

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **КУРСОВАЯ РАБОТА** | | | | | | | | | | | | | | |
| по: | ПМ. 11 Разработка, администрирование и защита баз данных | | | | | | | | | | | | | |
| междисциплинарного курса: | | | | | | | МДК.11.01 Технология разработки и защиты баз данных | | | | | | | |
| на тему: | | | | | | | | | | | | | | |
| «Разработка и автоматизация базы данных для учёта планируемых и реализованных объектов недвижимости» | | | | | | | | | | | | | | |
| Выполнена студентом: | | | | | | Максимом Павловичем Кириллиным | | | | | | | | |
|  | | | | | | И.О. Фамилия | | | | | | | | |
| Группы: | | 4ИСП9-14 | | | | | | | | | | | | |
|  |  | номер группы | | | | | | | | | | | | |
| Основная профессиональная образовательная программа по специальности 09.02.07 Информационные системы и программирование (администратор баз данных) | | | | | | | | | | | | | | |
| шифр и наименование специальности | | | | | | | | | | | | | | |
| Форма обучения | | | | | очная | | | | | | | | | |
| Руководитель | | |  | преподаватель | | | | |  | Олеся Павловна Куропаткина | | | | |
|  | | | ученая степень, должность | | | | | | | И.О. Фамилия | | | | |
|  | | |  | | | | | | | оценка | | | | |
|  | | |  | | | | | | | подпись, дата | | | | |
| Зав. отделением | | |  |  | | | | |  | Ирина Юрьевна Васильева | | | | |
|  | | |  | | | | | | | И.О. Фамилия | | | | |
|  |  |  | | | |  | |  | подпись, дата | | | | | |
|  |  |  | | | |  | |  |  | |  |  |  |  |
| г. Москва, 2024 г.   |  |  | | --- | --- | |  | **УТВЕРЖДАЮ**  Зам.директора по ОУП  \_\_\_\_\_\_\_\_\_\_И.Г.Бозрова  «\_\_\_»\_\_\_\_\_\_\_\_\_\_2024 г. |   **ЗАДАНИЕ**  к курсовой работе студента  Кириллина Максима Павловича  на тему: «Разработка и автоматизация базы данных для учёта планируемых и реализованных объектов недвижимости»   |  |  |  |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | Тема курсовой работы утверждена приказом директора № \_\_\_ от \_\_\_\_\_\_\_\_\_\_\_ | | | | | | | | | | | | | | Цель работы: | | | Проектирование и разработка базы данных для учёта планируемых и реализованных объектов недвижимости. | | | | | | | | | | | Основные вопросы, подлежащие разработке: | | | | | | | | | | | | | | 1. Проектирование базы данных. | | | | | | | | | | | | | | 2. Разработка и автоматизация базы данных в конкретных СУБД. | | | | | | | | | | | | | | Теоретическая часть: | | | | |  | | | | | | | | | 1. Анализ предметной области. | | | | | | | | | | | | | | 2. Анализ и обоснование выбора средств проектирования и разработки базы данных. | | | | | | | | | | | | | | Практическая часть: | | | | |  | | | | | | | | | 1. Проектирование базы данных. | | | | | | | | | | | | | | 2. Разработка базы данных в СУБД. | | | | | | | | | | | | | | 3. Создание объектов с целью автоматизации работы базы данных. | | | | | | | | | | | | | | Основная литература: | | | | | | | | | | | | | | 1. Разработка, администрирование и защита баз данных: учебник. Федорова Г.Н., Академия, 2021 г. | | | | | | | | | | | | | | 2. Базы данных. Проектирование, программирование, управление и администрирование. Волк В.К., Лань, 2021 г. | | | | | | | | | | | | | | 3. Базы данных: модели, разработка, реализация: учебное пособие. Карпова Т. С., ИНТУИТ, 2022 г. | | | | | | | | | | | | | | Руководитель: | | | | Олеся Павловна Куропаткина | | | | | | |  | | |  |  | | |  | |  |  |  |  |  |  | подпись | | Дата выдачи задания: | | | | | | дд.мм.2024 г. | | | | | | | | Задание получил: | | | | Максим Павлович Кириллин | | | | | | |  |  | |  | |  | |  | | | | | | |  | подпись | | Дата получения задачи: | | | | | | дд.мм.2024 г. | | | | | | | | | | | | | | | | | | | | | |

**ПЛАН-ГРАФИК**

курсовой работы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Срок выполнения курсовой работы | |  | | |
| Выбор темы курсовой работы | |  | | |
| Подбор и анализ исходной информации | |  | | |
| Подготовка и утверждение плана курсовой работы | |  | | |
| Выполнение задач курсовой работы | |  | | |
| Оформление курсовой работы | |  | | |
| Исправление замечаний курсовой работы | |  | | |
| Предоставление готовой курсовой работы для рецензирования | |  | | |
| Предзащита курсовой работы | |  | | |
| Защита курсовой работы | |  | | |
|  |  | |  |  |

|  |  |
| --- | --- |
| Руководитель | Олеся Павловна Куропаткина |
|  | подпись |
| План принял к исполнению | Максим Павлович Кириллин |
|  | подпись |
|  | дд.мм.2024 г. |
|  |  |

**СОДЕРЖАНИЕ**

**ВВЕДЕНИЕ**

Городское планирование — это процесс разработки и реализации стратегий и планов, нацеленных на улучшение использования городской территории, обеспечение экономического развития и качества жизни горожан. В современном мире крайне важной является работа автоматизированных систем, ведь благодаря им большое количество процессов можно оптимизировать, повысить эффективность. Многие процессы в целом невозможны без автоматизированных систем, так как человек просто не может обрабатывать такой объем данных без посторонней помощи компьютера.

Таким образом удастся сэкономить время и исключить ошибки, более того можно автоматизировать и сделать удобным весь процесс управления какой-либо системой.

Для информационной системы управления процессом городского планирования и развития, она может быть полезна следующим:

* понятное и доступное описание всех видов планирования;
* учет объектов недвижимости и земельных участков;
* управление заказами;
* наглядное расписание предоставляемых услуг.

Актуальность данной курсовой работы обусловлена необходимостью создания автоматизированной системы для городского планирования и повышения её эффективности.

Целью курсовой работы является систематизация и закрепление полученных теоретических знаний и практических умений, проектирование и разработка базы данных для учёта планируемых и реализованных объектов недвижимости, для организации, осуществляющей услуги по планированию районов и их урбанизации.

Для достижения цели были поставлены следующие задачи:

* изучение особенностей конкретной предметной области, относящихся к теме курсовой работы;
* анализ программных средств с обоснованием выбора;
* проектирование и разработка базы данных минимум в двух СУБД;
* создание объектов базы (представления, хранимые процедуры, пользовательские функции, триггеры) данных для автоматизации работы;
  + анализ полученных результатов работы разработанного программного обеспечения.

Объектом исследования является система управления процессом городского планирования и развития.

Предметом исследования является разработка базы данных для автоматизации системы управления процессом городского планирования и развития.

**ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ПРИМЕНЯЕМЫХ ПРОГРАММНЫХ СРЕДСТВ**

1.1 Анализ предметной области

В рамках данной курсовой работы рассматривается предметная область организации, осуществляющей управление процессом городского планирования, развития и продажи недвижимости.

