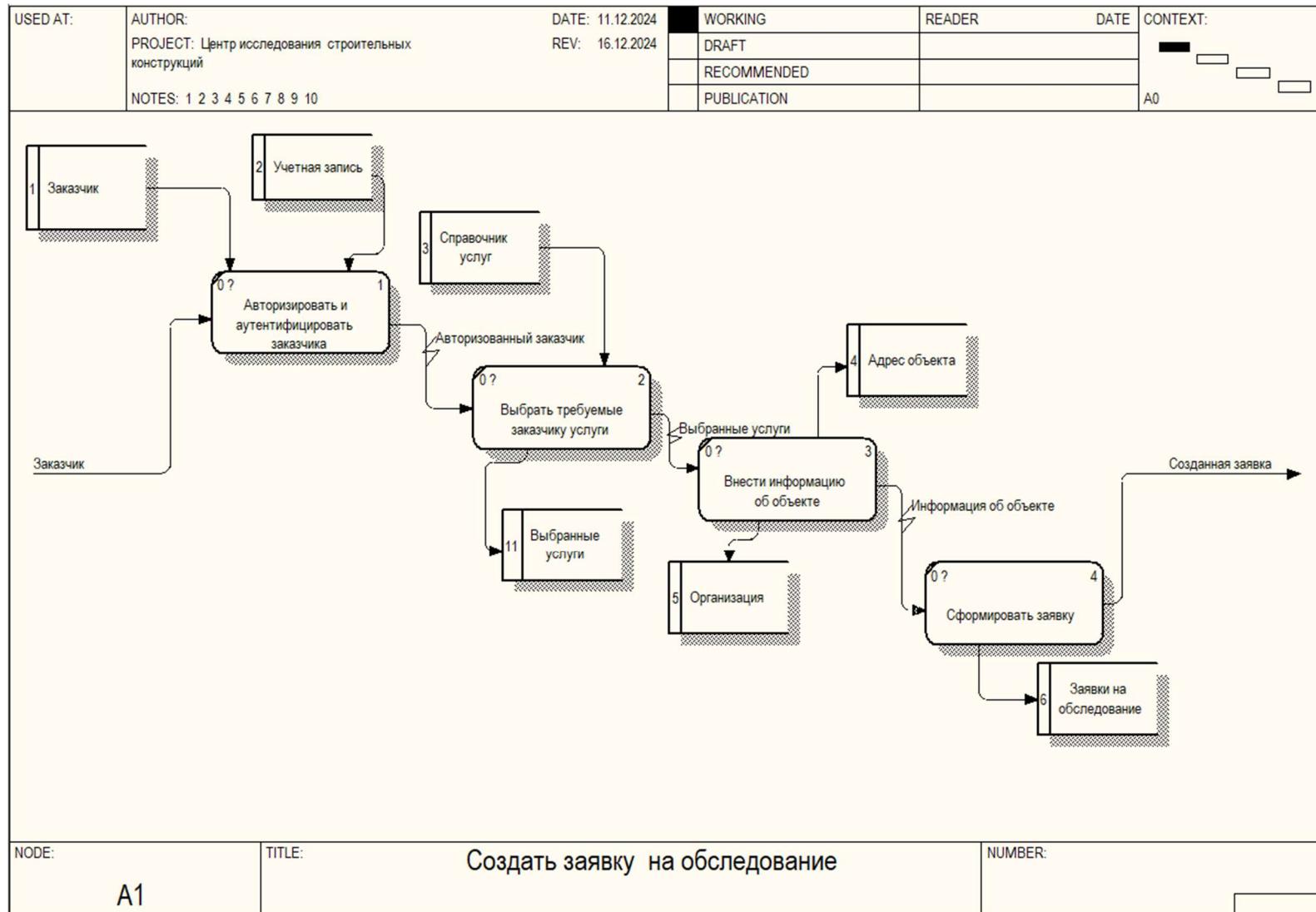


Рисунок 3 – Декомпозиция блока «Создать заявку на обследование»

Создание заявки на обследование подробнее можно рассмотреть с помощью следующих этапов:

- 1) авторизовать и аутентифицировать заказчика. Для подачи заявки на обследование заказчику необходимо авторизоваться и аутентифицироваться в системе. Данным процессом занимается сам заказчик;
- 2) выбрать требуемые заказчику услуги. После авторизации и аутентификации заказчика для подачи заявки ему необходимо выбрать требуемые услуги. Данным процессом занимается также заказчик при помощи справочника услуг, который хранит информацию об предоставляемых центром исследования услугах. Выбранные услуги заказчиком заносятся в хранилище выбранные услуги;
- 3) внести информацию об объекте. В данном блоке заказчик должен внести информацию об объекте, на котором будет производится обследование, он должен будет предоставить адрес и при наличии определенной организации сообщить о ней;
- 4) сформировать заявку. После того как заказчик выбрал требуемые услуги и внес всю информацию по объекту обследования производится формирование заявки на обследование.

На рисунке 4 показана декомпозиция второго уровня блока «Выполнить обследование и сформировать отчет».

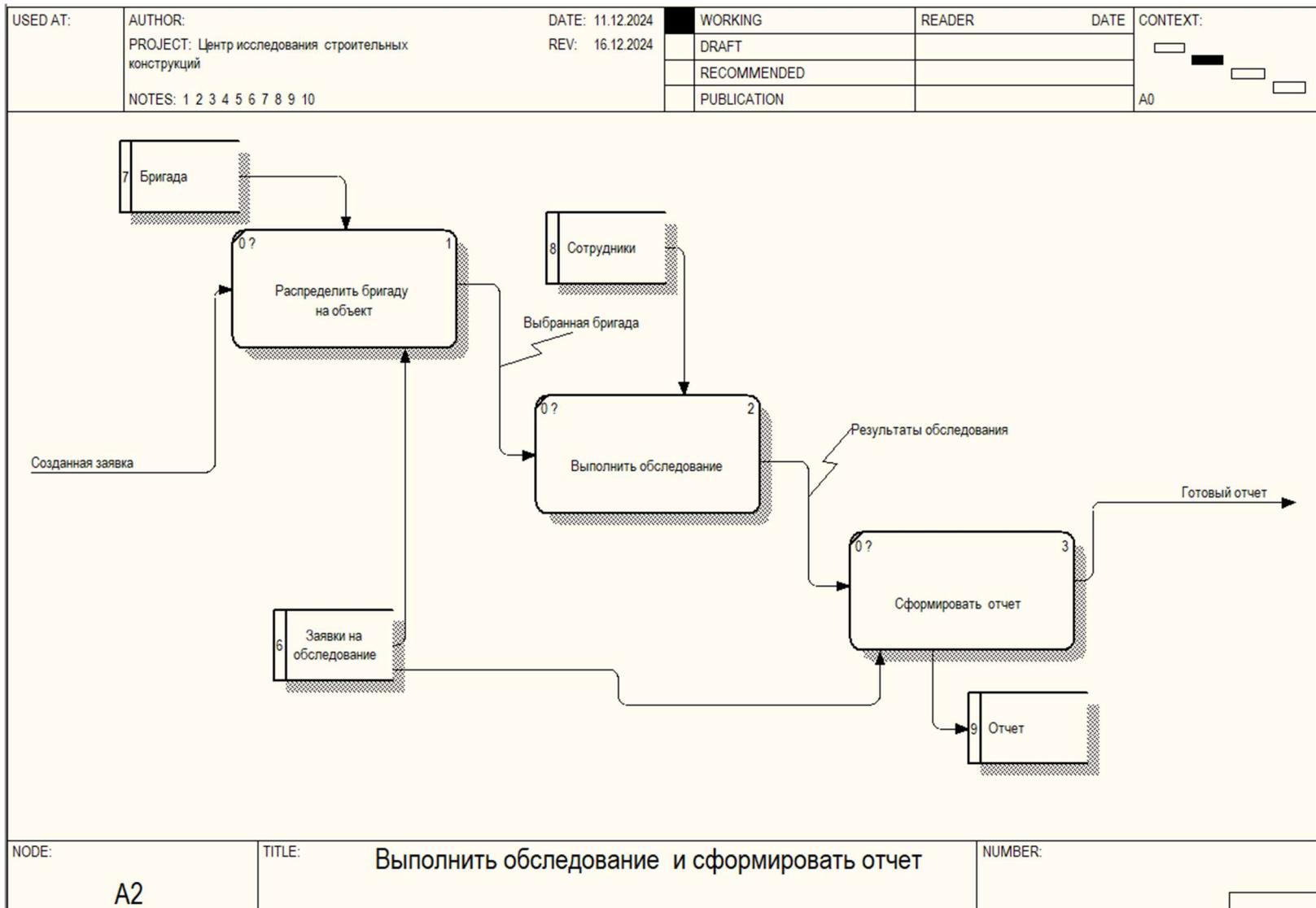


Рисунок 4 - Декомпозиция блока «Выполнить обследование и сформировать отчет»

Выполнение обследования и формирование отчета подробнее можно рассмотреть с помощью следующих этапов:

1) распределить бригаду на объект. Для проведения обследования необходимо выбрать бригаду, которая будет его выполнять. Выбор бригады должен проводится автоматически соответственно с графиком загруженности бригад который должен рассчитываться с учетом активных заявок;

2) выполнить обследование. После того как была выбрана бригада, которая будет выполнять данное обследование соответствующие сотрудники из данной бригады должны провести обследование;

3) сформировать отчет. После выполнения обследования соответствующий сотрудник из выбранной бригады должен сформировать отчет из результатов обследования.

На рисунке 5 показана декомпозиция второго уровня блока «Рассчитать договор».

USED AT:	AUTHOR: PROJECT: Центр исследования строительных конструкций NOTES: 1 2 3 4 5 6 7 8 9 10	DATE: 11.12.2024 REV: 16.12.2024	WORKING DRAFT RECOMMENDED PUBLICATION	READER	DATE	CONTEXT: A0
----------	------------------------------------------------------------------------------------------------	-------------------------------------	------------------------------------------------	--------	------	----------------

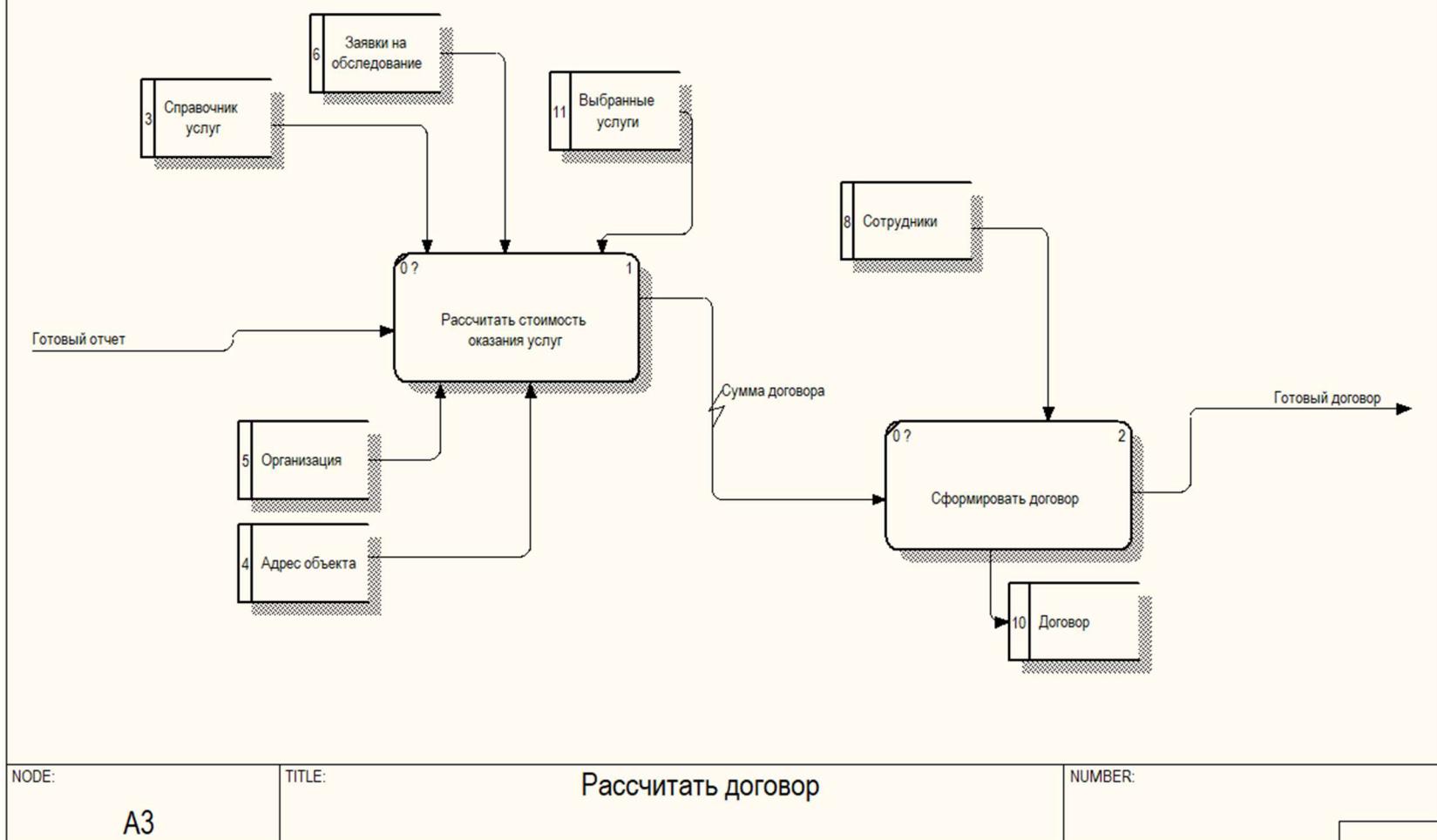


Рисунок 5 - Декомпозиция блока «Рассчитать договор»

Расчёт договора подробнее можно рассмотреть с помощью следующих этапов:

1) рассчитать стоимость оказания услуг. В данном блоке происходит расчет суммы, который заказчик должен будет оплатить за проведенное обследование. Она подсчитывается в соответствии с ценами выбранных услуг и выполненным объемом работы;

2) сформировать договор. После того как была рассчитана сумма за оказанные услуги формируется договор о представлении услуг заказчику.

На рисунке 6 показана декомпозиция второго уровня блока «Предоставить заказчику договор и отчет».

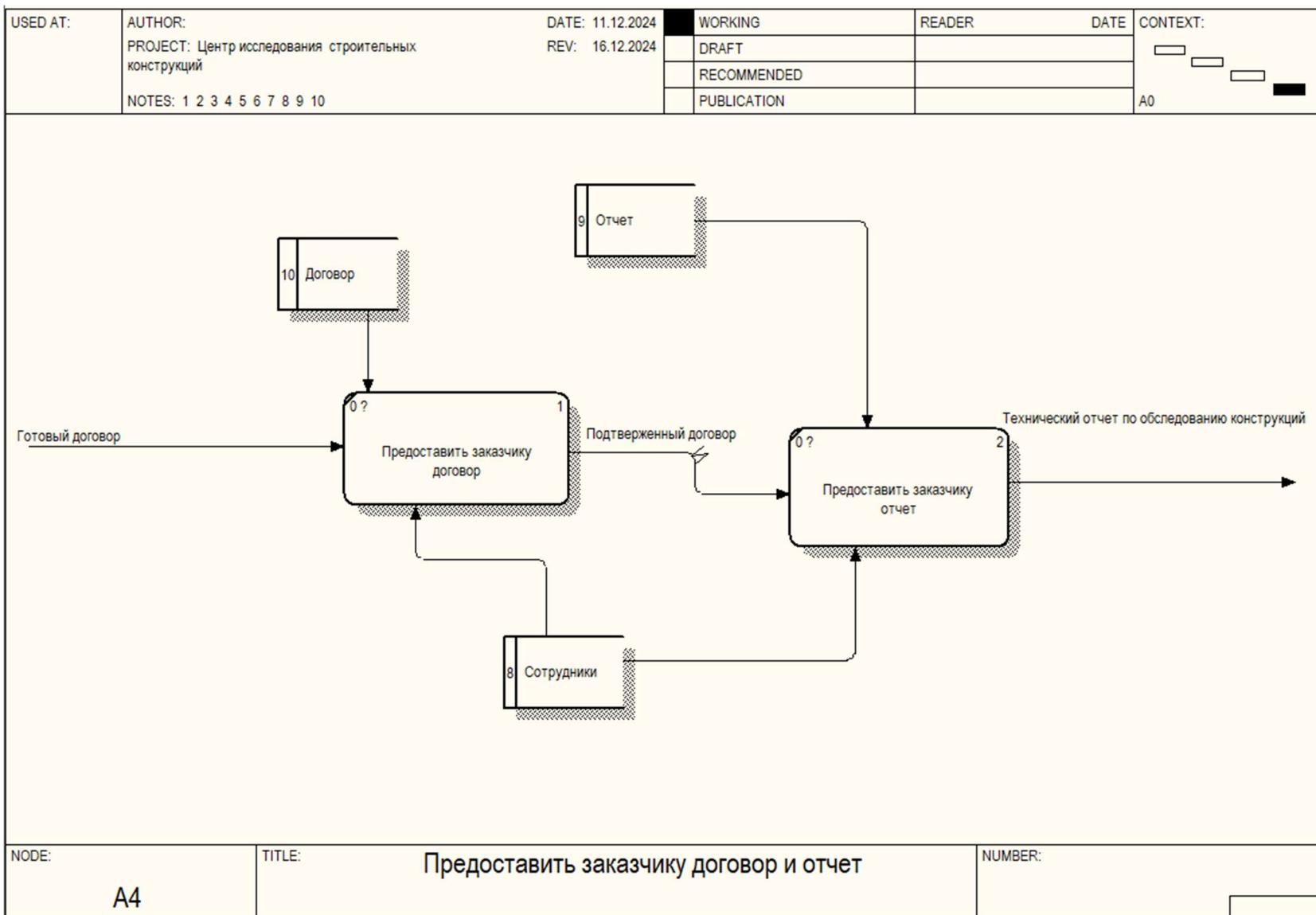


Рисунок 6 - Декомпозиция блока «Предоставить заказчику договор и отчет»

Предоставление заказчику договора и отчета подробнее можно рассмотреть с помощью следующих этапов:

- 1) предоставить заказчику договор. Заказчику предоставляется сформированный договор, в котором он может ознакомиться с суммой, которую ему необходимо будет оплатить;
- 2) предоставить заказчику отчет. После того как заказчик ознакомился с договором и подтвердил его ему предоставляется отчет о проведенном обследовании.

## 2.3 Разработка локальных концептуальных моделей

После моделирования системы по методологии IDEF0 следует формализованное описание предметной области в методологии IDEF1X.