Основной деятельностью организации, осуществляющей процесс городского планирования, развития и продажи недвижимости является предоставление информации о недвижимости, земельных участках и возможности её приобретения. К таким услугам относятся:

* возможность просмотра доступных строений и земельных участков со стороны организации;
* возможность планировки строений со стороны организации;
* просмотр информации о недвижимости со стороны организации и клиента;
* покупка недвижимости со стороны клиента.

Данная автоматизированная система предназначена для оптимизации работы сотрудников и для удобства выбора недвижимости организации. Сотрудники будут разделены в соответствии с их должностями.

Роль менеджера – учёт, распределение и продажа недвижимости клиентам (физические и юридические лица).

Кадастровый инженер – это специалист, который производит межевание территории и вносит актуальную информацию в систему.

Архитектор – специалист, который создаёт планировки домов и квартир, внутри системы архитектор сможет добавлять планировки домов и квартир, указывать информацию о них.

Клиент – юридическое или физическое, на которого ориентирован весь проект. Внутри системы он сможет просматривать информацию о районе, домах и земельных участках, в следствии выбрать недвижимость для покупки или аренды.

Анализ предметной области для учета планируемых и реализованных объектов недвижимости включает в себя анализ факторов, влияющих на успешность реализации объектов недвижимости. Кроме того, важно иметь систему отчетности, которая позволяет отслеживать выполнение плановых показателей и анализировать результаты проектов недвижимости.

1.2 Сравнительный анализ используемых программных средств

При выборе СУБД круг поиска был сужен до трех. Было решено рассмотреть такие СУБД как:

* Microsoft SQL Server;
* PostgreSQL;
* Elasticsearch.

1.2.1 Анализ СУБД Elasticsearch

Elasticsearch – это масштабируемое не реляционное хранилище данных с открытым исходным кодом, аналитическая NoSQL-СУБД с широким набором функций полнотекстового поиска.

Достоинства:

* высокая скорость поиска и аналитики данных;
* масштабируемость и отказоустойчивость;
* удобный интерфейс для работы с большими объемами неструктурированных данных;
* поддержка множества типов поисковых запросов, включая текстовый и геолокационный;
* интеграция с различными языками программирования и платформами;
* возможность создания репликации данных для увеличения производительности;

Недостатки:

* высокие нагрузки на ресурсы сервера;
* сложность в процессе настройки и управления
* не поддерживает транзакции баз данных;
* требует дополнительных усилий для обеспечения безопасности и защиты данных;
* не всегда подходит для структурированных данных;
* высокая стоимость масштабирования при больших объемах данных;

1.2.2 Анализ СУБД Microsoft SQL Server

Microsoft SQL Server — это реляционная система управления базами данных (СУБД), разработанная корпорацией Microsoft. Она предлагает мощные инструменты для хранения, управления и анализа больших объемов структурированных данных.

Достоинства:

1. Надежность и безопасность: SQL Server обеспечивает высокую надежность и безопасность данных с помощью различных механизмов шифрования, аутентификации и управления доступом.
2. Высокая производительность: SQL Server обладает хорошей производительностью при обработке транзакционных запросов и аналитических операций.
3. Интеграция с другими продуктами Microsoft: SQL Server встраивается в экосистему Microsoft, что облегчает интеграцию с другими продуктами и средствами разработки компании.
4. Значительные возможности аналитики: предоставляет инструменты для бизнес-аналитики, включая OLAP-возможности и сервис отчетов.
5. Поддержка расширений: SQL Server поддерживает хранимые процедуры, триггеры, представления и другие средства для управления данными.

Недостатки:

1. Лицензирование и стоимость: лицензирование SQL Server может быть дорогим для небольших компаний и стартапов.
2. Ограничения операционных систем: некоторые функции могут быть ограничены в зависимости от используемой операционной системы.
3. Сложность конфигурации: настройка и администрирование SQL Server может требовать некоторой экспертизы.
4. Ограниченное расширяемость: некоторые функции и возможности могут быть ограничены и не так легко расширяемы, как в других СУБД.

1.2.3 Анализ СУБД PostgreSQL

PostgreSQL — это мощная объектно-реляционная система управления базами данных (СУБД), которая обладает открытым исходным кодом и широко используется в масштабных проектах.

Достоинства:

1. Расширяемость и гибкость: PostgreSQL предлагает богатые возможности для расширения и настройки функций, позволяя разработчикам создавать собственные типы данных, функции и расширения.
2. Поддержка стандарта SQL: PostgreSQL стремится к полной совместимости со стандартом SQL, что обеспечивает удобство именно для разработчиков, привыкших работать с SQL-запросами.
3. Надежность и целостность данных: PostgreSQL обеспечивает надежность данных, благодаря полной поддержке ACID и транзакционной целостности.
4. Комплексная документация и активное сообщество: PostgreSQL обладает хорошей документацией и развитым сообществом пользователей и разработчиков.

Недостатки:

1. Не самая высокая производительность: в сравнении с некоторыми другими СУБД, PostgreSQL может показывать несколько более низкую производительность при определенных типах запросов или операций.
2. Управление ресурсами: в некоторых случаях управление ресурсами, такими как память, процессор и дисковое пространство, может потребовать дополнительной оптимизации и настройки.
3. Сложность некоторых операций: Некоторые операции, такие как репликация и бэкап, могут потребовать более сложной настройки и администрирования.

Вывод

Из представленных СУБД для моего проекта Elasticsearch наименее всего подходит для исполнения поставленной задачи по следующим причинам:

1. Elasticsearch является не реляционной СУБД. Реляционные базы данных можно рекомендовать для обслуживания любых информационных потребностей, где элементы данных связаны между собой и необходимо обеспечивать безопасное и надежное управление ими на основе правил целостности.
2. Elasticsearch наименее популярна нежели Microsoft SQL Server и PostgreSQL, следовательно количество людей, способных грамотно обслуживать СУБД будет гораздо меньше, а затрат на их работу будет значительно больше.
3. Производительность при работе с транзакционными данными: Microsoft SQL Server и PostgreSQL выигрывают у Elasticsearch при обработке транзакционных запросов и операций записи.
4. SQL-запросы: Обе базы данных поддерживают SQL, что делает их более удобными для работы с данными приложения.
5. Транзакционная поддержка: Microsoft SQL Server и PostgreSQL обеспечивают ACID-совместимость и полную поддержку транзакций, что позволяет обеспечить целостность данных.
6. Широкие возможности аналитики данных: Обе базы данных предоставляют мощные инструменты для аналитики и отчетности, включая поддержку различных аналитических функций и OLAP-возможностей.
7. Сохранение данных: Microsoft SQL Server и PostgreSQL обладают возможностью хранения и обработки большого объема структурированных данных.
8. Поддержка XML и JSON: Обе базы данных имеют поддержку структурированных данных в форматах XML и JSON, что является важным функционалом для современных приложений.
9. Поддержка расширений и хранимых процедур: Microsoft SQL Server и PostgreSQL имеют возможность работы с хранимыми процедурами и расширениями, что расширяет их функциональность.
10. Расширенные возможности безопасности: Обе базы данных обеспечивают широкий спектр функций для обеспечения безопасности данных и аутентификации пользователей.
11. Сообщество и поддержка: Обе системы имеют активное сообщество пользователей и разработчиков, что обеспечивает доступ к обширной базе знаний и поддержке при возникновении проблем.
12. Поддержка транзакционных и консистентных данных. Microsoft SQL Server и PostgreSQL — это хороший выбор для хранения и обработки консистентных и транзакционных данных, в то время как Elasticsearch больше специализируется на масштабируемости и поиске.