Методология IDEF1X представляет собой инструмент для семантического моделирования данных, основанный на концепции "сущность-связь". Она используется для анализа и проектирования информационных систем, а также для создания логических моделей баз данных. Основные элементы IDEF1X — это сущности и связи между ними. Методология описывает логическую структуру данных, включая их атрибуты и взаимосвязи, что помогает в разработке непротиворечивой и оптимальной базы данных [9].

Сущность представляет собой множество реальных или абстрактных объектов, обладающих общими характеристиками. При помощи анализа dfd – диаграмм можно выявить ту информацию, которую нужно хранить в базе данных, на диаграммах потоков данных они представлены в виде «хранилищ». Данные хранилища используются для определения сущностей локальных концептуальных и логических моделей данных.

### 2.4.1 Определение атрибутов сущностей и их ключей

На основании проведенного анализа предметную область «Центр исследования строительных конструкций» можно описать следующими

сущностями: «Заказчик», «Учетная запись», «Сотрудник», «Должность», «Адрес», «Организация», «Объект обследования», «Заявка на обследование», «Бригада», «Каталог услуг», «Выбранные услуги», «Отчет об обследовании», «Договор». Результат представлен в таблице 1.

Таблица 1 – Список сущностей

№	Название	Значение
1	Заказчик	Информация о заказчиках
2	Сотрудник	Информация о сотрудниках
3	Уровень доступа	Содержит информацию обо всех возможных уровнях доступа
4	Учетная запись	Содержит информацию обо всех зарегистрированных в системе учетных записей, а также их уровень доступа
5	Должность	Информация о должностях, которые могут иметь сотрудники
6	Адрес	Содержит информацию обо всех адресах объектов обследования
7	Организация	Содержит информацию обо всех организациях, для которых происходили обследования
8	Объект обследования	Содержит информацию обо всех объектах обследования
9	Статус заявки	Содержит информацию о возможные статусы заявки
10	Заявка на обследование	Информация о заявках на обследование
11	Бригада	Информация о бригадах, занимающихся обследованием
12	Каталог услуг	Информация о предоставляемых услугах
13	Выбранные услуги	Содержит информацию о выбранных услугах в каждой заявке
14	Отчет об обследовании	Содержит информацию об отчете
15	Договор	Информацию о договорах

Описание атрибутов для каждой сущности приведены в таблицах 2-17.

Таблица 2 – Атрибуты сущности «Заказчик»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор заказчика	Числовой	Нет	Уникальный идентификатор
	ФИО	Строчный	Нет	Определяет ФИО заказчика
	Номер телефона	Строчный	Нет	Определяет номер телефона заказчика

Таблица 3 – Атрибуты сущности «Сотрудник»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор сотрудника	Числовой	Нет	Уникальный идентификатор
	ФИО	Строчный	Нет	Определяет ФИО сотрудника
	Номер телефона	Строчный	Нет	Определяет номер телефона сотрудника
Fk (foreign key)	Идентификатор должности	Числовой	Нет	Определяет занимаемую сотрудником должность
Fk (foreign key)	Идентификатор бригады	Числовой	Да	Определяет бригаду, в которой состоит сотрудник

Таблица 4 – Атрибуты сущности «Уровень доступа»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор уровня доступа	Числовой	Нет	Уникальный идентификатор
	Наименование уровня доступа	Строчный	нет	Содержит наименование уровня доступа

Таблица 5 – Атрибуты сущности «Учетная запись»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор учетной записи	Числовой	Нет	Уникальный идентификатор
	Почта	Строчный	Нет	Определяет почту владельца учетной записи
	Пароль	Строчный	Нет	Определяет пароль от учетной записи
Fk (foreign key)	Идентификатор уровня доступа	Числовой	Нет	Определяет уровень доступа учетной записи
Fk (foreign key)	Идентификатор заказчика	Числовой	Да	Определяет идентификатор заказчика если владельцем аккаунта является сотрудник
Fk (foreign key)	Идентификатор сотрудника	Числовой	Да	Определяет идентификатор сотрудника если владельцем аккаунта является не заказчик

Таблица 6 – Атрибуты сущности «Должность»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор должности	Числовой	Нет	Уникальный идентификатор
	Наименование должности	Строчный	Нет	Определяет наименование должности

Таблица 7 – Атрибуты сущности «Адрес»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор адреса	Числовой	Нет	Уникальный идентификатор
	Наименование города	Строчный	Нет	Определяет наименование города, в котором находится объект обследования
	Наименование улицы	Строчный	Нет	Определяет наименование улицы, на которой находится объект обследования
	Номер	Строчный	Нет	Определяет номер дома или строения на определенной улице

Таблица 8 – Атрибуты сущности «Организация»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор организации	Числовой	Нет	Уникальный идентификатор
	Наименование организации	Строчный	Нет	Содержит наименование организации, для которой проводится обследование
	ИНН	Строчный	Нет	Идентификационный номер налогоплательщика

Таблица 9 – Атрибуты сущности «Объект обследования»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор объекта	Числовой	Нет	Уникальный идентификатор
Fk (foreign key)	Идентификатор адреса	Числовой	Нет	Определяет по какому адресу находится объект обследования
Fk (foreign key)	Идентификатор организации	Числовой	Нет	Определяет к какой организации относится объект обследования
	Площадь объекта	Числовой	Нет	Определяет какую площадь занимает объект обследования

Таблица 11 – Атрибуты сущности «Статус заявки»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор статуса заявки	Числовой	Нет	Уникальный идентификатор
	Вид статуса	Числовой	Нет	Определяет вид статуса, который возможен в заявке

Таблица 12 – Атрибуты сущности «Заявка на обследование»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор заявки	Числовой	Нет	Уникальный идентификатор
Fk (foreign key)	Идентификатор заказчика	Числовой	Нет	Определяет какой заказчик оставил заявку на обследование

Продолжение таблицы 12

Fk (foreign key)	Идентификатор бригады	Числов ой	Нет	Определяет какая бригада была назначена на обследование
	Дата принятия заявки	Дата	Нет	Определяет, когда дату, когда была принята заявка
Fk (foreign key)	Идентификатор статуса заявки	Числов ой	Нет	Определяет этап, на котором находится заявка
	Дата начала обследования	Дата	Да	Определяет дату, когда начнется обследование
	Дата окончания обследования	Дата	Да	Определяет дату, когда закончится обследование

Таблица 13 – Атрибуты сущности «Бригада»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор бригады	Числов ой	Нет	Уникальный идентификатор
	Наименование бригады	Строо вый	Нет	Определяет наименование бригады

Таблица 14 – Атрибуты сущности «Каталог услуг»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор услуги	Числов ой	Нет	Уникальный идентификатор
	Наименование услуги	Строо вый	Нет	Определяет наименование услуги

Продолжение таблицы 14

	Стоимость услуги	Числовой	Нет	Определяет стоимость оказания услуги за единицу объема выполненной работы
	Единица измерения	Строочный	Нет	Определяет единицу измерения услуги
	Описание услуги	Строочный	Нет	Содержит информацию об услуги

Таблица 15 – Атрибуты сущности «Выбранные услуги»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор выбранной услуги	Числовой	Нет	Уникальный идентификатор
Fk (foreign key)	Идентификатор услуги	Числовой	Нет	Определяет выбранную услугу
Fk (foreign key)	Идентификатор заявки	Числовой	Нет	Определяет заявку, к которой относится выбранная услуга
	Объем	Числовой	Нет	Определяет объем, который требуется выполнить
	Стоимость за услугу	Числовой	Нет	Определяет общую стоимость за оказание выбранной услуги и указанный объем

Таблица 16 – Атрибуты сущности «Отчет об обследовании»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор отчета	Числовой	Нет	Уникальный идентификатор
Fk (foreign key)	Идентификатор заявки	Числовой	Нет	Определяет заявку, к которой относится отчет
Fk (foreign key)	Идентификатор сотрудника	Числовой	Нет	Определяет сотрудника, который сформировал отчет
	Отчет	Строчный	Нет	Содержит путь к файлу отчета

Таблица 17 – Атрибуты сущности «Договор»

Ключевое поле	Название	Тип данных	Допустимость NULL	Назначение
Pk (primary key)	Идентификатор договора	Числовой	Нет	Уникальный идентификатор
Fk (foreign key)	Идентификатор заявки	Числовой	Нет	Определяет заявку, которая относится к договору
Fk (foreign key)	Идентификатор отчета	Числовой	Нет	Определяет отчет, который относится к договору
Fk (foreign key)	Идентификатор сотрудника	Числовой	Нет	Определяет сотрудника, который сформировал договор
	Дата создания договора	Дата	Нет	Определяет дату создания договора
	Подтверждение	Логический	Нет	Определяет статус договора

## Продолжение таблицы 17

	Стоимость оказания услуг	Числовой	Нет	Определяет сумму, которую необходимо оплатить заказчику за оказанные услуги
--	--------------------------	----------	-----	-----------------------------------------------------------------------------

### 2.4.2 Выявление связей между сущностями

На основании описания предметной области и списка атрибутов из таблиц 2-13 описываются классы сущностей и их свойства, устанавливаются существующие связи между ними и приводится обоснование типов связей (1:1, 1:N и т. д.). Результат представлен в Таблице 18.

Таблица 18 – Список связей

№	Сущности, участвующие в связи	Тип связи	Обоснование
1	Заказчик – Учетная запись	1:1	У одного заказчика, может быть, одна учетная запись и у учетной записи может быть только один владелец
2	Сотрудник – Учетная запись	1:1	У одного сотрудника, может быть, одна учетная запись и у учетной записи может быть только один владелец
3	Уровень доступа – Учетная запись	1:N	У учетной записи может быть один уровень доступа, но может быть несколько учетных записей с одинаковым уровнем доступа
4	Бригада - Сотрудник	1:N	Один сотрудник может состоять в одной бригаде, но в бригаде может состоять несколько сотрудников

Продолжение таблицы 18

5	Должность - Сотрудник	1:N	Один сотрудник может иметь только одну должность, но несколько сотрудников могут иметь одинаковую должность
6	Адрес – Объект обследования	1:N	На одном адресе может быть несколько объектов обследования, но у одного объекта обследования может быть только один адрес
7	Организация – Объект обследования	1:N	У одной организации может быть несколько объектов обследования, но у одного объекта обследования может быть только одна организация
8	Заказчик – Заявка на обследование	1:N	У одного заказчика может быть несколько заявок на обследование, но у заявки на обследование может быть только один заказчик
9	Бригада – Заявка на обследование	1:N	У одной бригады может быть несколько заявок на обследование, но у заявки на обследование может быть только одна бригада
10	Статус заявки на обследование - Заявка на обследование	1:N	С одним статусом заявки на обследование может быть несколько заявок на обследование, но у заявки на обследование может быть только один статус
11	Объект – Заявка на обследование	1:N	У одного объекта обследования может быть несколько заявок на обследование, но у заявки на обследование может быть только один объект обследования

Продолжение таблицы 18

12	Каталог услуг – Выбранные услуги	1:N	У одной услуги из каталога, может быть, несколько выбранных услуг, но у одной выбранной услуги может быть только одна услуга из каталога
13	Заявка на обследование – Выбранные услуги	1:N	У одной заявки на обследование, может быть, несколько выбранных услуг, но у одной выбранной услуги может быть только одна заявка на объект
14	Заявка на обследование – Отчет об обследовании	1:1	У одной заявки, может быть, только один отчет об обследовании и у отчета об обследовании может быть только одна заявка на обследование
15	Заявка на обследование - Договор	1:1	У одной заявки, может быть, только один договор и у договора может быть только одна заявка на обследование
16	Отчет об обследовании - договор	1:1	У одного отчета об обследовании, может быть, только один договор и у договора может быть только один отчет об обследовании
17	Сотрудник - Отчет об обследовании	1:N	Один сотрудник может сделать несколько отчетов об обследовании, но у одного отчета об обследовании может быть только один сотрудник
18	Сотрудник - договор	1:N	Один сотрудник может сделать несколько договоров, но у одного договора может быть только один сотрудник

### **3      Разработка глобальных моделей**

#### **3.1    Разработка глобальной логической модели**

Глобальная логическая модель – представляет из себя слияние всех локальных логических моделей. Она описывает взаимосвязи между сущностями и их атрибутами, а также устанавливает правила работы с данными на концептуальном уровне. Глобальная логическая модель используется для обеспечения согласованности данных в масштабах всей информационной системы и формирования основы для дальнейшей реализации базы данных [10].

Разработанная глобальная модель показана на рисунке 7.

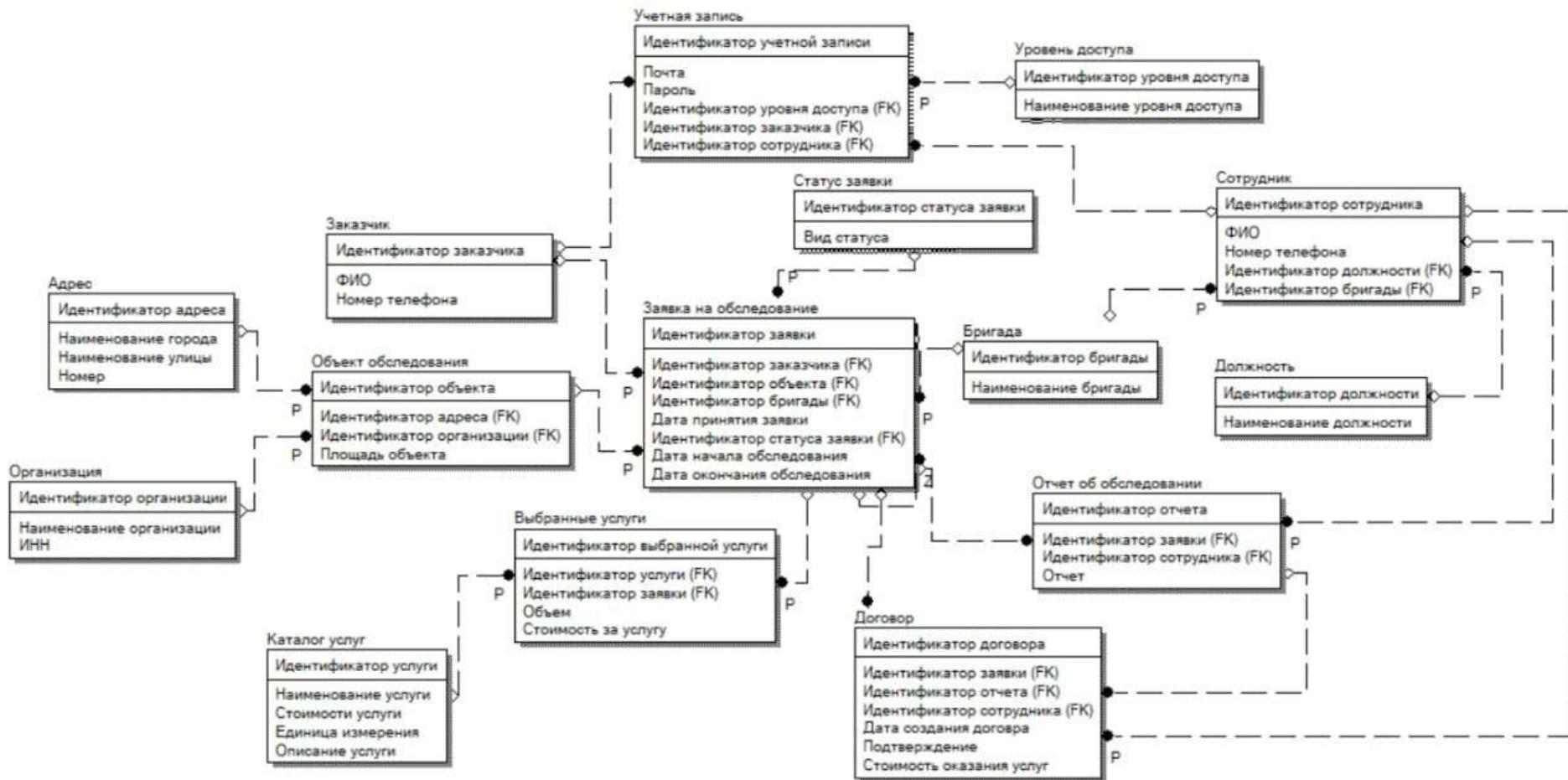


Рисунок 7 – Глобальная логическая модель данных

### **3.2 Разработка глобальной физической модели данных**

Физическая модель данных описывает структуру и расположение данных на физическом уровне их хранения. Она определяет реализацию баз данных в конкретной системе управления базами данных (СУБД), включая таблицы, их колонки, индексы, ограничения целостности и способы доступа к данным. Физическая модель разрабатывается с учетом технических требований, таких как производительность, объем хранимой информации, целостность данных и их безопасность [11].

В результате разработки получена следующая физическая модель для СУБД PostgreSQL. Разработанная глобальная модель показана на рисунке 8.

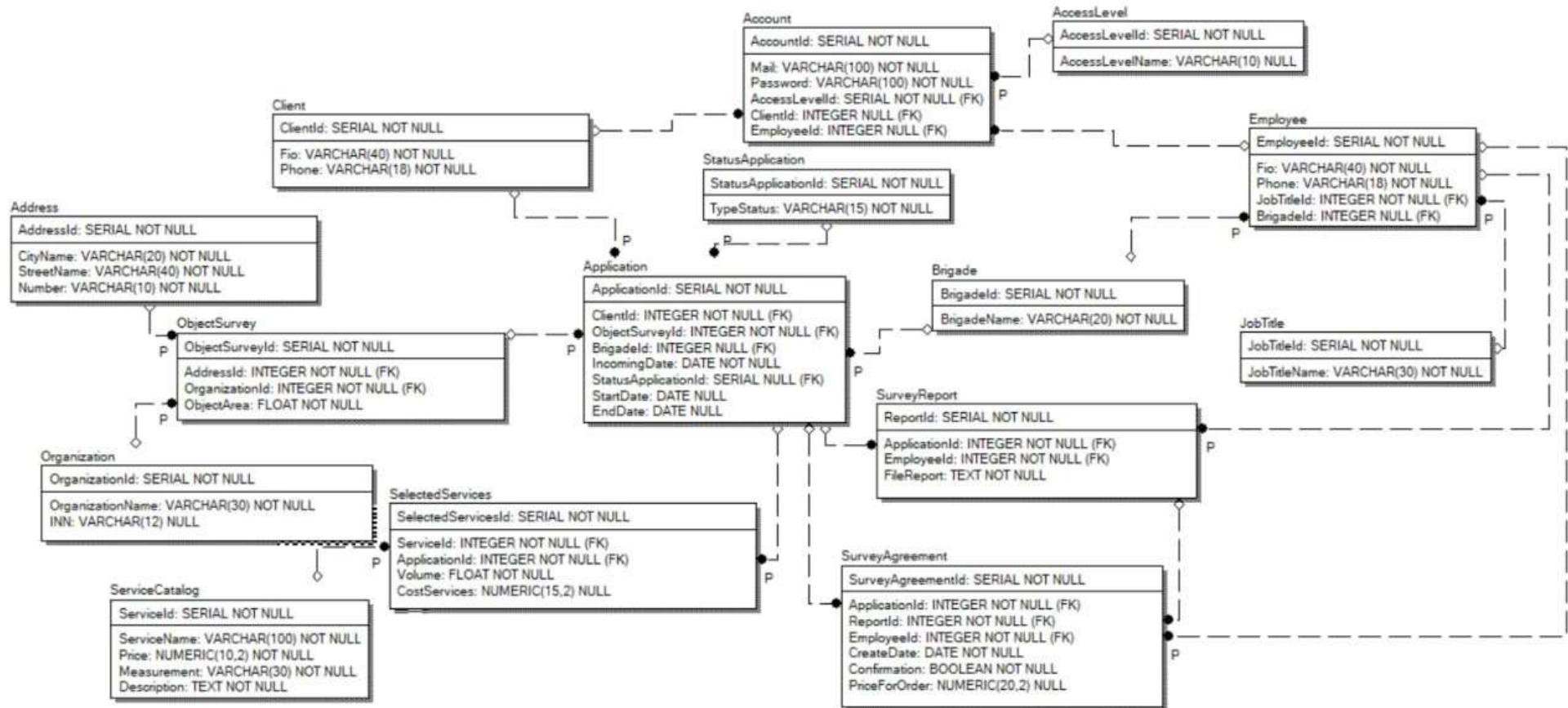


Рисунок 8 – Глобальная физическая модель данных

## **4      Детализация задания**

### **4.1    Проектирование таблиц [12]**

Исходя из п.3.2 физической модели данных необходимо создать следующие таблицы:

- 1)    заказчик:
  - ClientId (SERIAL, PK) – уникальный идентификатор заказчика (первичный ключ);
  - Fio (VARCHAR (40), NOT NULL) – фамилия, имя и отчество заказчика. Длина ограничена 40 символами;
  - Phone (VARCHAR (18), UNIQUE, NOT NULL) – контактный номер телефона заказчика. Значение должно быть уникальным;
- 2)    Уровень доступа:
  - AccessLevelId (SERIAL, PK) - уникальный идентификатор учетной уровня доступа (первичный ключ);
  - AccessLevelName (VARCHAR (10), NOT NULL) – наименование уровня доступа. Длина ограничена 10 символами;
- 3)    учетная запись:
  - AccountId (SERIAL, PK) – уникальный идентификатор учетной записи (первичный ключ);
  - Mail (VARCHAR (100), UNIQUE, NOT NULL) – почта к которой привязан аккаунт. Длина ограничена 100 символами. Значение должно быть уникальным;
  - Password (VARCHAR (100), NOT NULL) – пароль по которому происходит авторизация в аккаунт. Длина ограничена 100 символами;
  - AccessLevelId (VARCHAR (8), NOT NULL, FK → AccessLevel(AccessLevelId), ON DELETE SET NULL) – идентификатор уровня доступа учетной записи (внешний ключ, ссылается на AccessLevel);

- ClientId (INTEGER, UNIQUE, FK → Client(ClientId), ON DELETE CASCADE) – идентификатор клиента если аккаунт принадлежит клиенту (внешний ключ, ссылается на Client);
  - EmployeeId (INTEGER, UNIQUE, FK → Employee(EmployeeId), ON DELETE CASCADE) – идентификатор сотрудника если аккаунт принадлежит сотруднику (внешний ключ, ссылается на Employee);
- 4) сотрудник:
- EmployeeId (SERIAL, PK) – уникальный идентификатор сотрудника (первичный ключ);
  - Fio (VARCHAR (40), NOT NULL) – фамилия, имя и отчество сотрудника. Длина ограничена 40 символами;
  - Phone (VARCHAR (18), UNIQUE, NOT NULL) – контактный номер телефона сотрудника. Значение должно быть уникальным;
  - JobTitleId (INTEGER, NOT NULL, FK → JobTitle(JobTitleId), ON DELETE SET NULL) – идентификатор должности сотрудника (внешний ключ, ссылается на JobTitle);
  - BrigadeId (INTEGER, FK → Brigade(BrigadeId), ON DELETE SET NULL) – идентификатор бригады в которой состоит сотрудник (внешний ключ, ссылается на Brigade);
- 5) должность:
- JobTitleId (SERIAL, PK) - уникальный идентификатор должности (первичный ключ);
  - JobTitleName (VARCHAR (30), UNIQUE, NOT NULL) – наименование должности. Длина ограничена 30 символами;
- 6) адрес:
- AddressId (SERIAL, PK) - уникальный идентификатор адреса (первичный ключ);

- CityName (VARCHAR (20), NOT NULL) – название города.

Длинна ограничена 20 символами;

- StreetName (VARCHAR (40), NOT NULL) – название улицы.

Длинна ограничена 20 символами;

- Number (VARCHAR (10), NOT NULL) – номер строения. Длинна ограничена 10 символами;

7) организация:

- OrganizationId (SERIAL, PK) – уникальный идентификатор организации (первичный ключ);

• OrganizationName (VARCHAR (30), NOT NULL) – наименования организации. Длинна ограничена 30 символами;

- INN (VARCHAR (18), UNIQUE, NOT NULL) – идентификационный номер налогоплательщика. Длинна ограничена 18 символами;

8) объект обследования:

- ObjectSurveyId (SERIAL, PK) – уникальный идентификатор объекта обследования (первичный ключ);

• AddressId (INTEGER, NOT NULL, FK → Address(AddressId), ON DELETE SET NULL) – уникальный идентификатор адреса на котором находится обследуемый объект обследования (внешний ключ, ссылается на Address);

- OrganizationId (INTEGER, NOT NULL, FK → Organization(OrganizationId), ON DELETE SET NULL) – уникальный идентификатор организации которой принадлежит объект обследования (внешний ключ, ссылается на Organization);

• ObjectArea (FLOAT, NOT NULL) – площадь которую занимает объект;

9) статус заявки на обследование:

- StatusApplicationId (SERIAL, PK) – уникальный идентификатор статуса заявки на обследование (первичный ключ);
  - TypeStatus (varchar(15), NOTNULL) – вид статуса заявки на обследование. Длина ограничена 15 символами;
- 10) заявка на обследование:
- ApplicationId (SERIAL, PK) – уникальный идентификатор заявки на обследование (первичный ключ);
  - ClientId (INTEGER, NOT NULL, FK → Client(ClientId), ON DELETE SET NULL) – идентификатор клиента который подал заявку на обследование (внешний ключ, ссылается на Client);
  - ObjectSurveyId (INTEGER, NOT NULL, FK → ObjectSurvey(ObjectSurveyId), ON DELETE SET NULL) – идентификатор объекта обследования который необходимо обследовать клиенту (внешний ключ, ссылается на ObjectSurvey);
  - BrigadeId (INTEGER, FK → Brigade(BrigadeId), ON DELETE SET NULL) – идентификатор бригады которая будет выполнять обследование(внешний ключ, ссылается на Brigade);
  - IncomingDate (Date), NOT NULL – дата принятия заказа;
  - StatusApplicationId (INTEGER, FK → StatusApplication(StatusApplicationId), ON DELETE SET NULL) – идентификатор статуса заявки показывающий в каком состоянии находится заявка;
  - StrateDate (Date, CHECK (IncomingDate < StrateDate)) – дата начала обследования;
  - EndDate (Date, CHECK (StrateDate < EndDate)) – дата окончания обследования;
- 11) бригада:
- BrigadeId (SERIAL, PK) – уникальный идентификатор бригады (первичный ключ);

- BrigadeName (VARCHAR (20), NOT NULL) – наименование бригады. Длина ограничена 20 символами;

12) каталог услуг:

- ServiceId (SERIAL, PK) – уникальный идентификатор услуги (первичный ключ);

- ServiceName (VARCHAR (100), NOT NULL) – наименование услуги. Длина ограничена 100 символами;

- Price (NUMERIC (10,2), NOT NULL) – цена на оказанию услуги за единицу измерения;

- Measurement (VARCHAR (30), NOT NULL) – мера в которой измеряется услуги. Длина ограничена 30 символами;

- Description (TEXT, NOT NULL) – описание услуги;

13) выбранные услуги:

- SelectedServicesId (SERIAL, PK) – уникальный идентификатор выбранной услуги (первичный ключ);

- ServiceId (INTEGER, NOT NULL, FK → Service(ServiceId), ON DELETE SET NULL, ON UPDATE CASCADE) – идентификатор услуги которая требуется клиенту(внешний ключ, ссылается на Service);

- ApplicationId (INTEGER, NOT NULL, FK → Application(ApplicationId), ON DELETE CASCADE) – идентификатор заявки на обследование к которой относится выбранная услуга (внешний ключ, ссылается на Application);

- Volume (FLOAT, NOT NULL) – объем выполняемой работы;

- CostServices (NUMERIC (15,2)) – цена оказания услуги за указанный объем работы;

14) отчет об обследовании:

- ReportId (SERIAL, PK) – уникальный идентификатор отчета об обследовании (первичный ключ);

- ApplicationId (INTEGER, NOT NULL, FK → Application (ApplicationId), ON DELETE RESTRICT) – идентификатор заявки на обследование к которой относится отчет об обследовании (внешний ключ, ссылается на Application);
  - EmployeeId (INTEGER, NOT NULL, FK → Employee (EmployeeId), ON DELETE SET NULL) – идентификатор сотрудника который сделал отчет об обследовании (внешний ключ, ссылается на Employee);
  - FileReport (TEXT, NOT NULL) – путь по которому лежит отчет;
- 15) договор:
- SurveyAgreementId (SERIAL, PK) - уникальный идентификатор договора (первичный ключ);
  - ApplicationId (INTEGER, NOT NULL, FK → Application (ApplicationId), ON DELETE RESTRICT) – идентификатор заявки на обследование к которой составляется договор (внешний ключ, ссылается на Application);
  - ReportId (INTEGER, NOT NULL, FK → SurveyReport (ReportId), ON DELETE SET NULL) – идентификатор отчета об обследовании (внешний ключ, ссылается на SurveyReport);
  - EmployeeId (INTEGER, NOT NULL, FK → Employee (EmployeeId), ON DELETE SET NULL) – идентификатор сотрудника (внешний ключ, ссылается на Employee);
  - CreateDate (DATE, NOT NULL) – дата создания;
  - Confirmation (BOOLEAN, NOT NULL) – флаг подтверждения клиента;
  - PriceForOrder (NUMERIC (20,2) – цена за всю проделанную работу.

## **4.2 Проектирование хранимых функций**

### **4.2.1 Проектирование функций**

Функции формируют набор строк с указанными столбцами из одной или нескольких таблиц. Они дают возможность комбинировать данные из разных запросов и выводить их в виде единого результата [13]. На основе выделенных бизнес-процессов п.1.2 разработаем подобные функции:

1) функция позволяет анализировать все заявки клиента. Она возвращает таблицу с данными об адресе объекта обследования, организации, наименование исполнительной бригады, дате подачи заявки, статусе заявки, дате начала и окончания обследования и дате формирования договора. Результаты сортируются по дате подачи заявки что позволяет отслеживать клиенту поданные и завершенные заявки. Эта функция полезна для анализа времени исполнения заявки на различных объектах обследования клиента;

2) функция анализирует принесенную прибыль от каждой организации за определенный временной период. Она возвращает таблицу с данными об наименовании организации и её ИНН, и сумму стоимости оказания услуг всех договоров, заключенных с данной организацией. Эта функция полезна для оценки прибыли, принесенных от организаций;

3) функция анализирует принесенную прибыль от каждой исполнительной бригады за определенный временной период. Она возвращает таблицу с данными об наименовании исполнительной бригады и сумму стоимости оказания услуг всех договоров исполнителями которых является данная бригада. Эта функция полезна для оценки эффективности работы сотрудников, состоящих в исполнительных бригадах и принятия управленческих решений на основе их производительности;

### **4.2.2 Проектирование хранимых процедур**

Процедуры применяются для выполнения сложных алгоритмов обработки данных. В отличие от селективных, они могут либо не возвращать

строки вообще, либо выдавать результат в виде одной строки через выходные параметры [13]. На основе бизнес-процессов из п.1.2 разработаем соответствующие функции:

1) процедура выполняет несколько шагов для добавления нового заказчика:

- добавляется новый заказчик в таблицу Clients. Вставляются данные об клиенте и возвращается его уникальный идентификатор, который сохраняется в переменной reg\_clientid;
- добавляется новая учетная запись заказчика в таблице Account. Вставляется уникальный идентификатор заказчика, а также его почта и пароль, уровень доступа ставиться автоматически как client;
- если на каком-либо этапе возникает ошибка, все изменения откатываются, и выдается исключение с сообщением об ошибки.

Данная функция позволяет добавить нового клиента и создать новую учетную запись;

2) процедура выполняет несколько шагов для добавления новой заявки:

- клиент вводит свой номер телефона наименование и ИНН организации а также адрес объекта обследования и его площадь, если фио не найдено выдается ошибка, если организация не найдена тогда она добавляется в таблицу Organization и возвращается её уникальный идентификатор, который сохраняется в переменной new\_organization, если адрес не найден тогда он добавляется в таблицу Address и возвращается её уникальный идентификатор, который сохраняется в переменной new\_address, после этих проверок проверяется имеется ли объект обследования в таблице ObjectSurvey, если нет то он добавляется используя идентификаторы добавленной организации и добавленного адреса, а также площади объекта обследования;

- добавляется новая заявка в таблице Application вставляется уникальный идентификатор клиента и уникальный идентификатор объекта обследования;
- если на каком-либо этапе возникает ошибка, все изменения откатываются, и выдается исключение с сообщением об ошибки.

Данная функция позволяет добавить новую заявку, а также новую организацию, адрес, объект обследования и выбранные услуги;

3) процедура выполняет несколько шагов для добавления новой выбранной услуги:

- заказчик вводит свой номер телефона, дату подачи заявки, ИНН организации, адрес объекта обследования, наименование выбранной услуги и объем требуемой работы;
- если на каком-либо этапе возникает ошибка, все изменения откатываются, и выдается исключение с сообщением об ошибки.

Данная функция позволяет добавить выбранную услугу.

## 4.3 Проектирование триггеров [13]

### 4.3.1 Проектирование триггеров бизнес-требований

Реализуем бизнес-процессы и бизнес-требования посредством установленные в пунктах 1.2 и 1.3 по средствам разработки триггеров:

- 1) при добавлении или обновлении выбранной услуги необходимо автоматически рассчитывать стоимость её оказания, исходя из установленной стоимости за единицу объема и указанного объема;
- 2) при создании договора подсчитывать сумму оказанных услуг;
- 3) при создании новой заявки автоматически выбирать исполнительную бригаду.

### 4.3.2 Проектирование триггеров бизнес-ограничений

Реализуем бизнес-ограничения, установленные в пункте 1.4 посредством разработки триггеров:

- 1) отчет не может быть добавлен если заявки не является законченной;
- 2) договор не может быть добавлен если заявки не является законченной и отчет об обследовании не был добавлен;
- 3) нельзя установить дату формирования договора на более ранний срок чем дату проведения обследования.

#### **4.3.3 Проектирование триггера ограничений целостности**

Реализуем бизнес-ограничение, установленное в пункте 1.4 посредством разработки триггера. Нельзя удалить отчет если связанный договор подтвердил клиент.

#### **4.3.4 Проектирование триггера ограничения действия**

Реализуем бизнес-ограничения, установленные в пункте 1.4 посредством разработки триггеров:

- 1) нельзя добавить отчет об обследовании если сотрудник не состоит в бригаде, которая исполняла заявку;
- 2) нельзя удалить договор если его подтвердил клиент.

#### **4.3.5 Проектирование триггера DDL**

DDL-триггер будет создан как дополнительный уровень защиты базы данных, предотвращающие удаление таблиц всеми пользователями, кроме postgres.

#### **4.3.6 Проектирование триггера БД**

Триггер базы данных будет создан как дополнительный уровень защиты базы данных, записывающий в таблицу userlog имя пользователя и время его подключения к БД.

### **4.4 Разработка реализации политики безопасности**

Администратор – сотрудник, который контролирует работу системы. Роль должна обладать правами:

- полные права (CREATE, SELECT, INSERT, UPDATE, DELETE) на все таблицы;
- доступ ко всем функциям и триггерам;
- возможность создавать и назначать права другим пользователям.

Сотрудник – человек, который осуществляет управление заявками, договорами и отчетами, а также должен иметь возможность просматривать клиентов, выбранные услуги, услуги, организации, адреса, объекты обследования и бригады. Роль должна обладать правами:

- разрешение на работу с заявками (SELECT, INSERT, UPDATE в Application, SurveyReport, SurveyAgreement);
- просмотр клиентов, выбранных услуг, услуг, организаций, адресы, объекты обследования и бригады (SELECT в Client, SelectedServices, ServiceCatalog, Organiztion, Address, ObjectSurvey, Brigade);
- использование хранимой функции для анализа принесенной прибыли от каждой исполнительной бригады;
- использование хранимой функции для анализа принесенной прибыли от каждой организации.

Заказчик – человек, осуществляющий формирование заявки, добавление организаций, адресов, объектов обследования, а также иметь возможность просматривать услуги, выбранные услуги, которые относятся к его заявкам, заявки оставленные им, отчеты относящиеся к его заявкам и договора, относящиеся к его заявкам. Роль должна обладать правами:

- разрешение на работу с адресами и организациями (SELECT, INSERT, UPDATE в Organization, Address);
- разрешение на добавление новых заявок и выбранных услуг (INSERT в Application, SelectedServices);

- разрешение на просмотр каталога услуг, заявок, договоров и отчетов об исследовании (SELECT в ServiceCatalog, Application, SurveyAgreement, SurveyReport);
- использования хранимой функции для анализа заявок заказчика;
- использование хранимой функции для добавления нового клиента и аккаунта;
- использование хранимой функции для создания новой заявки;
- использование хранимой функции для добавления выбранной услуги.