Таким образом можно сказать, что Microsoft SQL Server и PostgreSQL обе представляют собой мощные и надежные системы управления базами данных. Обе СУБД обладают расширенным функционалом, обеспечивают высокую производительность, имеют обширные возможности аналитики данных, обеспечивают надежность и безопасность данных. Исходя из данного заключения я могу создать базу данных для учёта планируемых и реализованных объектов недвижимости в соответствии с целью своего проекта.

**ГЛАВА 2. ПРОЕКТИРОВАНИЕ, РАЗРАБОТКА И АВТОМАТИЗАЦИЯ РАБОТЫ БАЗЫ ДАННЫХ**

Проведя анализ предметной области и применяемых программных средств, приступаем к процессу проектирования базы данных. На этапе исследования особенностей предметной области определены основные объекты, подлежащие учету.

2.1 Проектирование базы данных

Разработка проекта базы данных предполагает несколько этапов. Первый – концептуальный, в рамках которого осуществляется анализ предметной области, выделяются основные процессы и объекты, подлежащие учету. Далее следует этапы логического и физического проектирования базы данных, рассмотрим их подробнее.

2.1.1 Логическая модель базы данных

Логическая модель - графическое представление структуры базы данных  
с учетом принимаемой модели данных (иерархической, сетевой, реляционной и  
т.д.), независимое от конечной реализации базы данных и аппаратной  
платформы.

Моя логическая модель базы данных была разработана в приложении Visio.

Организации нужно хранить информацию о всех сделках, проведённых между клиентами, а также информацию о учёте объектов недвижимости. Клиент может быть как физическим, так и юридическим лицом, поэтому у клиента есть данные, принадлежащие как физическим, так и юридическим лицам. Объекты недвижимости представляют собой участки, строения и квартиры, при этом необходимо учитывать, какая конкретная квартира находится в конкретном доме, который находится на конкретном участке, при этом дома могут не иметь квартир, а участки могут не иметь домов. Предусмотрена возможность добавления новых объектов недвижимости архитекторами, которые могут добавлять фотографии объектам.

На основе этой задачи были выделены ключевые сущности: объект недвижимости, клиент, работник, сделка. И дополнительные: пол, должность работника, почтовый индекс, форма собственности, тип объекта, фотографии объекта, связи между участками и домами, связи между домами и квартирами.

Первой из основных сущностей является сущность «Client» (рис. 2.1),  
которая нужна для формирования клиентской базы. Она хранит в себе всю  
основную информацию о клиенте, такие как: ФИО, Email, дата  
рождения, номер телефона, поле, которое показывает является ли клиент физическим или юридическим лицом, паспортные данные в случае, если клиент физическое лицо, данные о компании в случае, если клиент юридическое лицо, пол клиента.

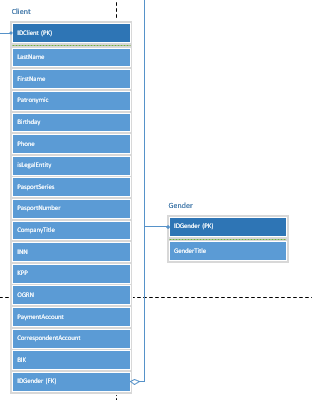


Рисунок 2.1 Сущности Client (Пользователь/Клиент) и Gender (Пол)

Полная диаграмма спроектированной базы данных находится в Приложении 1.

2.1.2 Физическая модель базы данных в СУБД Microsoft SQL Server

Сущности создавались посредством визуальных инструментов - через  
создание диаграммы базы данных в приложении Microsoft SQL Server Management studio. Для текстовых полей использовались типы данных nvarchar (информация нефиксированной длины с использованием кодировки UNICODE) и char (информация фиксированной длины). Для числовых полей - integer (ID сущностей) и decimal (цена, площадь). К таким датам как день рождения и дата применялись типы данных date и datetime. К полям, предполагающим значения логического типа, применялся тип bit (1 - активен, 0 - неактивен). На диаграмме можно увидеть ограничения первичного ключа, внешнего и запрет на ввод пустых значений. Ограничение первичного ключа необходимо для идентификации записей, а ограничение внешнего для обеспечения целостности данных.

Сущность «EstateObject» (рис 2.2) хранит всю основную информацию об объектах недвижимости: площадь, цена, дата начала строительства, дата ввода в эксплуатацию, адрес, номер, почтовый индекс, тип недвижимости (участок, квартира, строение), формат объекта недвижимости (частная, государственная, муниципальная) (рис 2.3). Для сущности «EstateObject» было создано проверочное ограничение на поле IDEstateObject для соблюдения уникальности и целостности данных, были созданные вспомогательные сущности EstateRelation и FlatRelation, они необходимы для указания какому дому принадлежит какая квартира, и какому участку принадлежит какой дом.

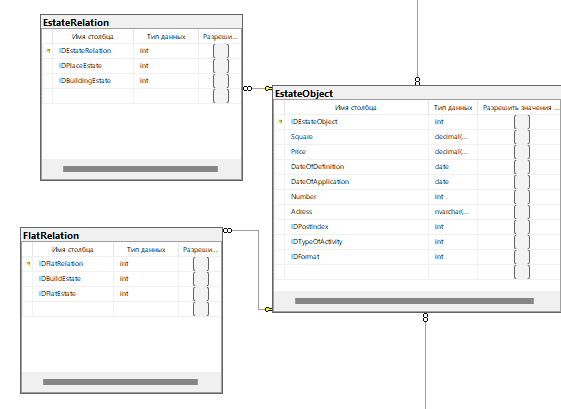


Рисунок 2.2 Сущность EstateObject (Объект недвижимости)

Создан внешний ключ на полях IDPostIndex, IDTypeOfActivity и IDFormat для соблюдения целостности данных.

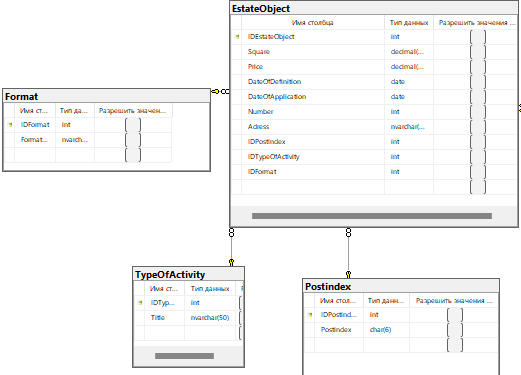


Рисунок 2.3 Сущности EstateObject (Объект недвижимости), Format (Формат недвижимости), TypeOfActivity (Тип объекта), Postindex (почтовый индекс)

Полная диаграмма спроектированной базы данных находится в Приложении 2.

2.1.3 Физическая модель базы данных в СУБД PostgreSQL

Сущности создавались посредством визуальных инструментов - через  
создание сущностей базы данных в приложении DBeaver. Для текстовых полей использовались типы данных varchar (информация нефиксированной длины с использованием кодировки UNICODE) и char (информация фиксированной длины). Для числовых полей - integer (ID сущностей) и decimal (цена, площадь). К таким датам как день рождения и дата применялись типы данных date и timestamp. К полям, предполагающим значения логического типа, применялся тип bool (1 - активен, 0 - неактивен). На диаграмме можно увидеть ограничения первичного ключа, внешнего и запрет на ввод пустых значений. Ограничение первичного ключа необходимо для идентификации записей, а ограничение внешнего для обеспечения целостности данных.