## 5 Скрипт базы данных

### 5.1 Скрипты создания базы данных и таблиц базы данных

```
CREATE DATABASE "Center"
    ENCODING 'UTF8'
    LC_COLLATE 'en_US.UTF-8'
    LC_CTYPE 'en_US.UTF-8'
    TEMPLATE template0;

-- Создание таблицы Заказчик
create table Client(
    ClientId SERIAL primary key,
    Fio VARCHAR (40) NOT NULL,
    Phone VARCHAR (18) unique NOT NULL
);

-- Создание таблицы Уровень доступа
create table AccessLevel(
    AccessLevelId SERIAL primary key,
    AccessLevelName VARCHAR (10) NOT null
);

-- Создание таблицы Учетная запись
create table Account(
    AccountId SERIAL primary key,
    Mail VARCHAR (100) unique NOT NULL,
    Password VARCHAR (100) NOT NULL,
    AccessLevelId int NOT null REFERENCES AccessLevel(AccessLevelId) ON DELETE SET
NULL,
    ClientId int unique REFERENCES Client(ClientId) ON DELETE CASCADE,
    Employeeid int unique REFERENCES Employee(EmployeeId) ON DELETE CASCADE
);

-- Создание таблицы Должность
create table JobTitle(
    JobTitleId Serial primary key,
    JobTitleName VARCHAR (30) NOT NULL
);

-- Создание таблицы Бригада
create table Brigade(
    BrigadeId Serial primary key,
    Fio VARCHAR (40) NOT null,
    Phone VARCHAR (18) unique NOT null
);

-- Создание таблицы Сотрудник
create table Employee(
    Employeeid Serial primary key,
    Fio VARCHAR (40) NOT NULL,
    Phone VARCHAR (18) unique NOT NULL,
    JobTitleId int REFERENCES JobTitle(JobTitleId) ON DELETE SET NULL,
    BrigadeId int REFERENCES Brigade(BrigadeId) ON DELETE SET NULL
);

-- Создание таблицы Адрес
```

```

create table Address(
    AddressId Serial primary key,
    CityName VARCHAR (20) NOT NULL,
    StreetName VARCHAR (40) NOT NULL,
    Number VARCHAR (10) NOT NULL
);

-- Создание таблицы Организация
create table Organization(
    OrganizationId Serial primary key,
    OrganizationName VARCHAR (30) NOT NULL,
    INN VARCHAR (12) unique NOT NULL
);

-- Создание таблицы Объект обследования
create table ObjectSurvey(
    ObjectSurveyId Serial primary key,
    AddressId int REFERENCES Address(AddressId) ON DELETE SET NULL,
    OrganizationId int REFERENCES Organization(OrganizationId) ON DELETE SET NULL,
    ObjectArea FLOAT NOT NULL
);

-- Создание Статус заявки на обследование
create table StatusApplication(
    StatusApplicationId Serial primary key,
    TypeStatus VARCHAR (15) NOT NULL
);

-- Создание таблицы Заявка на обследование
create table Application(
    ApplicationId Serial primary key,
    ClientId int not null REFERENCES Client(ClientId) ON DELETE SET NULL,
    ObjectSurveyId int not null REFERENCES ObjectSurvey(ObjectSurveyId) ON DELETE
SET null,
    BrigadeId int REFERENCES Brigade(BrigadeId) ON DELETE SET null,
    IncomingDate date not null default (current_date),
    StatusApplicationId int not null REFERENCES
StatusApplication(StatusApplicationId) ON DELETE SET null,
    StarteDate date check (IncomingDate<StarteDate),
    EndDate date check (StarteDate<EndDate)
);

-- Создание таблицы Каталог услуг
create table ServiceCatalog(
    ServiceId Serial primary key,
    ServiceName VARCHAR (100) NOT null,
    Price NUMERIC(10,2) NOT null,
    Measurement varchar (30) NOT null,
    Description text not NULL
);

-- Создание таблицы Выбранные услуги
create table SelectedServices(
    SelectedServicesId Serial primary key,
    ServiceId int REFERENCES ServiceCatalog(ServiceId) ON DELETE SET null on
update cascade,
    ApplicationId int REFERENCES Application(ApplicationId) ON DELETE CASCADE,
    Volume FLOAT NOT null,
    CostServices NUMERIC(15,2)
);

```

```

);

-- Создание таблицы Отчет об обследовании
create table SurveyReport(
    ReportId Serial primary key,
    ApplicationId int REFERENCES Application(ApplicationId) ON DELETE RESTRICT,
    EmployeeId int REFERENCES Employee(EmployeeId) ON DELETE SET null,
    FileReport text not NULL
);

-- Создание таблицы Договор
create table SurveyAgreement(
    SurveyAgreementId Serial primary key,
    Applicationid int REFERENCES Application(ApplicationId) ON DELETE RESTRICT,
    ReportId int REFERENCES SurveyReport(ReportId) ON DELETE set null,
    EmployeeId int REFERENCES Employee(EmployeeId) ON DELETE SET null,
    CreateDate date not null default(current_date),
    Confirmation BOOLEAN not null default false,
    PriceForOrder NUMERIC (20,2)
);

```

## 5.2 Скрипты данных

```

insert into client(FIO,Phone) values('Кузьмин Сергей Иванович','8(905)584-98-94');
insert into client(FIO,Phone) values('Евсеев Адам Романович','8(910)952-05-13');
insert into client(FIO,Phone) values('Пахомова Алина Михайловна','8(950)005-31-93');
insert into client(FIO,Phone) values('Мальцев Константин Артёмович','8(942)507-57-05');
insert into client(FIO,Phone) values('Романова Валентина Александровна','8(952)174-27-75');

insert into AccessLevel(AccessLevelName) values('Client');
insert into AccessLevel(AccessLevelName) values('Admin');
insert into AccessLevel(AccessLevelName) values('Employee');

Insert into JobTitle(JobTitleName) values('Директор');
Insert into JobTitle(JobTitleName) values('Главный инженер проектировщик');
Insert into JobTitle(JobTitleName) values('Инженер проектировщик');
Insert into JobTitle(JobTitleName) values('Менеджер');

insert into StatusApplication(TypeStatus) values('Принята');
insert into StatusApplication(TypeStatus) values('Выполняется');
insert into StatusApplication(TypeStatus) values('Закончена');

INSERT INTO ServiceCatalog(ServiceName, Price, Measurement, Description) VALUES
('Составление отчета', 10000.00, 'ШТ', 'Наши специалисты составят отчет, в котором будут указаны проблемные места конструкции, а также рекомендации по их устранению');
INSERT INTO ServiceCatalog(ServiceName, Price, Measurement, Description) VALUES
('Обследование технического состояния несущих и ограждающих конструкций', 350.00,
'M2', 'Проводится детальное обследование несущих и ограждающих конструкций здания на предмет повреждений, деформаций и нарушений нормативов');
INSERT INTO ServiceCatalog(ServiceName, Price, Measurement, Description) VALUES
('Поиск и обследование технического состояния инженерных коммуникаций', 400.00, 'M2',
'Выполняется поиск и диагностика состояния внутренних коммуникаций: водоснабжения, канализации, электрики и вентиляции');
INSERT INTO ServiceCatalog(ServiceName, Price, Measurement, Description) VALUES
('Обследование технического состояния инженерных систем', 450.00, 'M2', 'Оценивается')

```

```

общее техническое состояние инженерных систем здания с предоставлением экспертного
заключения');
INSERT INTO ServiceCatalog(ServiceName, Price, Measurement, Description) VALUES
('Обмерные работы', 200.00, 'M2', 'Проводятся точные обмеры помещения или здания с
составлением чертежей и схем для последующего проектирования или ремонта');

insert into brigade(BrigadeName) values('Бригада №1');
insert into brigade(BrigadeName) values('Бригада №2');

insert into employee(Fio,Phone,JobTitleId,BrigadeId)values ('Ермолов Леон
Маркович','8(910)293-00-97',1,null);
insert into employee(Fio,Phone,JobTitleId,BrigadeId)values ('Симонов Николай
Егорович','8(965)867-22-69',2,1);
insert into employee(Fio,Phone,JobTitleId,BrigadeId)values ('Рябов Никита
Макарович','8(971)301-14-78',2,2);
insert into employee(Fio,Phone,JobTitleId,BrigadeId)values ('Григорьев Илья
Кириллович','8(944)991-09-27',3,1);
insert into employee(Fio,Phone,JobTitleId,BrigadeId)values ('Лобанов Виктор
Евгеньевич','8(956)295-85-89',3,2);
insert into employee(Fio,Phone,JobTitleId,BrigadeId)values ('Уткина Лея
Дмитриевна','8(985)874-96-44',4,null);

insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('Kyzmin1985@mail.ru','KyzminKyzmin',1,1,null);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('Evseev1990@yandex.ru','Evseev123',1,2,null);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('Alina41001@gmail.ru','Paxomova3012',1,3,null);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('Maltsev19885@mail.ru','Maltsev1988',1,4,null);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('Romanova62@yandex.ru','RomanovaRomanova',1,5,null);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('CentrIskIm@mail.ru','0emDHdZYztVH6pDw7u',2,null,1);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('NikolaCim@yandex.ru','NikolaCim1989',3,null,2);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('RabovNikita@yandex.ru','Nikitata',3,null,3);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('Grigrogev@gmail.ru','Grigrogev991',3,null,4);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('Lobanov2000@gmail.ru','Lobanov1100',3,null,5);
insert into Account(Mail,"password",accesslevelid,clientid,employeeid)
values('YtkinaLei@mail.ru','YtkinaLeiYtkinaLei',3,null,6);

insert into address(cityname,streetname,"number") values('г.Рязань','ул.
Интернациональная','д.1');
insert into address(cityname,streetname,"number") values('г.Рязань','ул.
Прижелезнодорожная','д.52');
insert into address(cityname,streetname,"number") values('г.Рязань','Московское
шоссе','д.5А');
insert into address(cityname,streetname,"number") values('г.Рязань','ул.
Бирюзова','д.28А');
insert into address(cityname,streetname,"number") values('г.Рязань','ул.
Интернациональная','д.23');
insert into address(cityname,streetname,"number") values('г.Рязань','шоссе
Солотчинское','д.11'));

```

```

insert into organization(organizationname,inn) values('Строительная
компания','0420520521');
insert into organization(organizationname,inn) values('Русская кожа','0602065848');
insert into organization(organizationname,inn) values('тц. Барс','6227009810');
insert into organization(organizationname,inn) values('Прайм','6234203335');
insert into organization(organizationname,inn) values('Круиз','6234085000');

insert into objectsurvey(addressid,organizationid,objectarea) values(1,1,15000.00);
insert into objectsurvey(addressid,organizationid,objectarea) values(2,2,160000.00);
insert into objectsurvey(addressid,organizationid,objectarea) values(3,3,36500.00);
insert into objectsurvey(addressid,organizationid,objectarea) values(4,4,1500.00);
insert into objectsurvey(addressid,organizationid,objectarea) values(5,4,1000.00);
insert into objectsurvey(addressid,organizationid,objectarea) values(5,5,27000.00);

insert into
application(clientid,objectsurveyid,brigadeid,incomingdate,statusapplicationid,starte
date,enddate) values(1,1,1,'2025-01-10',3,'2025-01-11','2025-01-15');
insert into
application(clientid,objectsurveyid,brigadeid,incomingdate,statusapplicationid,starte
date,enddate) values(2,2,2,'2025-01-15',3,'2025-01-16','2025-01-18');
insert into
application(clientid,objectsurveyid,brigadeid,incomingdate,statusapplicationid,starte
date,enddate) values(3,3,1,'2025-02-01',3,'2025-02-02','2025-02-05');
insert into
application(clientid,objectsurveyid,brigadeid,incomingdate,statusapplicationid,starte
date,enddate) values(4,4,2,'2025-02-10',3,'2025-02-11','2025-02-13');
insert into
application(clientid,objectsurveyid,brigadeid,incomingdate,statusapplicationid,starte
date,enddate) values(4,5,1,'2025-02-20',3,'2025-02-21','2025-02-25');
insert into
application(clientid,objectsurveyid,brigadeid,incomingdate,statusapplicationid,starte
date,enddate) values(5,6,2,'2025-03-01',2,'2025-04-07',null);
insert into
application(clientid,objectsurveyid,brigadeid,incomingdate,statusapplicationid,starte
date,enddate) values(1,2,1,'2025-03-10',2,'2025-04-09',null);
insert into
application(clientid,objectsurveyid,brigadeid,incomingdate,statusapplicationid,starte
date,enddate) values(2,3,2,'2025-03-20',1,null,null);

insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(1,1,1,10000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(2,1,15000.00,1500000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(1,2,1,10000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(3,2,5000.00,1000000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(1,3,1,10000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(4,3,15000.00,1500000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(1,4,1,10000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(5,4,1500.00,262500.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(1,5,1,10000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(2,5,1000.00,1000000.00);

```

```

insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(1,6,1,10000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(3,6,27000.00,5400000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(1,7,1,10000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(4,7,15000.00,15000000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(1,8,1,10000.00);
insert into SelectedServices(ServiceId,ApplicationId,volume,costservices)
values(5,8,20000.00,350000.00);

insert into SurveyReport(applicationid,employeeid,filereport)
values(1,2,'/reports/Жилойдом1.pdf');
insert into SurveyReport(applicationid,employeeid,filereport)
values(2,4,'/reports/Кожзавод1.pdf');
insert into SurveyReport(applicationid,employeeid,filereport)
values(3,3,'/reports/Барс1.pdf');
insert into SurveyReport(applicationid,employeeid,filereport)
values(4,5,'/reports/Прайм1.pdf');
insert into SurveyReport(applicationid,employeeid,filereport)
values(5,2,'/reports/Прайм2.pdf');

insert into
SurveyAgreement(applicationid,reportid,employeeid,createdate,confirmation,pricefororder) values(1,1,2,'2025-01-15',true,15010000.00);
insert into
SurveyAgreement(applicationid,reportid,employeeid,createdate,confirmation,pricefororder) values(2,2,4,'2025-01-18',true,1010000.00);
insert into
SurveyAgreement(applicationid,reportid,employeeid,createdate,confirmation,pricefororder) values(3,3,3,'2025-02-05',false,15010000.00);
insert into
SurveyAgreement(applicationid,reportid,employeeid,createdate,confirmation,pricefororder) values(4,4,5,'2025-02-13',true,272500.00);
insert into
SurveyAgreement(applicationid,reportid,employeeid,createdate,confirmation,pricefororder) values(5,5,2,'2025-02-25',true,1010000.00);

```

## 5.3 Скрипты хранимых функций и процедур

### 5.3.1 Скрипты селективных функций

```

--Функция для анализа заявок клиента
CREATE OR REPLACE FUNCTION analize_client_application(sPhone VARCHAR(18))
RETURNS TABLE (
    cityname VARCHAR(20),
    streetname VARCHAR(20),
    number VARCHAR(10),
    objectarea FLOAT,
    organizationname VARCHAR(30),
    inn VARCHAR(12),
    brigadename VARCHAR(40),
    incomingdate DATE,
    status VARCHAR(12),
    startdate DATE,
    enddate DATE,
    createdate DATE

```

```

)
AS $$

BEGIN
    IF sPhone IS NULL THEN
        RAISE EXCEPTION 'Ошибка: Телефонный номер заказчика не может быть null';
    END IF;

    IF NOT EXISTS (SELECT 1 FROM client WHERE phone = sPhone) THEN
        RAISE EXCEPTION 'Ошибка: Данный клиент не найден';
    END IF;

    IF NOT EXISTS (
        SELECT 1
        FROM application
        WHERE clientid = (SELECT clientid FROM client WHERE phone = sPhone)
    ) THEN
        RAISE EXCEPTION 'Ошибка: У данного клиента нет заявок';
    END IF;

    RETURN QUERY
    SELECT
        a2.cityname,
        a2.streetname,
        a2.number,
        o.objectarea,
        org.organizationname,
        org.inn,
        b.brigadename,
        a.incomingdate,
        s2.TypeStatus,
        a.startdate,
        a.enddate,
        s.createdate
    FROM application a
    JOIN objectsurvey o USING (objectsurveyid)
    JOIN address a2 USING (addressid)
    JOIN organization org USING (organizationid)
    JOIN brigade b USING (brigadeid)
    left JOIN surveyagreement s USING (ApplicationId)
    Join StatusApplication s2 USING(StatusApplicationId)
    WHERE a.clientid = (SELECT clientid FROM client WHERE phone = sPhone);
END;
$$ LANGUAGE plpgsql;

-- Функция для анализа прибыли организаций
CREATE OR REPLACE FUNCTION analize_organization(nstartdate date, nenddate date)
RETURNS TABLE (
    organizationname VARCHAR(30),
    inn VARCHAR(12),
    sumprice NUMERIC(20,2),
    countapplication BIGINT,
    percentapplication NUMERIC(5,2),
    countagreement BIGINT,
    percentagreement NUMERIC(5,2),
    firstapplicationdate DATE,
    lastapplicationdate DATE
)
AS $$

BEGIN

```

```

IF NOT EXISTS ( SELECT 1 FROM organization) THEN
    RAISE EXCEPTION 'Ошибка: Нет организаций';
END IF;

IF NOT EXISTS (SELECT 1 FROM application) THEN
    RAISE EXCEPTION 'Ошибка: Нет заявок';
END IF;

    if nstartdate is null and nenddate is null then
    RETURN QUERY
    SELECT
        o.organizationname,
        o.inn,
        SUM(CASE WHEN s.confirmation IS TRUE THEN s.pricefororder ELSE 0 END) AS
sumprice,
        COUNT(DISTINCT a.applicationid) AS countapplication,
        ROUND(100.0 * SUM(CASE WHEN a.StatusApplicationId = (select
StatusApplicationId from StatusApplication where TypeStatus = 'Закончена') THEN 1
ELSE 0 END) / NULLIF(COUNT(a.applicationid), 0), 2) AS percentapplication,
        COUNT(DISTINCT s.surveyagreementid) AS countagreement,
        ROUND(100.0 * SUM(CASE WHEN s.confirmation IS TRUE THEN 1 ELSE 0 END) /
NULLIF(COUNT(s.surveyagreementid), 0), 2) AS percentagreement,
        MIN(a.incomingdate) AS firstapplicationdate,
        MAX(a.incomingdate) AS lastapplicationdate
    FROM organization o JOIN objectsurvey os USING (organizationid) JOIN application
a USING (objectsurveyid) JOIN surveyagreement s using(applicationid)
    GROUP BY o.organizationname, o.inn;
    END IF;

    if nstartdate is null and nenddate is not null then
    RETURN QUERY
    SELECT
        o.organizationname,
        o.inn,
        SUM(CASE WHEN s.confirmation IS TRUE THEN s.pricefororder ELSE 0 END) AS
sumprice,
        COUNT(DISTINCT a.applicationid) AS countapplication,
        ROUND(100.0 * SUM(CASE WHEN a.StatusApplicationId = (select
StatusApplicationId from StatusApplication where TypeStatus = 'Закончена') THEN 1
ELSE 0 END) / NULLIF(COUNT(a.applicationid), 0), 2) AS percentapplication,
        COUNT(DISTINCT s.surveyagreementid) AS countagreement,
        ROUND(100.0 * SUM(CASE WHEN s.confirmation IS TRUE THEN 1 ELSE 0 END) /
NULLIF(COUNT(s.surveyagreementid), 0), 2) AS percentagreement,
        MIN(a.incomingdate) AS firstapplicationdate,
        MAX(a.incomingdate) AS lastapplicationdate
    FROM organization o JOIN objectsurvey os USING (organizationid) JOIN application
a USING (objectsurveyid) JOIN surveyagreement s using(applicationid)
    where s.createdate <= nenddate
    GROUP BY o.organizationname, o.inn;
    END IF;

    if nstartdate is not null and nenddate is null then
    RETURN QUERY
    SELECT
        o.organizationname,
        o.inn,
        SUM(CASE WHEN s.confirmation IS TRUE THEN s.pricefororder ELSE 0 END) AS
sumprice,
        COUNT(DISTINCT a.applicationid) AS countapplication,