Сущность «Employee» (рис 2.4) хранит всю информацию о работниках компании: ФИО, Email, дата рождения, номер телефона, паспортные данные, логин, пароль, должность, пол. Также была создана сущность «estate\_photo», дающая возможность архитекторам прикреплять фотографии планировок для объектов. Для сущности «Employee» было создано проверочное ограничение на поле IDEmloyee для соблюдения уникальности и целостности данных, создан внешний ключ на полях IDGender и IDPost для соблюдения целостности данных.

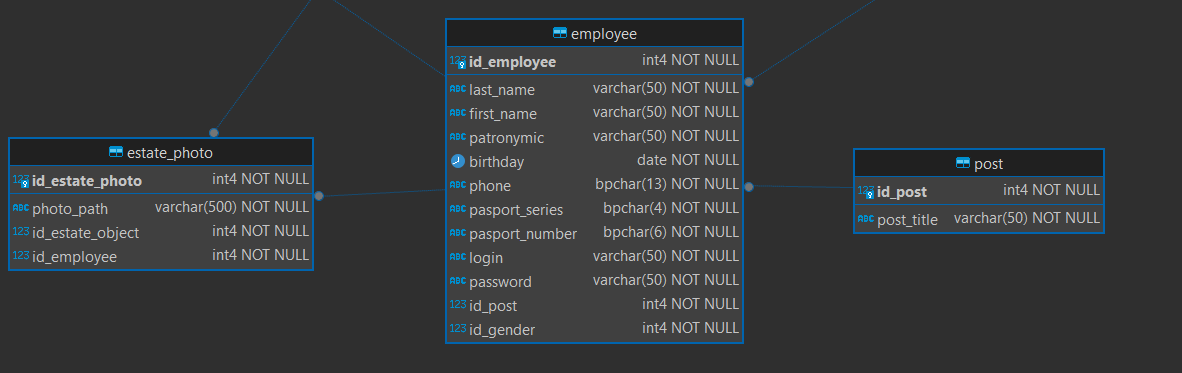


Рисунок 2.4 Сущности employee (Работник), estate\_photo (фотографии планировок), post (должность)

Сущность «Check» (Рис. 2.5) хранит всю основную информацию о сделках с недвижимостью: дата совершения сделки, полная стоимость сделки (будет формироваться автоматически), ID Работника, проводившего сделку, ID Клиента, совершающего сделку, ID объекта недвижимости. Для сущности «Check» было создано проверочное ограничение на поле IDCheck для соблюдения уникальности и целостности данных, создан внешний ключ на полях IDEmployee, IDClient и IDEstateObject для соблюдения целостности данных.

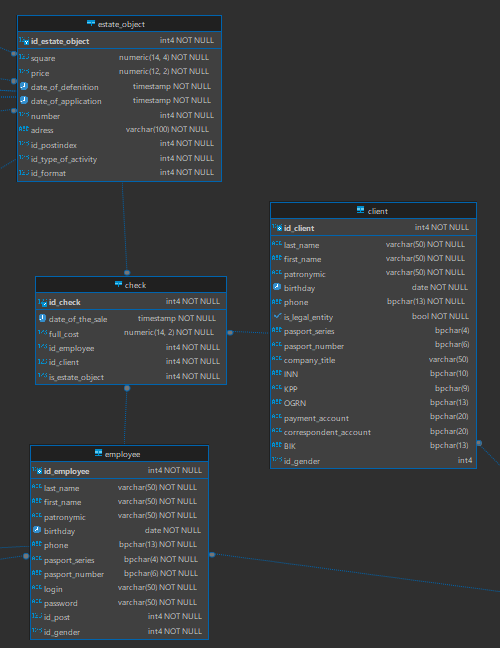


Рисунок 2.5 Сущности сheck (сделка), employee (Работник), client (клиент), estate\_object (объект недвижимости)

Полная диаграмма спроектированной базы данных находится в Приложении 3.

2.2 Создание объектов автоматизации базы данных в СУБД Microsoft SQL Server

2.2.1 Разработка представлений

Представление – виртуальная таблица, представляющая собой поименованный запрос, который будет представлен как подзапрос при использовании представления. В отличие от обычных таблиц реляционных баз данных, представление не является самостоятельной частью набора данных, хранящегося в базе.

Было разработано 3 представления. Первое - VW\_Places, позволяет быстро получить всю возможную информацию о всех объектах недвижимости, типа – участок с указанием всех идентификаторов строений, стоящих на этом участке. Код создания и результат выполнения представлен на рисунке 2.6.



Рисунок 2.6 Создание представления VW\_Places

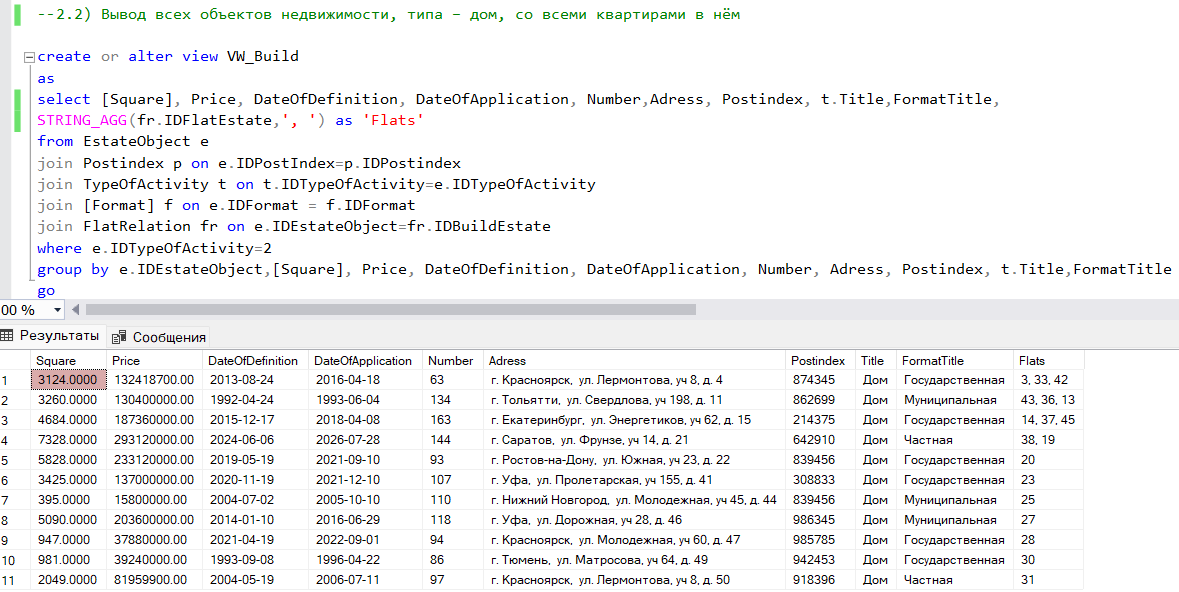
Второе - VW\_Build, позволяет быстро получить всю возможную информацию о всех объектах недвижимости с указанием всех идентификаторов квартир, находящихся в этом доме, типа – строение. Код создания и результат выполнения представлен на рисунке 2.7.