```

```

        ROUND(100.0 * SUM(CASE WHEN a.StatusApplicationId = (select
StatusApplicationId from StatusApplication where TypeStatus = 'Закончена') THEN 1
ELSE 0 END) / NULLIF(COUNT(a.applicationid), 0), 2) AS percentapplication,
        COUNT(DISTINCT s.surveyagreementid) AS countagreement,
        ROUND(100.0 * SUM(CASE WHEN s.confirmation IS TRUE THEN 1 ELSE 0 END) /
NULLIF(COUNT(s.surveyagreementid), 0), 2) AS percentagreement,
        MIN(a.incomingdate) AS firstapplicationdate,
        MAX(a.incomingdate) AS lastapplicationdate
    FROM organization o JOIN objectsurvey os USING (organizationid) JOIN application
a USING (objectsurveyid) JOIN surveyagreement s using(applicationid)
        where s.createdate >= nstartdate
    GROUP BY o.organizationname, o.inn;
    END IF;

    if nstartdate is not null and nenddate is not null then
RETURN QUERY
SELECT
    o.organizationname,
    o.inn,
    SUM(CASE WHEN s.confirmation IS TRUE THEN s.pricefororder ELSE 0 END) AS
sumprice,
    COUNT(DISTINCT a.applicationid) AS countapplication,
    ROUND(100.0 * SUM(CASE WHEN a.StatusApplicationId = (select
StatusApplicationId from StatusApplication where TypeStatus = 'Закончена') THEN 1
ELSE 0 END) / NULLIF(COUNT(a.applicationid), 0), 2) AS percentapplication,
    COUNT(DISTINCT s.surveyagreementid) AS countagreement,
    ROUND(100.0 * SUM(CASE WHEN s.confirmation IS TRUE THEN 1 ELSE 0 END) /
NULLIF(COUNT(s.surveyagreementid), 0), 2) AS percentagreement,
    MIN(a.incomingdate) AS firstapplicationdate,
    MAX(a.incomingdate) AS lastapplicationdate
    FROM organization o JOIN objectsurvey os USING (organizationid) JOIN application
a USING (objectsurveyid) JOIN surveyagreement s using(applicationid)
        where s.createdate between nstartdate and nenddate
    GROUP BY o.organizationname, o.inn;
    END IF;
END;
$$ LANGUAGE plpgsql;

```

-- Функция для анализа прибыли исполнительных бригад

```
CREATE OR REPLACE FUNCTION analize_brigade(nstartdate DATE, nenddate DATE)
RETURNS TABLE (

```

```

brigadename VARCHAR(50),
sumprice NUMERIC(20,2),
countapplication BIGINT,
percentapplication NUMERIC(5,2),
countagreement BIGINT,
percentagreement NUMERIC(5,2),
firstapplicationdate DATE,
lastapplicationdate DATE
)
```

```
AS $$
```

```
BEGIN
```

```
IF NOT EXISTS (SELECT 1 FROM brigade) THEN
```

```
    RAISE EXCEPTION 'Ошибка: Нет бригад';
```

```
END IF;
```

```
IF NOT EXISTS (SELECT 1 FROM application) THEN
```

```
    RAISE EXCEPTION 'Ошибка: Нет заявок';
```

```

END IF;

IF nstartdate IS NULL AND nenddate IS NULL THEN
    RETURN QUERY
    SELECT
        b.brigadename,
        SUM(CASE WHEN s.confirmation IS TRUE THEN s.pricefororder ELSE 0 END),
        COUNT(DISTINCT a.applicationid),
        ROUND(100.0 * SUM(CASE WHEN a.StatusApplicationId = (select
StatusApplicationId from StatusApplication where TypeStatus = 'Закончена') THEN 1
ELSE 0 END) / NULLIF(COUNT(a.applicationid), 0), 2),
        COUNT(DISTINCT s.surveyagreementid),
        ROUND(100.0 * SUM(CASE WHEN s.confirmation IS TRUE THEN 1 ELSE 0 END) /
NULLIF(COUNT(s.surveyagreementid), 0), 2),
        MIN(a.incomingdate),
        MAX(a.incomingdate)
    FROM brigade b
    JOIN application a USING (brigadeid)
    JOIN surveyagreement s using(applicationid)
    GROUP BY b.brigadename;
END IF;

IF nstartdate IS NULL AND nenddate IS NOT NULL THEN
    RETURN QUERY
    SELECT
        b.brigadename,
        SUM(CASE WHEN s.confirmation IS TRUE THEN s.pricefororder ELSE 0 END),
        COUNT(DISTINCT a.applicationid),
        ROUND(100.0 * SUM(CASE WHEN a.StatusApplicationId = (select
StatusApplicationId from StatusApplication where TypeStatus = 'Закончена') THEN 1
ELSE 0 END) / NULLIF(COUNT(a.applicationid), 0), 2),
        COUNT(DISTINCT s.surveyagreementid),
        ROUND(100.0 * SUM(CASE WHEN s.confirmation IS TRUE THEN 1 ELSE 0 END) /
NULLIF(COUNT(s.surveyagreementid), 0), 2),
        MIN(a.incomingdate),
        MAX(a.incomingdate)
    FROM brigade b
    JOIN application a USING (brigadeid)
    JOIN surveyagreement s using(applicationid)
    WHERE s.createdate <= nenddate
    GROUP BY b.brigadename;
END IF;

IF nstartdate IS NOT NULL AND nenddate IS NULL THEN
    RETURN QUERY
    SELECT
        b.brigadename,
        SUM(CASE WHEN s.confirmation IS TRUE THEN s.pricefororder ELSE 0 END),
        COUNT(DISTINCT a.applicationid),
        ROUND(100.0 * SUM(CASE WHEN a.StatusApplicationId = (select
StatusApplicationId from StatusApplication where TypeStatus = 'Закончена') THEN 1
ELSE 0 END) / NULLIF(COUNT(a.applicationid), 0), 2),
        COUNT(DISTINCT s.surveyagreementid),
        ROUND(100.0 * SUM(CASE WHEN s.confirmation IS TRUE THEN 1 ELSE 0 END) /
NULLIF(COUNT(s.surveyagreementid), 0), 2),
        MIN(a.incomingdate),
        MAX(a.incomingdate)
    FROM brigade b
    JOIN application a USING (brigadeid)

```

```

        JOIN surveyagreement s using(applicationid)
        WHERE s.createdate >= nstartdate
        GROUP BY b.brigadename;
    END IF;

    IF nstartdate IS NOT NULL AND nenddate IS NOT NULL THEN
        RETURN QUERY
        SELECT
            b.brigadename,
            SUM(CASE WHEN s.confirmation IS TRUE THEN s.pricefororder ELSE 0 END),
            COUNT(DISTINCT a.applicationid),
            ROUND(100.0 * SUM(CASE WHEN a.StatusApplicationId = (select
StatusApplicationId from StatusApplication where TypeStatus = 'Закончена') THEN 1
ELSE 0 END) / NULLIF(COUNT(a.applicationid), 0), 2),
            COUNT(DISTINCT s.surveyagreementid),
            ROUND(100.0 * SUM(CASE WHEN s.confirmation IS TRUE THEN 1 ELSE 0 END) /
NULLIF(COUNT(s.surveyagreementid), 0), 2),
            MIN(a.incomingdate),
            MAX(a.incomingdate)
        FROM brigade b
        JOIN application a USING (brigadeid)
        JOIN surveyagreement s using(applicationid)
        WHERE s.createdate BETWEEN nstartdate AND nenddate
        GROUP BY b.brigadename;
    END IF;
END;
$$ LANGUAGE plpgsql;

```

### 5.3.2 Скрипты выполняемых процедур

```

-- Процедура для добавления заказчика
create or replace procedure new_client(nFio VARCHAR (40), nPhone VARCHAR (18), nMail
varchar(100), nPassword varchar(100))
AS $$

declare
reg_clientid integer;
begin
    if nFio is null then
        RAISE EXCEPTION 'Ошибка: ФИО не может быть Null';
    END IF;

    if nPhone is null then
        RAISE EXCEPTION 'Ошибка: Номер телефона не может быть Null';
    END IF;

    if exists (select 1 from client where phone = nPhone) then
        RAISE EXCEPTION 'Ошибка: Заказчик с таким номером телефона уже существует';
    END IF;

    if nMail is null then
        RAISE EXCEPTION 'Ошибка: Почта не может быть Null';
    END IF;

    if exists (select 1 from Account where Mail = nMail) then
        RAISE EXCEPTION 'Ошибка: Учетная запись с такой почтой уже существует';
    END IF;

    if nPassword is null then

```

```

        RAISE EXCEPTION 'Ошибка: Пароль не может быть NULL';
    END IF;

        insert into client(Fio,Phone) values(nFio,nPhone)
        returning clientid into reg_clientid;
        insert into Account (Mail,Password,AccessLevelId,ClientId) values
    (nMail,nPassword,(select AccessLevelId from AccessLevel where AccessLevelName =
    'Client'),reg_clientid);
    EXCEPTION
        WHEN OTHERS THEN
            RAISE EXCEPTION 'Ошибка при добавлении заказчика: %', SQLERRM;
    END;
$$ LANGUAGE plpgsql;

-- Процедура для добавления заявки
create or replace procedure new_application(nPhone VARCHAR (18), nOrgName VARCHAR
(30), nINN VARCHAR (12), nCity VARCHAR (20), nStreet VARCHAR (20), nNumber VARCHAR
(10), nArea Float)
as $$

declare
nOrgId int;
naddressId int;
nObjectId int;
begin
    IF nPhone IS NULL THEN
        RAISE EXCEPTION 'Ошибка: Телефон не может быть NULL';
    END IF;

    IF NOT exists (select 1 from client where phone = nPhone) THEN
        RAISE EXCEPTION 'Ошибка: Клиента с указанным номером телефона нет';
    END IF;

    IF nOrgName IS NULL THEN
        RAISE EXCEPTION 'Ошибка: Наименование организации не может быть NULL';
    END IF;

    IF nINN IS NULL THEN
        RAISE EXCEPTION 'Ошибка: ИНН организации не может быть NULL';
    END IF;

    IF ncity IS NULL THEN
        RAISE EXCEPTION 'Ошибка: Город не может быть NULL';
    END IF;

    IF nstreet IS NULL THEN
        RAISE EXCEPTION 'Ошибка: Улица не может быть NULL';
    END IF;

    IF nNumber IS NULL THEN
        RAISE EXCEPTION 'Ошибка: Номер строения не может быть NULL';
    END IF;

    IF nArea IS NULL THEN
        RAISE EXCEPTION 'Ошибка: Площадь объекта не может быть NULL';
    END IF;

    IF nArea <= 0 THEN
        RAISE EXCEPTION 'Ошибка: Площадь объекта не может быть отрицательной или
равняться нулю';
    END IF;

```

```

END IF;

if      EXISTS (select 1 from Organization where INN=nINN) then
select OrganizationId into nOrgId
from Organization where
inn=nINN;
else
insert into organization(OrganizationName,INN) values(nOrgName,nINN)
returning organizationid into nOrgId;
end if;
IF EXISTS (SELECT 1 FROM address WHERE cityname = nCity AND streetname =
nStreet AND number = nNumber) THEN
    SELECT addressid INTO naddressId
    FROM address
    WHERE cityname = nCity AND streetname = nStreet AND number = nNumber;
    ELSE
        INSERT INTO address (cityname, streetname, number) VALUES (nCity, nStreet,
nNumber)
        RETURNING addressid INTO naddressId;
    END IF;
    if EXISTS (select 1 from ObjectSurvey where AddressId = naddressId and
OrganizationId = nOrgId and ObjectArea = nArea) then
        select ObjectSurveyId into nObjectId
        from ObjectSurvey
        where AddressId = naddressId and OrganizationId = nOrgId and ObjectArea =
nArea;
        else
            insert into ObjectSurvey(AddressId,OrganizationId,ObjectArea)
values(naddressId,nOrgId,nArea)
            RETURNING ObjectSurveyId INTO nObjectId;
        end if;
        insert into
application(ClientId,ObjectSurveyId,StatusApplicationId,StarteDate,EndDate) values
((select ClientId from client where phone=nPhone),nObjectId,(select
StatusApplicationId from StatusApplication where TypeStatus = 'Принята'),null,null);
    EXCEPTION
        WHEN OTHERS THEN
            RAISE EXCEPTION 'Ошибка при добавлении заявки: %', SQLERRM;
end;
$$ LANGUAGE plpgsql;

-- Процедура для добавления выбранной услуги
create or replace procedure new_selectedservices(nPhone VARCHAR (18),nIncomingDate
date, nINN VARCHAR (12), nCity VARCHAR (20), nStreet VARCHAR (20), nNumber VARCHAR
(10),nServiceName VARCHAR (30), nVolume Float)
as $$
declare
nApplicationId int;
begin

IF nPhone IS NULL THEN
RAISE EXCEPTION 'Ошибка: Телефон не может быть NULL';
END IF;

IF NOT exists (select 1 from client where phone = nPhone) THEN
RAISE EXCEPTION 'Ошибка: Клиента с указанным номером телефона нет';
END IF;

IF nIncomingDate IS NULL THEN

```

```

RAISE EXCEPTION 'Ошибка: Дата не может быть NULL';
END IF;

IF nINN IS NULL THEN
RAISE EXCEPTION 'Ошибка: ИНН организации не может быть NULL';
END IF;

IF not exists(select 1 from organization where INN = nINN) THEN
RAISE EXCEPTION 'Ошибка: Нет организации с указанным ИНН';
END IF;

IF ncity IS NULL THEN
RAISE EXCEPTION 'Ошибка: Город не может быть NULL';
END IF;

IF nstreet IS NULL THEN
RAISE EXCEPTION 'Ошибка: Улица не может быть NULL';
END IF;

IF nNumber IS NULL THEN
RAISE EXCEPTION 'Ошибка: Номер строения не может быть NULL';
END IF;

if not exists (select 1 from address where cityname = nCity AND streetname =
nStreet AND number = nNumber) then
RAISE EXCEPTION 'Ошибка: Указанного адреса нет';
END IF;

IF nServiceName IS NULL THEN
RAISE EXCEPTION 'Ошибка: Наименование услуги не может быть NULL';
END IF;

IF not exists(select 1 from ServiceCatalog where ServiceName = nServiceName)
THEN
RAISE EXCEPTION 'Ошибка: Указанной услуги нет в каталоге услуг';
END IF;

IF nVolume IS NULL THEN
RAISE EXCEPTION 'Ошибка: Объем выполняемой работы не может быть NULL';
END IF;

IF nVolume <= 0 THEN
RAISE EXCEPTION 'Ошибка: Объем выполняемой работы не может быть отрицательной
или равняться нулю';
END IF;

select ApplicationId into nApplicationId
from Application
where ClientId = (select ClientId from Client where phone = nPhone) and
ObjectSurveyId = (select ObjectSurveyId from ObjectSurvey where OrganizationId =
(select OrganizationId from Organization where inn = nInn) and AddressId = (select
AddressId from address where cityname = nCity AND streetname = nStreet AND number =
nNumber))
and IncomingDate = nIncomingDate
limit 1;

if nApplicationId is null then
RAISE EXCEPTION 'Ошибка: Заявки на эту дату нет';
END IF;

```

```

        insert into SelectedServices(ServiceId,ApplicationId,Volume) values ((select
ServiceId from ServiceCatalog where
ServiceName=nServiceName),nApplicationId,nVolume);

        EXCEPTION
        WHEN OTHERS THEN
            RAISE EXCEPTION 'Ошибка при добавлении заявки: %', SQLERRM;
end;
$$ LANGUAGE plpgsql;

```

## 5.4 Скрипты триггеров

### 5.4.1 Скрипты триггеров бизнес-правил

```

--триггер для подсчета стоимости выбранной услуги
CREATE OR REPLACE FUNCTION calculate_costservices()
RETURNS TRIGGER AS $$

declare
nPrice Numeric(10,2);
BEGIN
    SELECT price INTO nPrice
    FROM servicecatalog
    WHERE ServiceId = NEW.ServiceId;

    NEW.costservices := NEW.volume * nPrice;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_calculate_costservices
BEFORE INSERT OR UPDATE ON SelectedServices
FOR EACH ROW
EXECUTE FUNCTION calculate_costservices();

--триггер для подсчета стоимости договора
CREATE OR REPLACE FUNCTION calculate_pricefororder()
RETURNS TRIGGER AS $$

BEGIN

    SELECT SUM(costservices) INTO NEW.pricefororder
    FROM SelectedServices
    WHERE ApplicationId = NEW.applicationid;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_calculate_pricefororder
BEFORE INSERT ON SurveyAgreement
FOR EACH ROW
EXECUTE FUNCTION calculate_pricefororder();

--триггер для назначения бригады на заявку
CREATE OR REPLACE FUNCTION assign_brigade_to_application()
RETURNS TRIGGER AS $$

DECLARE
    selected_brigade_id INT;
BEGIN

```

```

        SELECT b.brigadeid INTO selected_brigade_id
        FROM brigade b
        LEFT JOIN application a using (brigadeid)
        GROUP BY b.brigadeid
        ORDER BY COUNT(a.applicationid)
        LIMIT 1;
    NEW.brigadeid := selected_brigade_id;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_assign_brigade
BEFORE INSERT ON Application
FOR EACH ROW
EXECUTE FUNCTION assign_brigade_to_application();