Рисунок 2.7 Создание представления VW\_ Build

Третье - VW\_Flat, позволяет быстро получить всю возможную информацию о всех объектах недвижимости, типа – квартира. Код создания и результат выполнения представлен на рисунке 2.8.

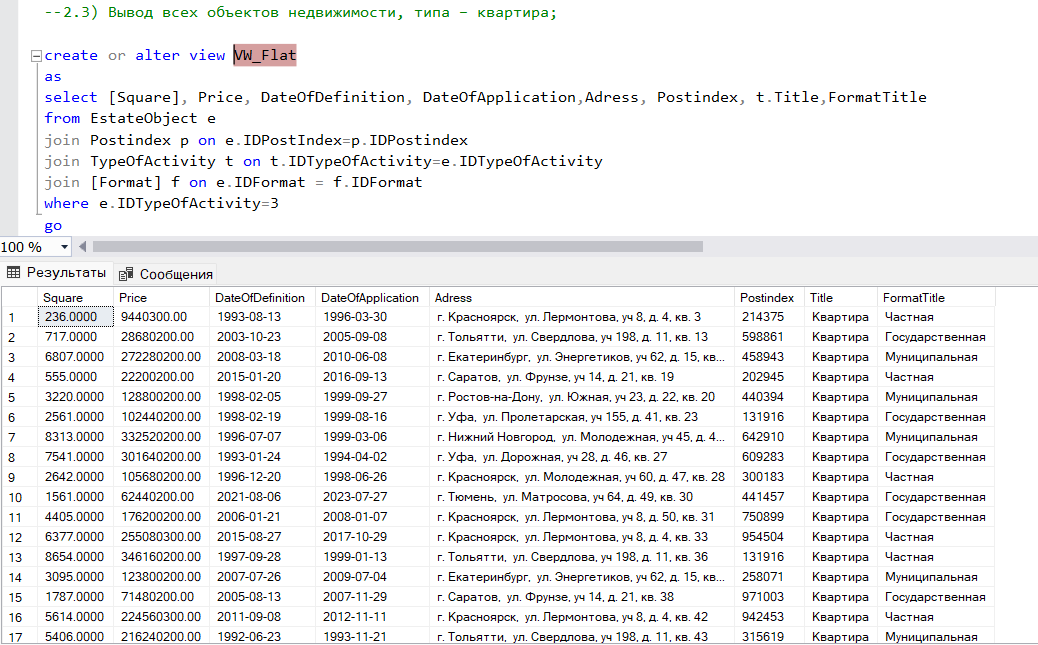


Рисунок 2.8 Создание представления VW\_Flat

2.2.2 Разработка пользовательских функций

Пользовательская функция в SQL — это набор инструкций, которые выполняют определенную операцию и могут быть вызваны из других запросов или скриптов SQL. Они позволяют повторно использовать код и упрощают работу с базой данных, так как позволяют абстрагировать сложную логику в отдельные функции. Пользовательские функции могут принимать параметры, возвращать значения и выполнять различные операции на данных.

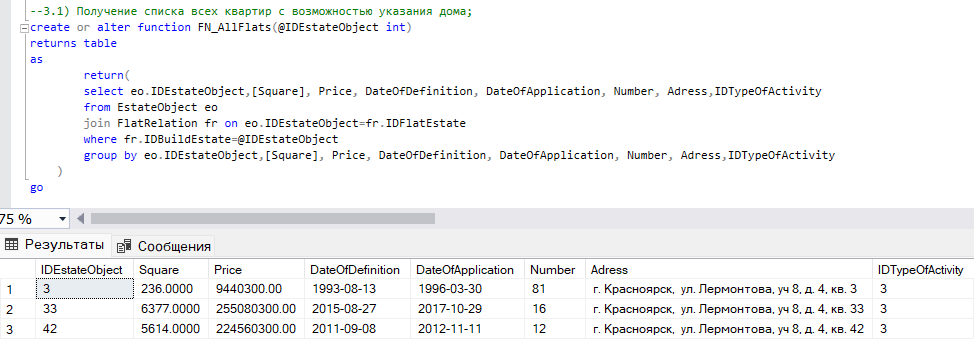
Было разработано 3 функции. Первая - FN\_AllFlats в качестве входного параметра принимает ID дома, возвращает список квартир в этом строении. Код создания и результат выполнения представлен на рисунке 2.9.

Рисунок 2.9 Создание функции FN\_AllFlats

Вторая - FN\_AllBuilds, в качестве входного параметра принимает ID участка, возвращает список домов на этом участке. Код создания и результат выполнения представлен на рисунке 2.10.

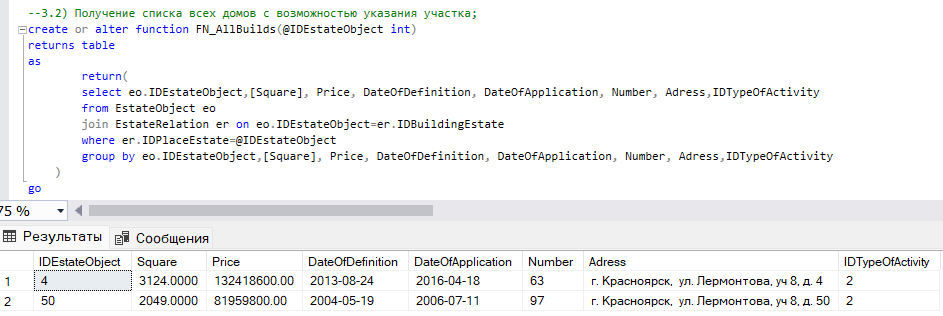


Рисунок 2.10 Создание функции FN\_AllBuilds

Третья – FN\_HistoryCheck, позволяет получить список истории продаж определённого объекта недвижимости, функция принимает в себя значение идентификатора объекта недвижимости и возвращает его полную историю продаж, актуальную цену и владельца . Код создания и результат выполнения представлен на рисунке 2.11.

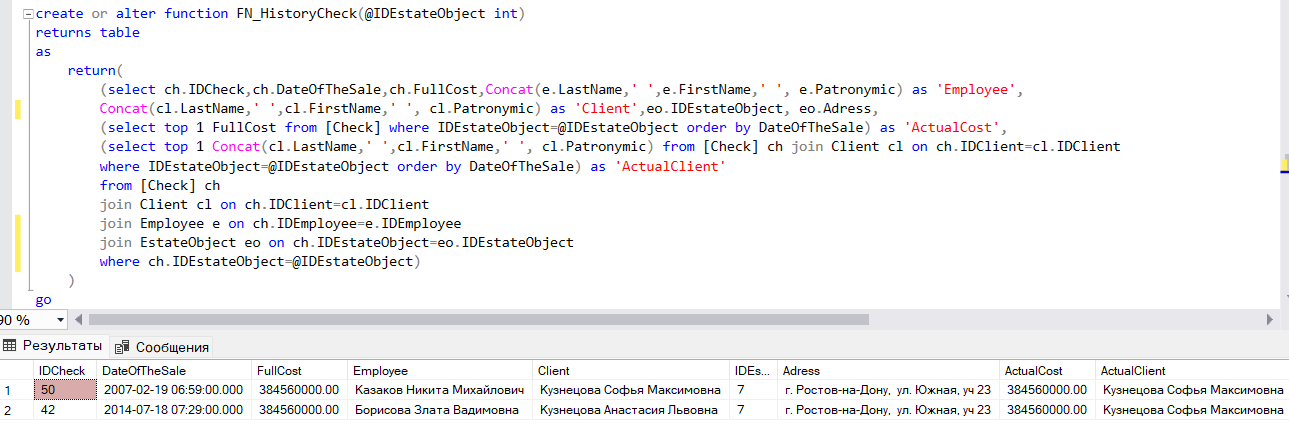


Рисунок 2.11 Создание функции FN\_HistoryCheck

2.2.3 Разработка хранимых процедур

Хранимая процедура в SQL — это предварительно скомпилированный набор инструкций, который хранится в базе данных и может быть вызван пользователем для выполнения определенной операции или набора операций. Они упрощают и оптимизируют работу с базой данных, позволяют повторно использовать код.

Было разработано 2 процедуры. Первая - PR\_AddCommonCost, позволяет повышать или понижать цены домов, находящихся на одном общем участке на заданную стоимость. Процедура принимает в себя 2 значения, первое – идентификатор участка, второе – сумма, на которую нужно повысить или понизить цену. Код создания и результат выполнения представлен на рисунке 2.12.

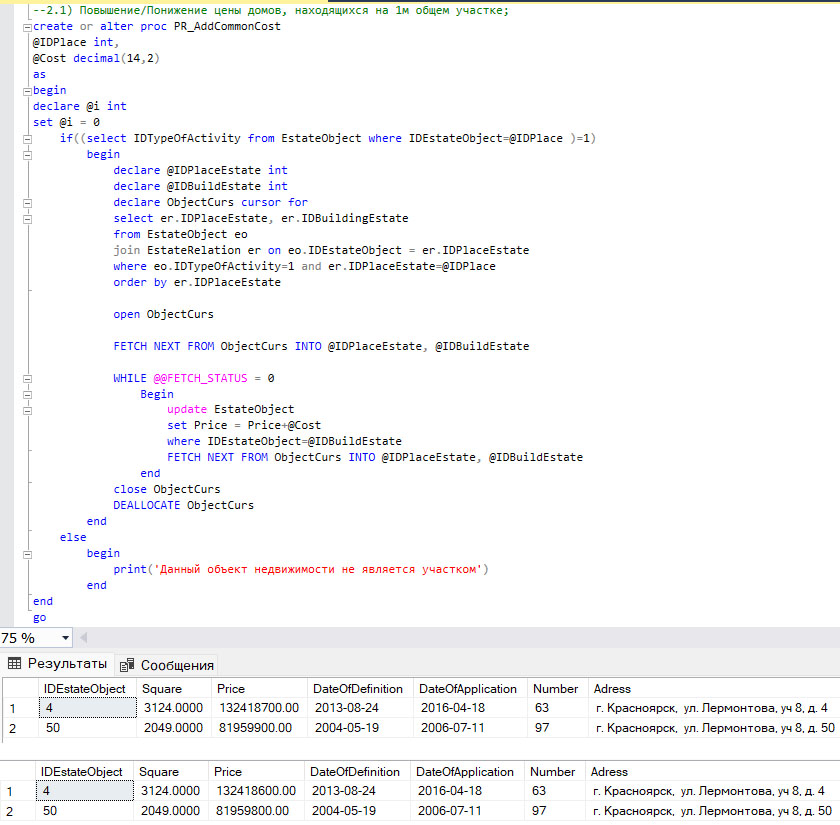


Рисунок 2.12 Создание процедуры PR\_AddCommonCost

Второе - PR\_AddCommonFlatCost, позволяет повышать или понижать цены квартир, находящихся в одном общем строении. Процедура принимает в себя 2 значения, первое – идентификатор строения, второе – сумма, на которую нужно повысить или понизить цену. Код создания и результат выполнения представлен на рисунке 2.13.

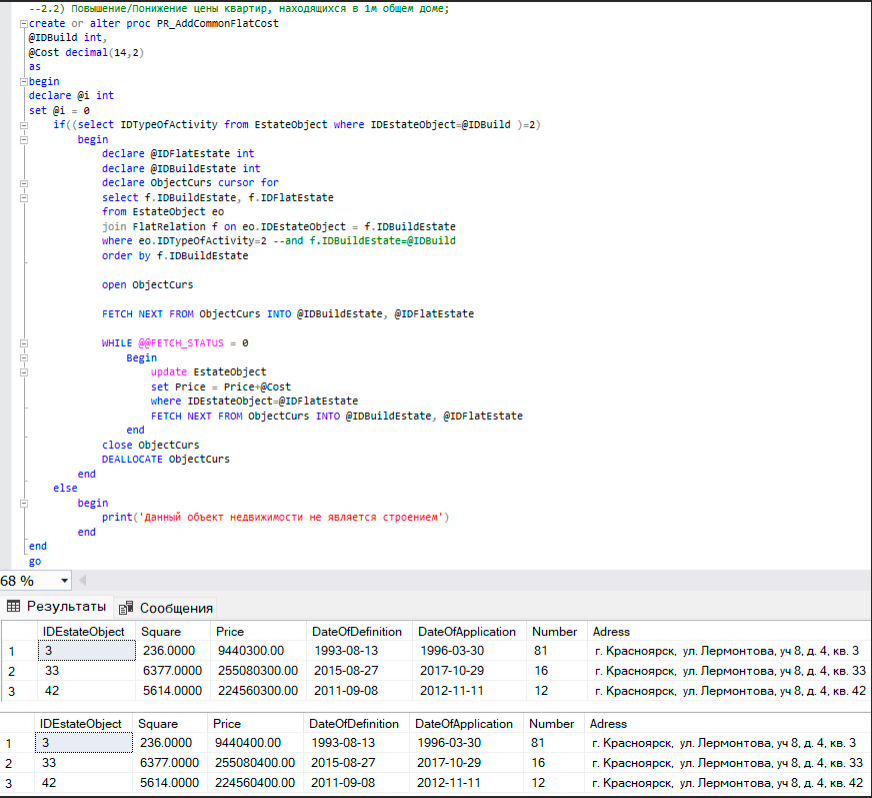


Рисунок 2.13 Создание процедуры PR\_AddCommonFlatCost

2.2.4 Разработка триггеров

Триггер в SQL — это специальный тип хранимого объекта, который автоматически выполняется при определенных событиях, происходящих в базе данных, таких как вставка, обновление или удаление данных из таблицы.

Было разработано 4 триггера. Первый - TR\_DifferencePlaceBlock, запрещает добавление записи в таблицу EstateRelation в случае несоответствия объекта (например, нельзя добавить к объекту типа «участок» объект с типом «квартира» или «участок»). Код создания и результат выполнения представлен на рисунке 2.14.