```

#### 5.4.2 Скрипты триггеров бизнес-ограничений

```

create or replace function check_application_status_before_report()
returns trigger as $$

declare
    app_status integer;
begin
    select StatusApplicationId into app_status
    from Application
    where ApplicationId = NEW.ApplicationId;

    if app_status != 3 then
        raise exception 'Невозможно добавить отчет: заявка ещё не выполнена.';
    end if;

    return NEW;
end;
$$ language plpgsql;

create trigger trg_check_status_before_report
before insert on SurveyReport
for each row
execute function check_application_status_before_report();

-- триггер на ограничение добавления договора
create or replace function check_before_agreement()
returns trigger as $$

declare
    app_status integer;
    report_exists boolean;
begin
    select StatusApplicationId into app_status
    from Application
    where ApplicationId = NEW.ApplicationId;

    select exists (
        select 1 from SurveyReport
        where ApplicationId = NEW.ApplicationId
    ) into report_exists;

    if app_status != 3 or not report_exists then

```

```

        raise exception 'Невозможно сформировать договор: заявка не выполнена или
отсутствует отчет.';
    end if;

    return NEW;
end;
$$ language plpgsql;

create trigger trg_check_before_agreement
before insert on SurveyAgreement
for each row
execute function check_before_agreement();

-- триггер на ограничение даты создания договора
create or replace function check_agreement_date()
returns trigger as $$

declare
    app_end_date date;
begin
    select EndDate into app_end_date
    from Application
    where ApplicationId = NEW.ApplicationId;

    if NEW.CreateDate < app_end_date then
        raise exception 'Дата договора не может быть раньше окончания обследования.' ;
    end if;

    return NEW;
end;
$$ language plpgsql;

create trigger trg_check_agreement_date
before insert on SurveyAgreement
for each row
execute function check_agreement_date();

```

### 5.4.3 Скрипт триггера ограничения целостности

```

create or replace function prevent_report_delete_if_confirmed()
returns trigger as $$

declare
    is_confirmed boolean;
begin
    select Confirmation into is_confirmed
    from SurveyAgreement
    where ReportId = OLD.ReportId;

    if is_confirmed is true then
        raise exception 'Отчет нельзя удалить: связан с подтвержденным договором.' ;
    end if;

    return OLD;
end;
$$ language plpgsql;

create trigger trg_prevent_report_delete
before delete on SurveyReport
for each row

```

```
execute function prevent_report_delete_if_confirmed();
```

#### 5.4.4 Скрипт триггеров ограничения действия

```
--триггер на ограничение добавления отчета
create or replace function check_employee_brigade_for_report()
returns trigger as $$

declare
    app_brigade_id integer;
    emp_brigade_id integer;
begin
    select BrigadeId into app_brigade_id
    from Application
    where ApplicationId = NEW.ApplicationId;

    select BrigadeId into emp_brigade_id
    from Employee
    where EmployeeId = NEW.EmployeeId;

    if app_brigade_id is distinct from emp_brigade_id then
        raise exception 'Сотрудник не состоит в бригаде, выполнившей заявку.';
    end if;

    return NEW;
end;
$$ language plpgsql;

create trigger trg_check_employee_brigade
before insert on SurveyReport
for each row
execute function check_employee_brigade_for_report();

-- триггер на ограничение удаления договора
create or replace function prevent_agreement_delete_if_confirmed()
returns trigger as $$

begin
    if OLD.Confirmation then
        raise exception 'Нельзя удалить подтвержденный договор.';
    end if;
    return OLD;
end;
$$ language plpgsql;

create trigger trg_prevent_agreement_delete
before delete on SurveyAgreement
for each row
execute function prevent_agreement_delete_if_confirmed();
```

#### 5.4.5 Скрипт триггера DDL

```
--триггер на ограничение удаления таблиц
CREATE OR REPLACE FUNCTION prevent_drop_table()
RETURNS event_trigger AS
$$
BEGIN
    IF current_user <> 'postgres' THEN
        RAISE EXCEPTION 'Удаление таблиц разрешено только пользователю postgres!';
    END IF;
```

```

END;
$$
LANGUAGE plpgsql;
CREATE EVENT TRIGGER drop_table_restrict
ON ddl_command_start
WHEN TAG IN ('DROP TABLE')
EXECUTE FUNCTION prevent_drop_table();

```

### 5.4.6 Скрипт триггера БД

```

--таблица для логирования подключения к бд
create table userlog (
id serial primary key,
usern text,
datetime TIMESTAMP WITH TIME zone default CURRENT_TIMESTAMP
);

--триггер для логирования подключения к бд
CREATE OR REPLACE FUNCTION connectuser()
RETURNS EVENT_TRIGGER AS $$%
DECLARE
username TEXT;
datetimes TIMESTAMP;
BEGIN
SELECT session_user INTO username;
SELECT CURRENT_TIMESTAMP INTO datetimes;
INSERT INTO userlog (usern, datetime)VALUES (username, datetimes);
END;
$$ LANGUAGE plpgsql;

CREATE EVENT TRIGGER log_user
ON LOGIN
EXECUTE FUNCTION connectuser();

```

### 5.5 Скрипты реализации политики безопасности

```

-- Роли соответствуют AccessLevelName
CREATE ROLE "Admin" LOGIN PASSWORD 'admin_pass';
CREATE ROLE "Employee" LOGIN PASSWORD 'employee_pass';
CREATE ROLE "Client" LOGIN PASSWORD 'client_pass';

-- Права администратора
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO "Admin";
GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO "Admin";
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO "Admin";
GRANT EXECUTE ON ALL PROCEDURES IN SCHEMA public TO "Admin";

ALTER ROLE "Admin" WITH CREATEROLE;

-- Права сотрудника
GRANT SELECT, INSERT, UPDATE ON Application TO "Employee";
GRANT SELECT, INSERT, UPDATE ON SurveyReport TO "Employee";
GRANT SELECT, INSERT, UPDATE ON SurveyAgreement TO "Employee";

GRANT SELECT ON
    Client,
    SelectedServices,
    ServiceCatalog,
    Organization,

```

```
Address,
ObjectSurvey,
Brigade
TO "Employee";

GRANT EXECUTE ON FUNCTION analize_brigade(date, date) TO "Employee";
GRANT EXECUTE ON FUNCTION analize_organization(date, date) TO "Employee";

-- Права клиента
GRANT SELECT, INSERT, UPDATE ON Organization TO "Client";
GRANT SELECT, INSERT, UPDATE ON Address TO "Client";

GRANT INSERT ON Application TO "Client";
GRANT INSERT ON SelectedServices TO "Client";

GRANT SELECT ON
ServiceCatalog,
Application,
SurveyAgreement,
SurveyReport
TO "Client";

GRANT EXECUTE ON FUNCTION analize_client_application(varchar) TO "Client";
GRANT EXECUTE ON PROCEDURE new_client(varchar, varchar, varchar, varchar) TO
"Client";
GRANT EXECUTE ON PROCEDURE new_application(varchar, varchar, varchar, varchar,
varchar, varchar, float) TO "Client";
GRANT EXECUTE ON PROCEDURE SelectedServices(varchar, date, varchar, varchar, varchar,
varchar, varchar, float) TO "Client";
```

## 6 Экспериментальная часть

### 6.1 Проверка базы данных и наличия таблиц

В результате выполнения скрипта из п. 5 была создана база данных, схемой которой представлена на рисунке 9.

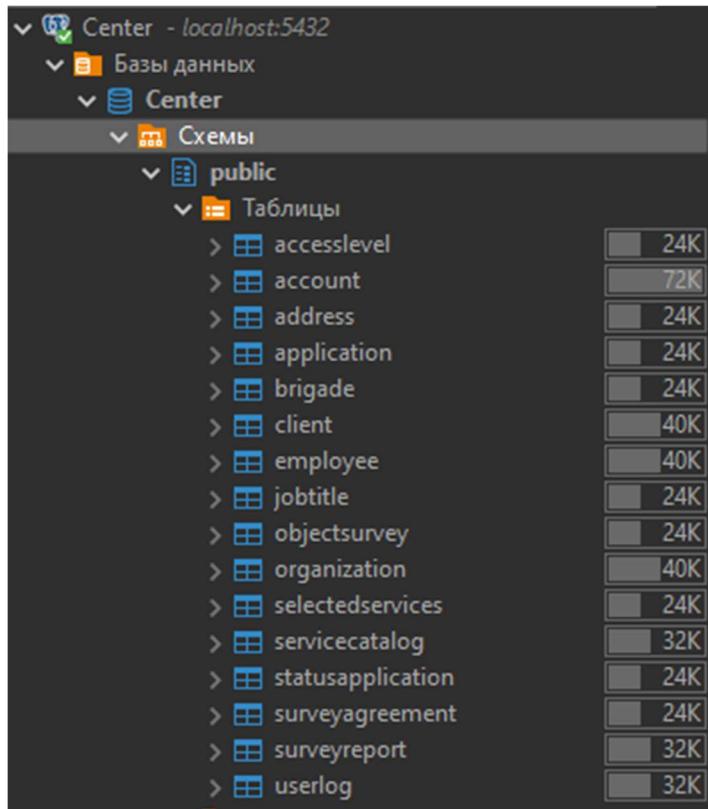


Рисунок 9 – Схема базы данных

### 6.2 Проверка таблиц

В базе данных 15 таблиц. Созданные и заполненные данными таблицы представлены на рисунках 10 – 24.

```
select * from client c
```

	clientid	fio	phone
1	1	Кузьмин Сергей Иванович	8(905)584-98-94
2	2	Евсеев Адам Романович	8(910)952-05-13
3	3	Пахомова Алина Михайловна	8(950)005-31-93
4	4	Мальцев Константин Артёмович	8(942)507-57-05
5	5	Романова Валентина Александровна	8(952)174-27-75

Рисунок 10 – Таблица «Client»

```
select * from employee e
```

	employeeid	fio	phone	jobtitleid	brigadeid
1	1	Ермолов Леон Маркович	8(910)293-00-97	1	[NULL]
2	2	Симонов Николай Егорович	8(965)867-22-69	2	1
3	3	Рябов Никита Макарович	8(971)301-14-78	2	2
4	4	Григорьев Илья Кириллович	8(944)991-09-27	3	1
5	5	Лобанов Виктор Евгеньевич	8(956)295-85-89	3	2
6	6	Уткина Лея Дмитриевна	8(985)874-96-44	4	[NULL]

Рисунок 11 – Таблица «Employee»

```
select * from jobtitle j
```

	jobtitleid	jobtitlename
1	1	Директор
2	2	Главный инженер проектировщик
3	3	Инженер проектировщик
4	4	Менеджер

Рисунок 12 – Таблица «JobTitle»

```
select * from accesslevel a
```

	accesslevelid	accesslevelname
1	1	Client
2	2	Admin
3	3	Employee

Рисунок 13 – Таблица «AccessLevel»

```
select * from account a
```

	accountid	mail	password	accesslevelid	clientid	employeeid
1	1	Kyzmin1985@mail.ru	KyzminKyzmin	1	1	[NULL]
2	2	Evseev1990@yandex.ru	Evseev123	1	2	[NULL]
3	3	Alina41001@gmail.ru	Paxomova3012	1	3	[NULL]
4	4	Maltsev1988@mail.ru	Maltsev1988	1	4	[NULL]
5	5	Romanova62@yandex.ru	RomanovaRomanova	1	5	[NULL]
6	6	Centrlsklm@mail.ru	OemDHdZYztVH6pDw7u	2	[NULL]	1
7	7	NikolaCim@yandex.ru	NikolaCim1989	3	[NULL]	2
8	8	RabovNikita@yandex.ru	Nikitata	3	[NULL]	3
9	9	Grigrorev@gmail.ru	Grigrorev991	3	[NULL]	4
10	10	Lobanov2000@gmail.ru	Lobanov1100	3	[NULL]	5
11	11	YtkinaLei@mail.ru	YtkinaLeiYtkinaLei	3	[NULL]	6

Рисунок 14 – Таблица «Account»

```
select * from address a
```

	123 addressid	A-Z cityname	A-Z streetname	A-Z number
1	1	г.Рязань	ул. Интернациональная	д.1
2	2	г.Рязань	ул. Прижелезнодорожная	д.52
3	3	г.Рязань	Московское шоссе	д.5А
4	4	г.Рязань	ул. Бирюзова	д.28А
5	5	г.Рязань	ул. Интернациональная	д.23
6	6	г.Рязань	шоссе Солотчинское	д.11

Рисунок 15 – Таблица «Address»

```
select * from organization o
```

	123 organizationid	A-Z organizationname	A-Z inn
1	1	Строительная компания	0420520521
2	2	Русская кожа	0602065848
3	3	тц. Барс	6227009810
4	4	Прайм	6234203335
5	5	Круиз	6234085000

Рисунок 16 – Таблица «Organization»

```
select * from objectsurvey o
```

	123 objectsurveyid	123 addressid	123 organizationid	123 objectarea
1	1	1	1	15 000
2	2	2	2	160 000
3	3	3	3	36 500
4	4	4	4	1 500
5	5	5	4	1 000
6	6	5	5	27 000

Рисунок 17 – Таблица «ObjectSurvey»

```
select * from brigade b
```

	123 brigadeid	A-Z brigadename
1	1	Бригада №1
2	2	Бригада №2

Рисунок 18 – Таблица «Brigade»

```
select * from statusapplication s
```

	123 statusapplicationid	A-Z typestatus
1	1	Принята
2	2	Выполняется
3	3	Закончена

Рисунок 19 – Таблица «StatusApplication»

```
select * from application a
```

	123 applicationid	123 clientid	123 objectsurveyid	123 brigadeid	incomingdate	123 statusapplicationid	startedate	enddate
1	1	1	1	1	2025-01-10	3	2025-01-11	2025-01-15
2	2	2	2	2	2025-01-15	3	2025-01-16	2025-01-18
3	3	3	3	1	2025-02-01	3	2025-02-02	2025-02-05
4	4	4	4	2	2025-02-10	3	2025-02-11	2025-02-13
5	5	4	5	1	2025-02-20	3	2025-02-21	2025-02-25
6	6	5	6	2	2025-03-01	2	2025-04-07	[NULL]
7	7	1	2	1	2025-03-10	2	2025-04-09	[NULL]
8	8	2	3	2	2025-03-20	1	[NULL]	[NULL]

Рисунок 20 – Таблица «Application»

```
select * from servicecatalog s
```

	123 serviceid	A-Z servicename	123 price	A-Z measurement	A-Z description
1	1	Составление отчета	10 000	ШТ	Наши специалисты составят отчет, в котором будут указаны проблемные места конструкции
2	2	Обследование технического состояния несущих и ограждающих конструкций	350	M2	Проводится детальное обследование несущих и ограждающих конструкций здания на пред
3	3	Поиск и обследование технического состояния инженерных коммуникаций	400	M2	Выполняется поиск и диагностика состояния внутренних коммуникаций: водоснабжения, ки
4	4	Обследование технического состояния инженерных систем	450	M2	Оценивается общее техническое состояние инженерных систем здания с предоставлением
5	5	Обмерные работы	200	M2	Проводятся точные обмеры помещения или здания с составлением чертежей и схем для п

Рисунок 21 – Таблица «ServiceCatalog»

```
select * from selectedservices s
```

	123 selectedservicesid	123 serviceid	123 applicationid	123 volume	123 costservices
1	1	1	1	1	10 000
2		2	1	15 000	15 000 000
3		1	2	1	10 000
4		3	2	5 000	1 000 000
5		1	3	1	10 000
6		4	3	15 000	15 000 000
7		1	4	1	10 000
8		5	4	1 500	262 500
9		1	5	1	10 000
10		2	5	1 000	1 000 000
11		1	6	1	10 000
12		12	6	27 000	5 400 000
13		13	7	1	10 000
14		14	7	15 000	15 000 000
15		15	8	1	10 000
16		5	8	20 000	350 000

Рисунок 22 – Таблица «SelectedServices»

```
select * from surveyreport s
```

	123 reportid	123 applicationid	123 employeeid	A-Z filereport
1	1	1	2	/reports/Жилойдом1.pdf
2		2	4	/reports/Кожзавод1.pdf
3		3	3	/reports/Барс1.pdf
4		4	5	/reports/Прайм1.pdf
5		5	2	/reports/Прайм2.pdf

Рисунок 23 – Таблица «SurveyReport»

```
select * from surveyagreement s
```

	123 surveyagreementid	123 applicationid	123 reportid	123 employeedid	createdate	confirmation	123 pricefororder
1	1	1	1	2	2025-01-15	[v]	15 010 000
2	2	2	2	3	2025-01-18	[v]	1 010 000
3	3	3	3	4	2025-02-05	[ ]	15 010 000
4	4	4	4	5	2025-02-13	[v]	272 500
5	5	5	5	2	2025-02-25	[v]	1 010 000

Рисунок 24 – Таблица «SurveyAgreement»

## 6.3 Проверка работы хранимых функций и процедур

### 6.3.1 Проверка работы селективных функций

1. Проверка функции «analyze\_client\_application». Выполним функцию, выведем информацию для клиента с номером телефона 8(905)584-98-94 (рисунок 25 - 27).

```
select * from analyze_client_application('8(905)584-98-94')
```

	A-Z cityname	A-Z streetname	A-Z number	123 objectarea	A-Z organizationname	A-Z inn	A-Z brigadename	123 incomingdate	A-Z status	123 startdate	123 enddate	123 createdate
1	г.Рязань	ул. Прижелезнодорожная	д.52	160 000	Русская кожа	0602065848	Бригада №1	2025-03-10	Выполняется	2025-04-09	[NULL]	[NULL]
2	г.Рязань	ул. Интернациональная	д.1	15 000	Строительная компания	0420520521	Бригада №1	2025-01-10	Закончена	2025-01-11	2025-01-15	2025-01-15

Рисунок 25 – Результат выполнения функции

```
select * from analyze_client_application(NULL)
```

```
SQL Error [P0001]: ОШИБКА: Ошибка: Телефонный номер заказчика не может быть null
Где: функция PL/pgSQL analyze_client_application(character varying), строка 4, оператор RAISE
```

Рисунок 26 – Результат выполнения функции

```
select * from analyze_client_application('')
```

```
SQL Error [P0001]: ОШИБКА: Ошибка: Данный клиент не найден
Где: функция PL/pgSQL analyze_client_application(character varying), строка 8, оператор RAISE
```

Рисунок 27 – Результат выполнения функции

2. Проверка функции «analyze\_organization». Выполним функцию несколько раз с входными значениями от 15.01.2025 до 05.02.2025, с 15.01.2025, до 05.02.2025 и null (рисунок 28 - 31).

```
select * from analyze_organization('2025-01-15','2025-02-05')
```