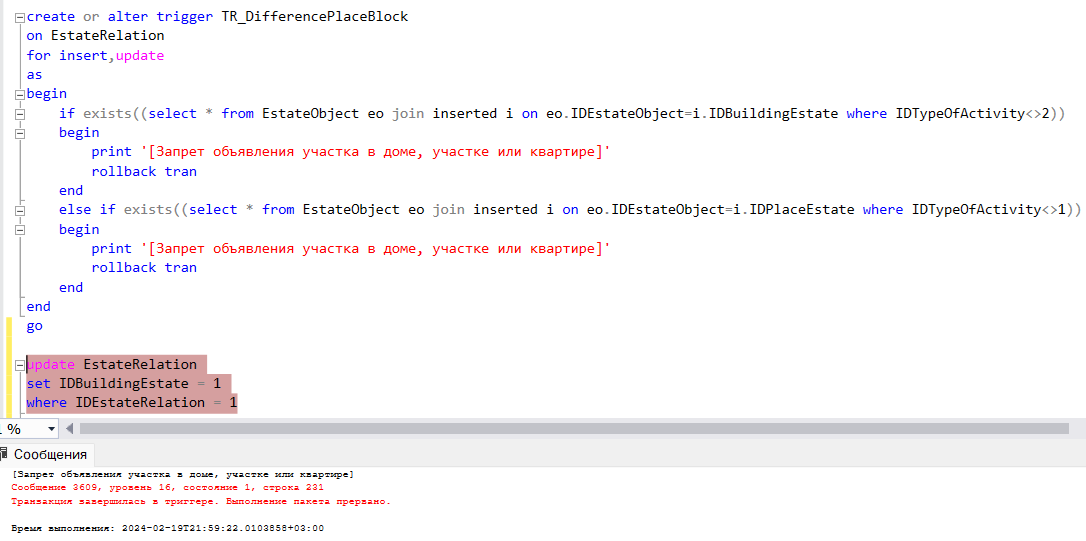


Рисунок 2.14 Создание триггера TR\_DifferencePlaceBlock

Второй - TR\_DifferenceFlatBlock, запрещает добавление записи в таблицу FlatRelation в случае несоответствия объекта (например, нельзя добавить к объекту типа «квартира» объект с типом «участок» или «квартира»). Код создания и результат выполнения представлен на рисунке 2.15.

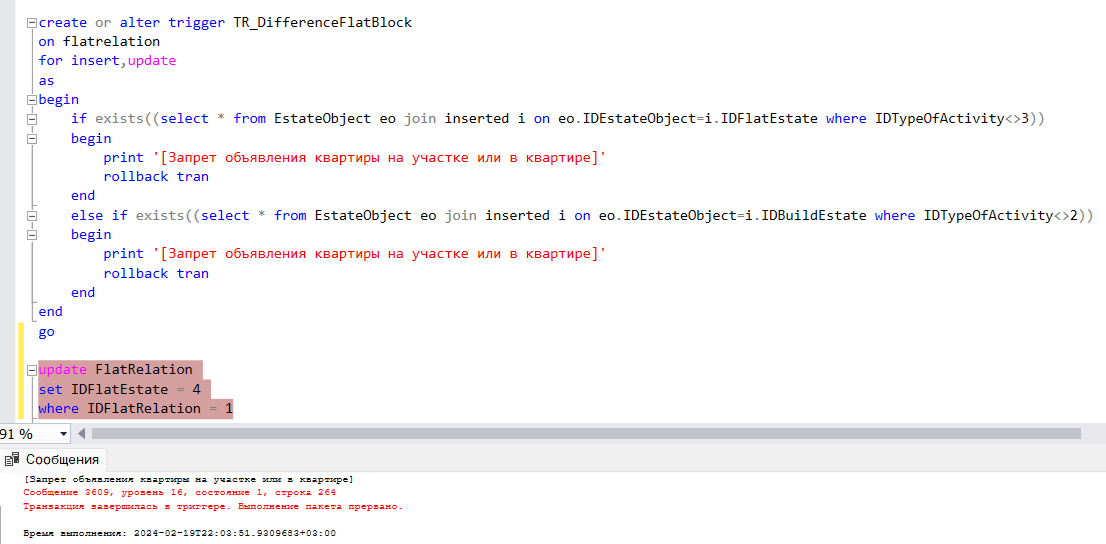


Рисунок 2.15 Создание триггера TR\_DifferenceFlatBlock

Третий - TR\_IsLegalEntity, запрещает добавлять данные, предназначенные для физического лица в столбцы для юридического лица, и наоборот. Код создания и результат выполнения представлен на рисунке 2.16.

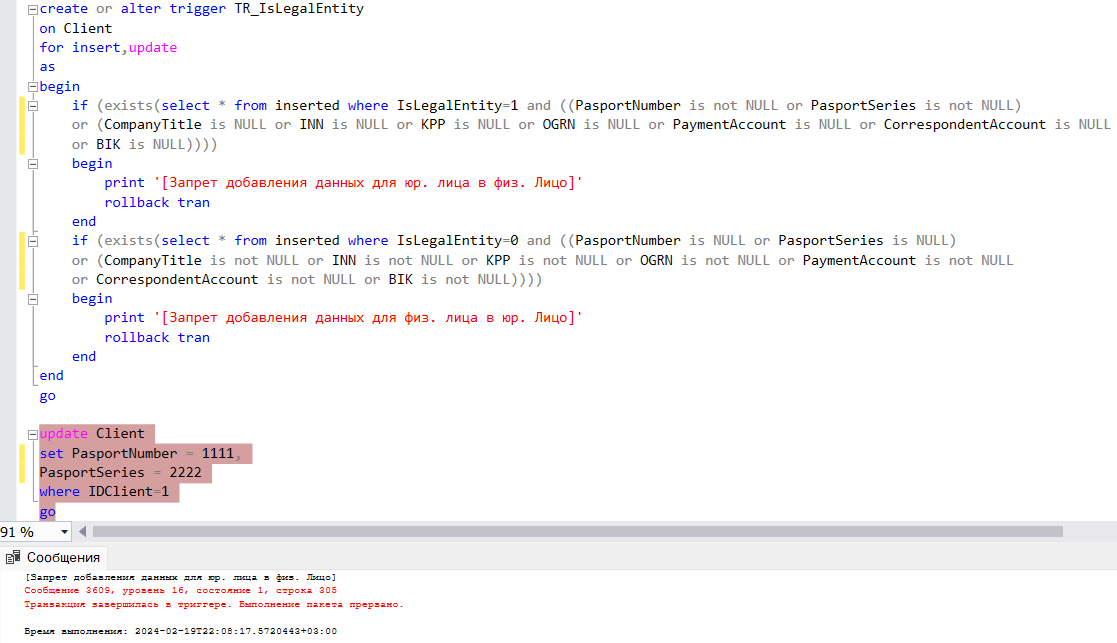


Рисунок 2.16 Создание триггера TR\_IsLegalEntity

Четвёртый - TR\_AutoFullCost, автоматически подставляет цену из стоимости объекта, в случае если это поле оставить пустым. Код создания и результат выполнения представлен на рисунке 2.17.