A-Z organizationname	A-Z inn	123 sumprice	123 countapplication	123 percentapplication	123 countagreement	123 percentagreement	123 firstapplicationdate	123 lastapplicationdate
Русская кожа	0602065848	1 010 000	1	100	1	100	2025-01-15	2025-01-15
Строительная компания	0420520521	15 010 000	1	100	1	100	2025-01-10	2025-01-10
тц. Барс	6227009810	0	1	100	1	0	2025-02-01	2025-02-01

Рисунок 28 – Результат выполнения функции

```
select * from analyze_organization('2025-01-15',null)
```

A-Z organizationname	A-Z inn	I23 sumprice	I23 countapplication	I23 percentapplication	I23 countagreement	I23 percentagreement	firstapplicationdate	lastapplicationdate
Прайм	6234203335	1 282 500	2	100	2	100	2025-02-10	2025-02-20
Русская кожа	0602065848	1 010 000	1	100	1	100	2025-01-15	2025-01-15
Строительная компания	0420520521	15 010 000	1	100	1	100	2025-01-10	2025-01-10
тц. Барс	6227009810	0	1	100	1	0	2025-02-01	2025-02-01

Рисунок 29 – Результат выполнения функции

```
select * from analize_organization(null,'2025-02-05')
```

A-Z organizationname	A-Z inn	I23 sumprice	I23 countapplication	I23 percentapplication	I23 countagreement	I23 percentagreement	firstapplicationdate	lastapplicationdate
Русская кожа	0602065848	1 010 000	1	100	1	100	2025-01-15	2025-01-15
Строительная компания	0420520521	15 010 000	1	100	1	100	2025-01-10	2025-01-10
тц. Барс	6227009810	0	1	100	1	0	2025-02-01	2025-02-01

Рисунок 30 – Результат выполнения функции

```
select * from analize_organization(null,null)
```

A-Z organizationname	A-Z inn	I23 sumprice	I23 countapplication	I23 percentapplication	I23 countagreement	I23 percentagreement	firstapplicationdate	lastapplicationdate
Прайм	6234203335	1 282 500	2	100	2	100	2025-02-10	2025-02-20
Русская кожа	0602065848	1 010 000	1	100	1	100	2025-01-15	2025-01-15
Строительная компания	0420520521	15 010 000	1	100	1	100	2025-01-10	2025-01-10
тц. Барс	6227009810	0	1	100	1	0	2025-02-01	2025-02-01

Рисунок 31 – Результат выполнения функции

3. Проверка функции «analize\_brigade». Выполним функцию несколько раз с входными значениями от 15.01.2025 до 05.02.2025, с 15.01.2025, до 05.02.2025 и null (рисунок 32 - 35).

```
select * from analize_brigade('2025-01-15','2025-02-05')
```

A-Z brigadename	I23 sumprice	I23 countapplication	I23 percentapplication	I23 countagreement	I23 percentagreement	firstapplicationdate	lastapplicationdate
Бригада №1	15 010 000	2	100	2	50	2025-01-10	2025-02-01
Бригада №2	1 010 000	1	100	1	100	2025-01-15	2025-01-15

Рисунок 32 – Результат выполнения функции

```
select * from analize_brigade('2025-01-15',null)
```

A-Z brigadename	I23 sumprice	I23 countapplication	I23 percentapplication	I23 countagreement	I23 percentagreement	firstapplicationdate	lastapplicationdate
Бригада №1	16 020 000	3	100	3	66,67	2025-01-10	2025-02-20
Бригада №2	1 282 500	2	100	2	100	2025-01-15	2025-02-10

Рисунок 33 – Результат выполнения функции

```
select * from analize_brigade(null,'2025-02-05')
```

A-Z brigadename	I23 sumprice	I23 countapplication	I23 percentapplication	I23 countagreement	I23 percentagreement	firstapplicationdate	lastapplicationdate
Бригада №1	15 010 000	2	100	2	50	2025-01-10	2025-02-01
Бригада №2	1 010 000	1	100	1	100	2025-01-15	2025-01-15

Рисунок 34 – Результат выполнения функции

```
select * from analize_brigade(null,null)
```

A-Z brigadename	I23 sumprice	I23 countapplication	I23 percentapplication	I23 countagreement	I23 percentagreement	firstapplicationdate	lastapplicationdate
Бригада №1	16 020 000	3	100	3	66,67	2025-01-10	2025-02-20
Бригада №2	1 282 500	2	100	2	100	2025-01-15	2025-02-10

Рисунок 35 – Результат выполнения функции

### 6.3.2 Проверка работы выполняемых процедур

1. Проверка процедуры «new\_client». Данная процедура добавляет нового клиента в таблицу Client и новую учетную запись в таблицу Account (рисунок 36 - 43).

```
CALL new_client('Тестов Иван Игоревич', '8(999)999-99-99', 'test_client@mail.ru', 'secure_pass');
```

	clientid	fio	phone
1	1	Кузьмин Сергей Иванович	8(905)584-98-94
2	2	Евсеев Адам Романович	8(910)952-05-13
3	3	Пахомова Алина Михайловна	8(950)005-31-93
4	4	Мальцев Константин Артёмович	8(942)507-57-05
5	5	Романова Валентина Александровна	8(952)174-27-75
6	6	Тестов Иван Игоревич	8(999)999-99-99

Рисунок 36 – Результат выполнения процедуры

	accountid	mail	password	accesslevelid	clientid	employeeid
1	1	Kyzmin1985@mail.ru	KyzminKyzmin	1	1	[NULL]
2	2	Evseev1990@yandex.ru	Evseev123	1	2	[NULL]
3	3	Alina41001@gmail.ru	Paxomova3012	1	3	[NULL]
4	4	Maltsev1988@mail.ru	Maltsev1988	1	4	[NULL]
5	5	Romanova62@yandex.ru	RomanovaRomanova	1	5	[NULL]
6	6	Centrlsklm@mail.ru	OemDHdZYztVH6pDw7u	2	[NULL]	1
7	7	NikolaCim@yandex.ru	NikolaCim1989	3	[NULL]	2
8	8	RabovNikita@yandex.ru	Nikitata	3	[NULL]	3
9	9	Grigorgrev@gmail.ru	Grigorgrev991	3	[NULL]	4
10	10	Lobanov2000@gmail.ru	Lobanov1100	3	[NULL]	5
11	11	YtkinaLei@mail.ru	YtkinaLeiYtkinaLei	3	[NULL]	6
12	12	test_client@mail.ru	secure_pass	1	6	[NULL]

Рисунок 37 – Результат выполнения процедуры

```
CALL new_client(NULL, '8(999)999-99-98', 'err1@mail.ru', 'pass');
```

SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заказчика: Ошибка: ФИО не может быть NULL  
Где: функция PL/pgSQL new\_client(character varying,character varying,character varying), строка 34, оператор RAISE

Рисунок 38 – Результат выполнения процедуры

```
CALL new_client('Иванов Иван', NULL, 'err2@mail.ru', 'pass');
```

⚠ SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заказчика: Ошибка: Номер телефона не может быть NULL  
Где: функция PL/pgSQL new\_client(character varying,character varying,character varying,character varying), строка 34, оператор RAISE

Рисунок 39 – Результат выполнения процедуры

```
CALL new_client('Дубликат', '8(999)999-99-99', 'err3@mail.ru', 'pass');
```

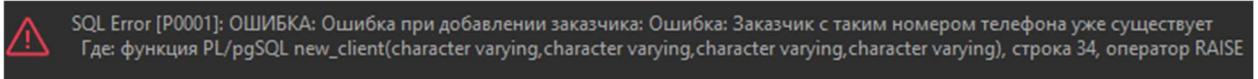


Рисунок 40 – Результат выполнения процедуры

```
CALL new_client('Иванов Иван', '8(999)999-99-97', NULL, 'pass');
```

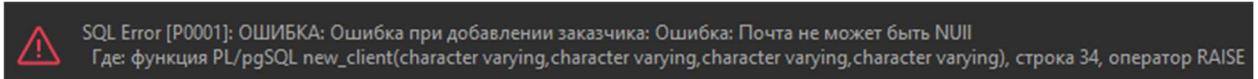


Рисунок 41 – Результат выполнения процедуры

```
CALL new_client('Иванов Иван', '8(999)999-99-96', 'test_client@mail.ru', 'pass');
```

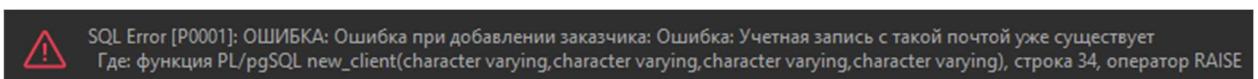


Рисунок 42 – Результат выполнения процедуры

```
CALL new_client('Иванов Иван', '8(999)999-99-95', 'err4@mail.ru', NULL);
```

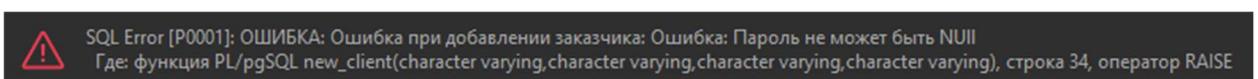


Рисунок 43 – Результат выполнения процедуры

2. Проверка процедуры «new\_application». Данная процедура добавляет новую заявку в таблицу Application и если необходимо новый адрес, организацию и объект обследования (рисунок 44 - 56).

```
CALL new_application('8(999)999-99-99', 'ТестОрг', '123456789012', 'г. Рязань', 'ул. Новая', 'д.99', 200.0);
```

applicationid	clientid	objectsurveyid	brigadeid	incomingdate	statusapplicationid	startdate	enddate
1	1	1	1	2025-01-10	3	2025-01-11	2025-01-15
2	2	2	2	2025-01-15	3	2025-01-16	2025-01-18
3	3	3	1	2025-02-01	3	2025-02-02	2025-02-05
4	4	4	2	2025-02-10	3	2025-02-11	2025-02-13
5	4	5	1	2025-02-20	3	2025-02-21	2025-02-25
6	5	6	2	2025-03-01	2	2025-04-07	[NULL]
7	1	2	1	2025-03-10	2	2025-04-09	[NULL]
8	2	3	2	2025-03-20	1	[NULL]	[NULL]
9	6	9	2	2025-04-10	1	[NULL]	[NULL]

Рисунок 44 – Результат выполнения процедуры

123 addressid	A-Z cityname	A-Z streetname	A-Z number
1	г.Рязань	ул. Интернациональная	д.1
2	г.Рязань	ул. Прижелезнодорожная	д.52
3	г.Рязань	Московское шоссе	д.5А
4	г.Рязань	ул. Бирюзова	д.28А
5	г.Рязань	ул. Интернациональная	д.23
6	г.Рязань	шоссе Солотчинское	д.11
9	г. Рязань	ул. Новая	д.99

Рисунок 45 – Результат выполнения процедуры

123 organizationid	A-Z organizationname	A-Z inn
1	Строительная компания	0420520521
2	Русская кожа	0602065848
3	тц. Барс	6227009810
4	Прайм	6234203335
5	Круиз	6234085000
8	ТестОрг	123456789012

Рисунок 46 – Результат выполнения процедуры

123 objectsurveyid	123 addressid	123 organizationid	123 objectarea
1	1	1	15 000
2	2	2	160 000
3	3	3	36 500
4	4	4	1 500
5	5	4	1 000
6	6	5	27 000
7	9	8	200

Рисунок 47 – Результат выполнения процедуры

```
CALL new_application(NULL, 'ТестОрг', '123456789012', 'г. Рязань', 'ул. Тест', 'д.1', 150);
```

 SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Телефон не может быть NULL  
Где: функция PL/pgSQL new\_application(character varying,character varying,character varying,character varying,character varying,double precision), строка 70, оператор RAISE

Рисунок 48 – Результат выполнения процедуры

```
CALL new_application('8(000)000-00-00', 'ТестОрг', '123456789012', 'г. Рязань', 'ул. Тест', 'д.1', 150);
```

 SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Клиента с указанным номером телефона нет  
Где: функция PL/pgSQL new\_application(character varying,character varying,character varying,character varying,character varying,double precision), строка 70, оператор RAISE

Рисунок 49 – Результат выполнения процедуры

```
CALL new_application('8(999)999-99-99',NULL,'123456789012','г. Рязань','ул. Новая','д.99',200.0);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Наименование организации не может быть NULL  
Где: функция PL/pgSQL new\_application(character varying,character varying,character varying,character varying,character varying,double precision), строка 70, оператор RAISE

Рисунок 50 – Результат выполнения процедуры

```
CALL new_application('8(999)999-99-99', 'ТестОрг', NULL, 'г. Рязань', 'ул. Тест', 'д.1', 150);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: ИНН организации не может быть NULL  
Где: функция PL/pgSQL new\_application(character varying,character varying,character varying,character varying,character varying,double precision), строка 70, оператор RAISE

Рисунок 51 – Результат выполнения процедуры

```
CALL new_application('8(999)999-99-99', 'ТестОрг', '123456789012', NULL, 'ул. Тест', 'д.1', 150);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Город не может быть NULL  
Где: функция PL/pgSQL new\_application(character varying,character varying,character varying,character varying,character varying,double precision), строка 70, оператор RAISE

Рисунок 52 – Результат выполнения процедуры

```
CALL new_application('8(999)999-99-99', 'ТестОрг', '123456789012', 'г. Рязань', NULL, 'д.99', 200.0);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Улица не может быть NULL  
Где: функция PL/pgSQL new\_application(character varying,character varying,character varying,character varying,character varying,double precision), строка 70, оператор RAISE

Рисунок 53 – Результат выполнения процедуры

```
CALL new_application('8(999)999-99-99', 'ТестОрг', '123456789012', 'г. Рязань', 'ул. Новая', NULL, 200.0);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Номер строения не может быть NULL  
Где: функция PL/pgSQL new\_application(character varying,character varying,character varying,character varying,character varying,double precision), строка 70, оператор RAISE

Рисунок 54 – Результат выполнения процедуры

```
CALL new_application('8(999)999-99-99', 'ТестОрг', '123456789012', 'г. Рязань', 'ул. Тест', 'д.1', null);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Площадь объекта не может быть NULL  
Где: функция PL/pgSQL new\_application(character varying,character varying,character varying,character varying,character varying,double precision), строка 70, оператор RAISE

Рисунок 55 – Результат выполнения процедуры

```
CALL new_application('8(999)999-99-99', 'ТестОрг', '123456789012', 'г. Рязань', 'ул. Тест', 'д.1', -20);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Площадь объекта не может быть отрицательной или равняться нулю  
Где: функция PL/pgSQL new\_application(character varying,character varying,character varying,character varying,character varying,double precision), строка 70, оператор RAISE

Рисунок 56 – Результат выполнения процедуры

3. Проверка процедуры «new\_selectedservices». Данная процедура добавляет новую выбранную услугу в таблицу SelectedServices (рисунок 57 - 68).

```
CALL new_selectedservices('8(999)999-99-99','2025-04-10','123456789012','г. Рязань','ул. Новая','д.99','Составление отчета',1);
```

	selectedservicesid	serviceid	applicationid	volume	costservices
1	1	1	1	1	10 000
2		2	1	15 000	15 000 000
3		1	2	1	10 000
4		3	2	5 000	1 000 000
5		1	3	1	10 000
6		4	3	15 000	15 000 000
7		1	4	1	10 000
8		5	4	1 500	262 500
9		1	5	1	10 000
10		2	5	1 000	1 000 000
11		1	6	1	10 000
12		3	6	27 000	5 400 000
13		1	7	1	10 000
14		4	7	15 000	15 000 000
15		1	8	1	10 000
16		5	8	20 000	350 000
17		1	9	1	10 000

Рисунок 57 – Результат выполнения процедуры

```
CALL SelectedServices(NULL, '2025-04-10', '123456789012', 'г. Рязань', 'ул. Новая','д.99','Составление отчета', 1);
```

 SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Телефон не может быть NULL  
Где: функция PL/pgSQL selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 58 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)00-00-00','2025-04-10','123456789012','г. Рязань','ул. Новая','д.99','Составление отчета',1);
```

 SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Клиента с указанным номером телефона нет  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 59 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)999-99-99',NULL,'123456789012','г. Рязань', 'ул. Новая','д.99','Составление отчета',1);
```

 SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Дата не может быть NULL  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 60 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)999-99-99','2025-04-01','123456789012','г. Рязань',
'ул. Новая', 'д.99','Составление отчета',1);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Заявки на эту дату нет  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 61 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)999-99-99', '2025-04-10', '123456789012', 'г.
Рязань', 'ул. Новая', 'д.99', 'Несуществующая услуга', 1);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Указанной услуги нет в каталоге услуг  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 62 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)999-99-99', '2025-04-10', '123456789012', 'г.
Рязань', 'ул. Новая', 'д.99', 'Составление отчета', NULL);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Объем выполняемой работы не может быть NULL  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 63 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)999-99-99', '2025-04-10', '123456789012', 'г.
Рязань', 'ул. Новая', 'д.99', 'Составление отчета', -5);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Объем выполняемой работы не может быть отрицательной или равняться нулю  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 64 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)999-99-99','2025-04-10','123456789012','ГОРОД', 'ул.
Новая', 'д.99','Составление отчета',1);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Указанного адреса нет  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 65 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)999-99-99','2025-04-10','123456789012',NULL, 'ул.
Новая', 'д.99','Составление отчета',1);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Город не может быть NULL  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 66 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)999-99-99','2025-04-10','123456789012','г. Рязань',
NULL, 'д.99','Составление отчета',1);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Улица не может быть NULL  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 67 – Результат выполнения процедуры

```
CALL new_selectedservices('8(999)999-99-99','2025-04-10','123456789012','г. Рязань','ул. Новая',NULL,'Составление отчета',1);
```



SQL Error [P0001]: ОШИБКА: Ошибка при добавлении заявки: Ошибка: Улица не может быть NULL  
Где: функция PL/pgSQL new\_selectedservices(character varying,date,character varying,character varying,character varying,character varying,double precision), строка 73, оператор RAISE

Рисунок 68 – Результат выполнения процедуры

## 6.4 Проверка работы триггеров

### 6.4.1 Проверка триггеров бизнес-правил

1. Проверка триггера `trg_calculate_costservices` данный триггер при добавлении выбранной заказчиком услуги подсчитывает стоимость её оказания за указанный объем (рисунок 69).

```
INSERT INTO SelectedServices(ServiceId, ApplicationId, Volume)
VALUES (1, 1, 3);
```

selectedservicesid	serviceid	applicationid	volume	costservices
1	1	1	1	10 000
2	2	1	15 000	15 000 000
3	1	2	1	10 000
4	3	2	5 000	1 000 000
5	1	3	1	10 000
6	4	3	15 000	15 000 000
7	1	4	1	10 000
8	5	4	1 500	262 500
9	1	5	1	10 000
10	2	5	1 000	1 000 000
11	1	6	1	10 000
12	3	6	27 000	5 400 000
13	1	7	1	10 000
14	4	7	15 000	15 000 000
15	1	8	1	10 000
16	5	8	20 000	350 000
19	1	9	1	10 000
20	1	1	3	30 000

Рисунок 69 – Результат работы триггера

2. Проверка триггера `trg_calculate_pricefororder` данный триггер при добавлении договора подсчитывает стоимость оказания всех оказанных услуг (рисунок 70).

```
INSERT INTO SurveyAgreement(ApplicationId, ReportId, EmployeeId, CreateDate, Confirmation)
VALUES (2, 1, 2, CURRENT_DATE, FALSE);
```

	surveyagreementid	applicationid	reportid	employeeid	createdate	confirmation	pricefororder
1	1	1	1	2	2025-01-15	[v]	15 010 000
2	2	2	2	4	2025-01-18	[v]	1 010 000
3	3	3	3	3	2025-02-05	[ ]	15 010 000
4	4	4	4	5	2025-02-13	[v]	272 500
5	5	5	5	2	2025-02-25	[v]	1 010 000
6	6	2	1	2	2025-04-10	[ ]	1 010 000

Рисунок 70 – Результат работы триггера

3. Проверка триггера `trg_assign_brigade` данный триггер при добавлении заявки назначает на нее исполнительную бригаду (рисунок 71).

```
INSERT INTO Application(ClientId, ObjectSurveyId, IncomingDate, StatusApplicationId)
VALUES (1, 1, CURRENT_DATE, 1);
```

applicationid	clientid	objectsurveyid	brigadeid	incomingdate	statusapplicationid	startdate	enddate
1	1	1	1	2025-01-10	3	2025-01-11	2025-01-15
2	2	2	2	2025-01-15	3	2025-01-16	2025-01-18
3	3	3	1	2025-02-01	3	2025-02-02	2025-02-05
4	4	4	2	2025-02-10	3	2025-02-11	2025-02-13
5	4	5	1	2025-02-20	3	2025-02-21	2025-02-25
6	5	6	2	2025-03-01	2	2025-04-07	[NULL]
7	1	2	1	2025-03-10	2	2025-04-09	[NULL]
8	2	3	2	2025-03-20	1	[NULL]	[NULL]
9	6	9	2	2025-04-10	1	[NULL]	[NULL]
10	1	1	1	2025-04-10	1	[NULL]	[NULL]

Рисунок 71 – Результат работы триггера

#### 6.4.2 Проверка триггеров бизнес-ограничений

1. Проверка триггера `trg_check_status_before_report` данный триггер перед добавлением отчета проверяет что статус заявки является “Закончена” (рисунок 72).

```
INSERT INTO SurveyReport(ApplicationId, EmployeeId, FileReport)
VALUES (8, 3, '/reports/should_fail.pdf');
```

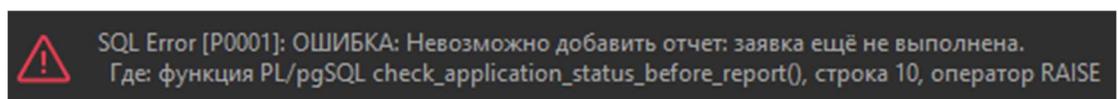


Рисунок 72 – Результат работы триггера

2. Проверка триггера `trg_check_before_agreement` данный триггер при добавлении договора проверяет что статус заявки является “Закончена” и отчет сформирован (рисунок 73).

```
INSERT INTO SurveyAgreement(ApplicationId, ReportId, EmployeeId, CreateDate, Confirmation)
VALUES (7, 99, 3, CURRENT_DATE, FALSE);
```

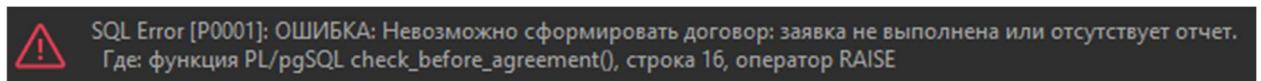


Рисунок 73 – Результат работы триггера

3. Проверка триггера `trg_check_agreement_date` данный триггер при добавлении договора проверяет что дата его формирования не раньше даты окончания связанного с ним обследования (рисунок 74).

```
INSERT INTO SurveyAgreement(ApplicationId, ReportId, EmployeeId, CreateDate, Confirmation)
VALUES (4, 4, 5, '2025-02-10', FALSE);
```

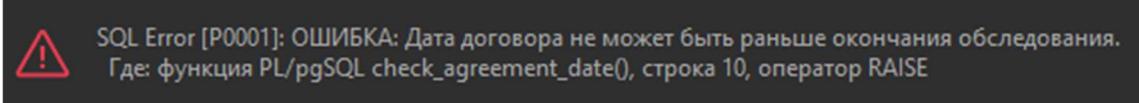


Рисунок 74 – Результат работы триггера

#### 6.4.3 Проверка триггера ограничения целостности

Проверка триггера `trg_prevent_report_delete` данный триггер запрещает удаление отчета если связанный с ним договор подтвержден клиентом (рисунок 75).

```
DELETE FROM SurveyReport WHERE ReportId = 1;
```

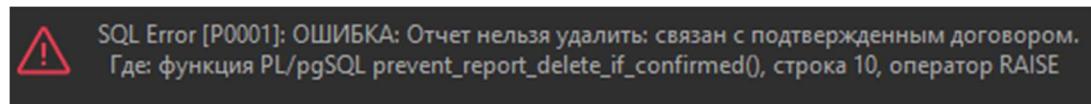


Рисунок 75 – Результат работы триггера

#### 6.4.4 Проверка триггеров ограничения действия

1. Проверка триггера `trg_check_employee_brigade` данный триггер запрещает добавление отчета сформированный сотрудником не состоящим в бригаде, которая выполняла заявку (рисунок 76).

```
INSERT INTO SurveyReport(ApplicationId, EmployeeId, FileReport)
```

```
VALUES (3, 6, '/reports/wrong_brigade.pdf');
```



SQL Error [P0001]: ОШИБКА: Сотрудник не состоит в бригаде, выполнившей заявку.  
Где: функция PL/pgSQL check\_employee\_brigade\_for\_report(), строка 15, оператор RAISE

Рисунок 76 – Результат работы триггера

2. Проверка триггера `trg_prevent_agreement_delete` данный триггер запрещает удалить договор если его подтвердил клиент (рисунок 77).

```
DELETE FROM SurveyAgreement WHERE SurveyAgreementId = 1;
```



SQL Error [P0001]: ОШИБКА: Нельзя удалить подтвержденный договор.  
Где: функция PL/pgSQL prevent\_agreement\_delete\_if\_confirmed(), строка 4, оператор RAISE

Рисунок 77 – Результат работы триггера

#### 6.4.5 Проверка триггера DDL

Проверка триггера `drop_table_restrict` данный триггер предотвращает удаление таблиц всеми пользователями кроме `postgres`. Для проверки создадим пользователя с ролью администратора и попытаемся от него удалить таблицу `SelectedServices` (рисунок 78).

```
-- от лица postgres
create USER admin1 WITH PASSWORD '123'
grant "Admin" to admin1

-- от лица admin1
drop table selectedservices
```

SQL Error [P0001]: ОШИБКА: Удаление таблиц разрешено только пользователю postgres!  
Где: функция PL/pgSQL prevent\_drop\_table(), строка 4, оператор RAISE

Рисунок 78 – Результат работы триггера

#### 6.4.6 Проверка триггера БД

Проверка триггера `log_user` данный триггер логирует подключение к бд в таблицу `userlog` (рисунок 79).

	123 id	A-Z usern	datetime
1	1	postgres	2025-04-10 11:51:04.146 +0300
2	2	postgres	2025-04-10 11:51:04.177 +0300
3	3	postgres	2025-04-10 12:19:55.081 +0300
4	4	postgres	2025-04-10 12:20:02.768 +0300
5	5	postgres	2025-04-10 13:05:25.688 +0300
6	6	postgres	2025-04-10 19:23:40.467 +0300
7	7	postgres	2025-04-10 19:23:40.607 +0300
8	8	postgres	2025-04-10 19:23:40.714 +0300
9	9	postgres	2025-04-10 19:35:53.181 +0300
10	10	postgres	2025-04-10 19:37:11.902 +0300
11	11	postgres	2025-04-10 19:38:31.960 +0300
12	12	postgres	2025-04-10 19:38:32.749 +0300
13	13	postgres	2025-04-10 19:42:02.337 +0300
14	14	admin1	2025-04-10 22:21:10.696 +0300
15	15	admin1	2025-04-10 22:21:10.726 +0300
16	16	admin1	2025-04-10 22:21:13.824 +0300

Рисунок 79 – Результат работы триггера

## 6.5 Проверка реализации политики безопасности

Описание создаваемых ролей «Клиент», «Сотрудник», «Администратор» и прав, присваиваемых им, приведено в п.4.4.

Проверка прав роли «Администратор» (рисунок 80 - 82).

```
SELECT grantee, privilege_type, table_schema, table_name
FROM information_schema.role_table_grants
WHERE grantee = 'Admin';
```

A-Z grantee	A-Z privilege_type	A-Z table_schema	A-Z table_name
Admin	INSERT	public	account
Admin	SELECT	public	account
Admin	UPDATE	public	account
Admin	DELETE	public	account
Admin	TRUNCATE	public	account
Admin	REFERENCES	public	account
Admin	TRIGGER	public	account
Admin	INSERT	public	userlog
Admin	SELECT	public	userlog
Admin	UPDATE	public	userlog
Admin	DELETE	public	userlog
Admin	TRUNCATE	public	userlog
Admin	REFERENCES	public	userlog
Admin	TRIGGER	public	userlog
Admin	INSERT	public	jobtitle
Admin	SELECT	public	jobtitle
Admin	UPDATE	public	jobtitle
Admin	DELETE	public	jobtitle
Admin	TRUNCATE	public	jobtitle
Admin	REFERENCES	public	jobtitle
Admin	TRIGGER	public	jobtitle
Admin	INSERT	public	employee
Admin	SELECT	public	employee
Admin	UPDATE	public	employee
Admin	DELETE	public	employee
Admin	TRUNCATE	public	employee
Admin	REFERENCES	public	employee
Admin	TRIGGER	public	employee
Admin	INSERT	public	brigade
Admin	SELECT	public	brigade
Admin	UPDATE	public	brigade
Admin	DELETE	public	brigade
Admin	TRUNCATE	public	brigade
Admin	REFERENCES	public	brigade
Admin	TRIGGER	public	brigade

Рисунок 80 – Права роли Admin

A-Z grantee	A-Z privilege_type	A-Z table_schema	A-Z table_name
Admin	INSERT	public	servicecatalog
Admin	SELECT	public	servicecatalog
Admin	UPDATE	public	servicecatalog
Admin	DELETE	public	servicecatalog
Admin	TRUNCATE	public	servicecatalog
Admin	REFERENCES	public	servicecatalog
Admin	TRIGGER	public	servicecatalog
Admin	INSERT	public	objectsurvey
Admin	SELECT	public	objectsurvey
Admin	UPDATE	public	objectsurvey
Admin	DELETE	public	objectsurvey
Admin	TRUNCATE	public	objectsurvey
Admin	REFERENCES	public	objectsurvey
Admin	TRIGGER	public	objectsurvey
Admin	INSERT	public	selectedservices
Admin	SELECT	public	selectedservices
Admin	UPDATE	public	selectedservices
Admin	DELETE	public	selectedservices
Admin	TRUNCATE	public	selectedservices
Admin	REFERENCES	public	selectedservices
Admin	TRIGGER	public	selectedservices
Admin	INSERT	public	application
Admin	SELECT	public	application
Admin	UPDATE	public	application
Admin	DELETE	public	application
Admin	TRUNCATE	public	application
Admin	REFERENCES	public	application
Admin	TRIGGER	public	application
Admin	INSERT	public	statusapplication
Admin	SELECT	public	statusapplication
Admin	UPDATE	public	statusapplication
Admin	DELETE	public	statusapplication
Admin	TRUNCATE	public	statusapplication
Admin	REFERENCES	public	statusapplication
Admin	TRIGGER	public	statusapplication

Рисунок 81 – Права роли Admin

Admin	INSERT	public	accesslevel
Admin	SELECT	public	accesslevel
Admin	UPDATE	public	accesslevel
Admin	DELETE	public	accesslevel
Admin	TRUNCATE	public	accesslevel
Admin	REFERENCES	public	accesslevel
Admin	TRIGGER	public	accesslevel
Admin	INSERT	public	client
Admin	SELECT	public	client
Admin	UPDATE	public	client
Admin	DELETE	public	client
Admin	TRUNCATE	public	client
Admin	REFERENCES	public	client
Admin	TRIGGER	public	client
Admin	INSERT	public	organization
Admin	SELECT	public	organization
Admin	UPDATE	public	organization
Admin	DELETE	public	organization
Admin	TRUNCATE	public	organization
Admin	REFERENCES	public	organization
Admin	TRIGGER	public	organization
Admin	INSERT	public	address
Admin	SELECT	public	address
Admin	UPDATE	public	address
Admin	DELETE	public	address
Admin	TRUNCATE	public	address
Admin	REFERENCES	public	address
Admin	TRIGGER	public	address

Рисунок 82 – Права роли Admin

Проверка прав роли «Сотрудник» (рисунок 83).

```
SELECT grantee, privilege_type, table_schema, table_name
FROM information_schema.role_table_grants
WHERE grantee = 'Employee';
```

A-Z grantee ▾	A-Z privilege_type ▾	A-Z table_schema ▾	A-Z table_name ▾
Employee	SELECT	public	brigade
Employee	SELECT	public	servicecatalog
Employee	SELECT	public	objectsurvey
Employee	SELECT	public	selectedservices
Employee	INSERT	public	application
Employee	SELECT	public	application
Employee	UPDATE	public	application
Employee	SELECT	public	client
Employee	SELECT	public	organization
Employee	SELECT	public	address

Рисунок 83 – Права роли Employee

Проверка прав роли «Клиент» (рисунок 84).

```
SELECT grantee, privilege_type, table_schema, table_name
FROM information_schema.role_table_grants
WHERE grantee = 'Client';
```

A-Z grantee	A-Z privilege_type	A-Z table_schema	A-Z table_name
Client	SELECT	public	servicecatalog
Client	INSERT	public	selectedservices
Client	INSERT	public	application
Client	SELECT	public	application
Client	INSERT	public	organization
Client	SELECT	public	organization
Client	UPDATE	public	organization
Client	INSERT	public	address
Client	SELECT	public	address
Client	UPDATE	public	address

Рисунок 84 – Права роли Client

## **Заключение**

В ходе разработки курсового проекта решены следующие задачи:

- описана предметную область;
- описаны и проанализированы бизнес-процессы предметной области;
- выявлены, проанализированы и сформулированы требования и ограничения;
- разработана функциональную модель;
- разработаны диаграммы потоков данных;
- разработана концептуальную модель;
- разработана глобальную логическую модель базы данных;
- разработана глобальную физическую модель базы данных;
- реализованы выявленные требования с помощью инструментария СУБД PostgreSQL;
- протестирована база данных.

Так как все поставленные задачи, необходимые для достижения цели курсового проекта - разработка программного обеспечения инжиниринга базы данных информационной системы центра исследования строительных конструкций, были выполнены, можно сделать вывод, что цель курсового проекта достигнута.

## **Библиографический список**

1. Модель автоматизации управления процессом обследования несущих строительных конструкций // Научный журнал "Инновации в науке". URL: <https://naukaru.ru/ru/nauka/article/88493/view> (дата обращения: 20.01.2025).
2. Аниanova И.С. Разработка информационной системы заявок на ремонт техники: дипломная работа. – Орёл: Орловский государственный университет экономики и торговли, 2023. – 42 с. URL: [https://elar.usfeu.ru/bitstream/123456789/12716/1/Anianova\\_23.pdf](https://elar.usfeu.ru/bitstream/123456789/12716/1/Anianova_23.pdf) (дата обращения: 20.01.2025).
3. Информационная система учета, контроля и анализа выполнения заказов по сборке и установке кухонной мебели: дипломная работа / Рощупкина А.А. — Томск: Томский политехнический университет, 2019. — 58 с. URL: <https://earchive.tpu.ru/bitstream/11683/27463/1/TPU193358.pdf> (дата обращения: 20.01.2025).
4. Применение системы мониторинга технического состояния строительных конструкций // RIOR Научные публикации. URL: <https://riorpublishing.com/ru/nauka/article/71994/view> (дата обращения: 20.01.2025).
5. Бербанк Д. Data Modeling Made Simple with CA ERwin Data Modeler r8 . - 1-е изд. - Technics Publications, LLC, 2011. - 536 с.
6. МЕТОДОЛОГИЯ ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЯ IDEF0 // URL: [https://www.businessstudio.ru/upload/files/notation\\_idef0.pdf](https://www.businessstudio.ru/upload/files/notation_idef0.pdf) (дата обращения: 15.02.2025).
7. DFD (Data Flow Diagram) Диаграммы — зачем они нужны и какие бывают // Habr URL: <https://habr.com/ru/articles/668684/> (дата обращения: 15.01.2025).

8. Использование DFD: как описать движение данных в бизнес-процессах // System education URL: <https://systems.education/data-flow-diagrams> (дата обращения: 15.02.2025).

9. Концептуальное проектирование // Википедия URL: [https://ru.wikipedia.org/wiki/Концептуальное\\_проектирование](https://ru.wikipedia.org/wiki/Концептуальное_проектирование) (дата обращения: 15.02.2025).

10. Основные понятия ER – диаграмм: сущность, экземпляр сущности, атрибуты сущности, ключ сущности, связь // Студопедия URL: [https://studopedia.ru/8\\_43410\\_osnovnie-ponyatiya-ER--diagramm-sushchnost-ekzempliar-sushchnosti-atributi-sushchnosti-klyuch-sushchnosti-svyaz.html](https://studopedia.ru/8_43410_osnovnie-ponyatiya-ER--diagramm-sushchnost-ekzempliar-sushchnosti-atributi-sushchnosti-klyuch-sushchnosti-svyaz.html) (дата обращения: 16.02.2025).

11. Коннолли Т., Бегг К., Страchan А. Базы данных: проектирование, реализация и сопровождение. — М.: Издательский дом «Вильямс», 2011. — 1488 с.

12. Маркин А.В. Программирование на SQL в 2 ч. Часть 1. - 2-е изд. - Москва: Издательство Юрайт, 2019. - 403 с.

13. Маркин А.В. Программирование на SQL в 2 ч. Часть 2. - 2-е изд. - Москва: Издательство Юрайт, 2019. - 403 с.