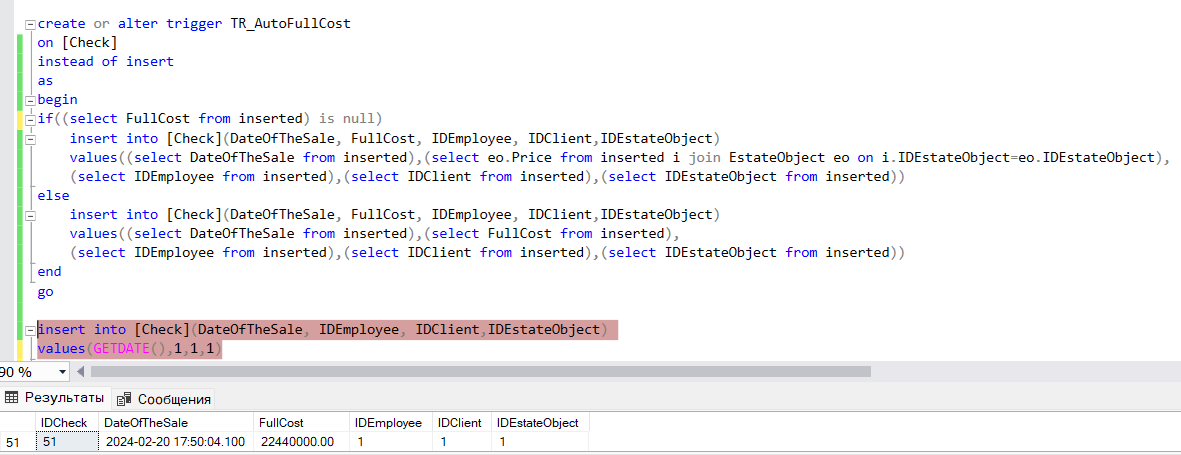


Рисунок 2.17 Создание триггера TR\_AutoFullCost

2.3 Создание объектов автоматизации базы данных в СУБД PostgreSQL

2.3.1 Разработка представлений

Отличие представлений в СУБД PostgreSQL лишь в том, что наименование столбцов пишется через двойную кавычку, с использованием правила snake\_case.

Аналогичное представление «vw\_places». Код создания и результат выполнения представлен на рисунке 2.18.

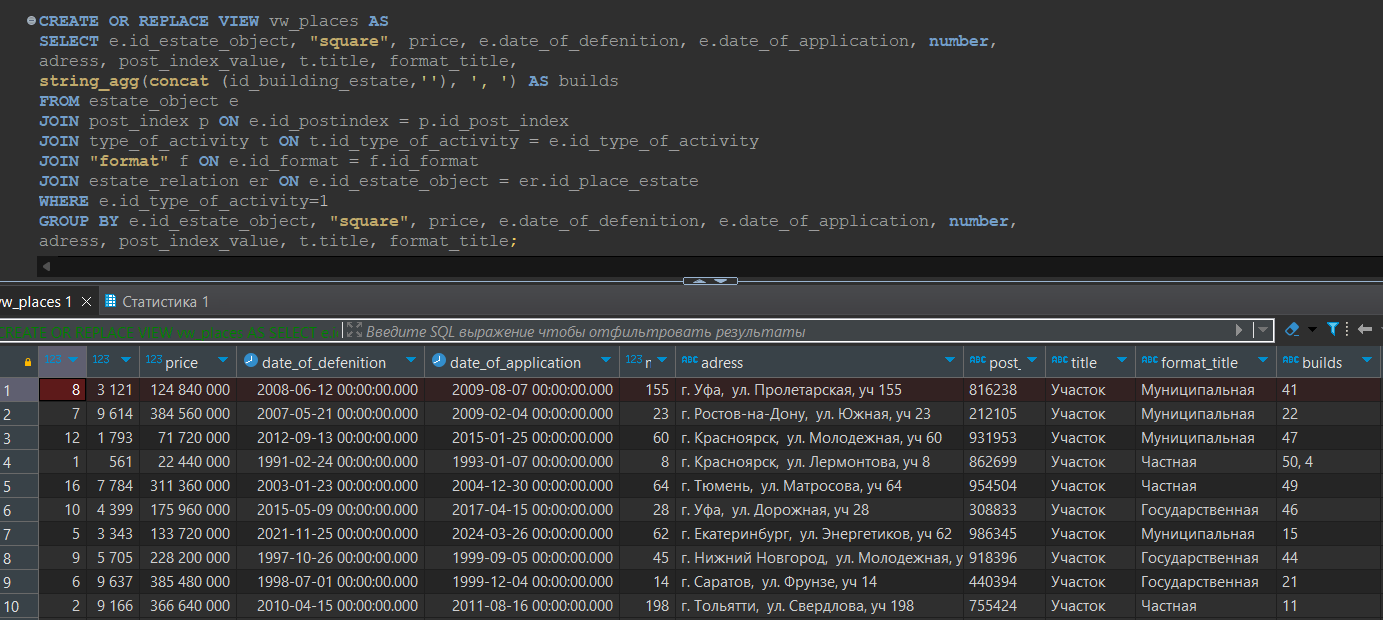


Рисунок 2.18 Создание представления vw\_places

Аналогичное представление «vw\_builds». Код создания и результат выполнения представлен на рисунке 2.19.

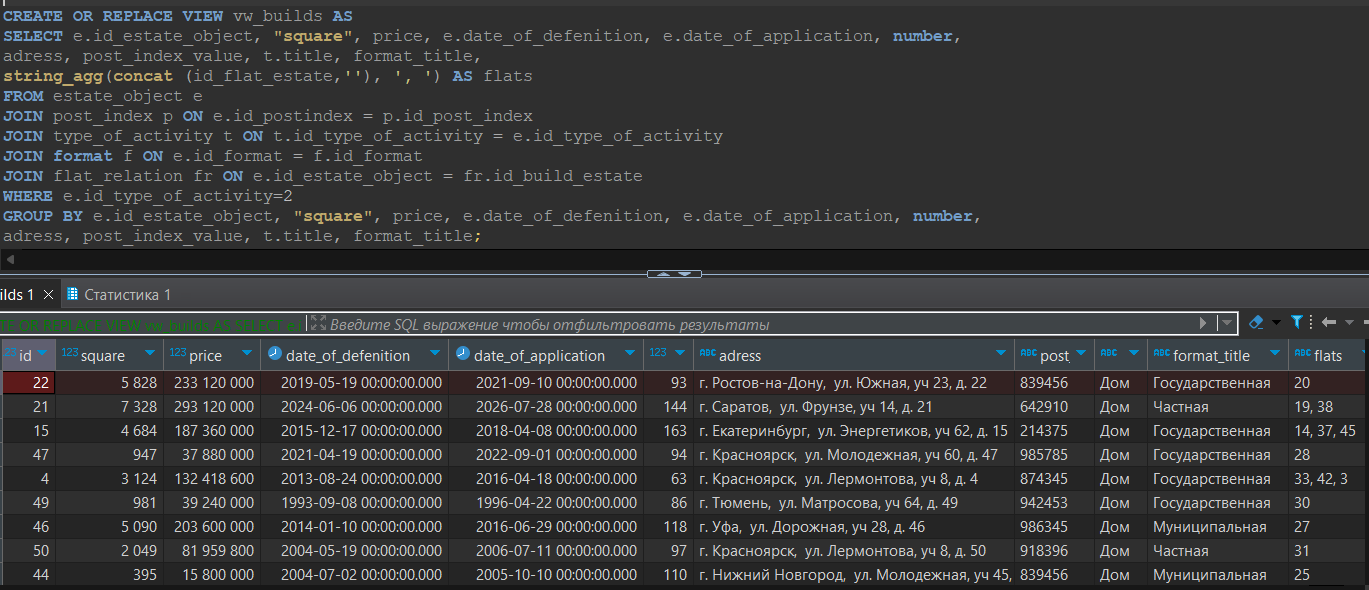


Рисунок 2.19 Создание представления vw\_builds

Аналогичное представление «vw\_flats». Код создания и результат выполнения представлен на рисунке 2.20.

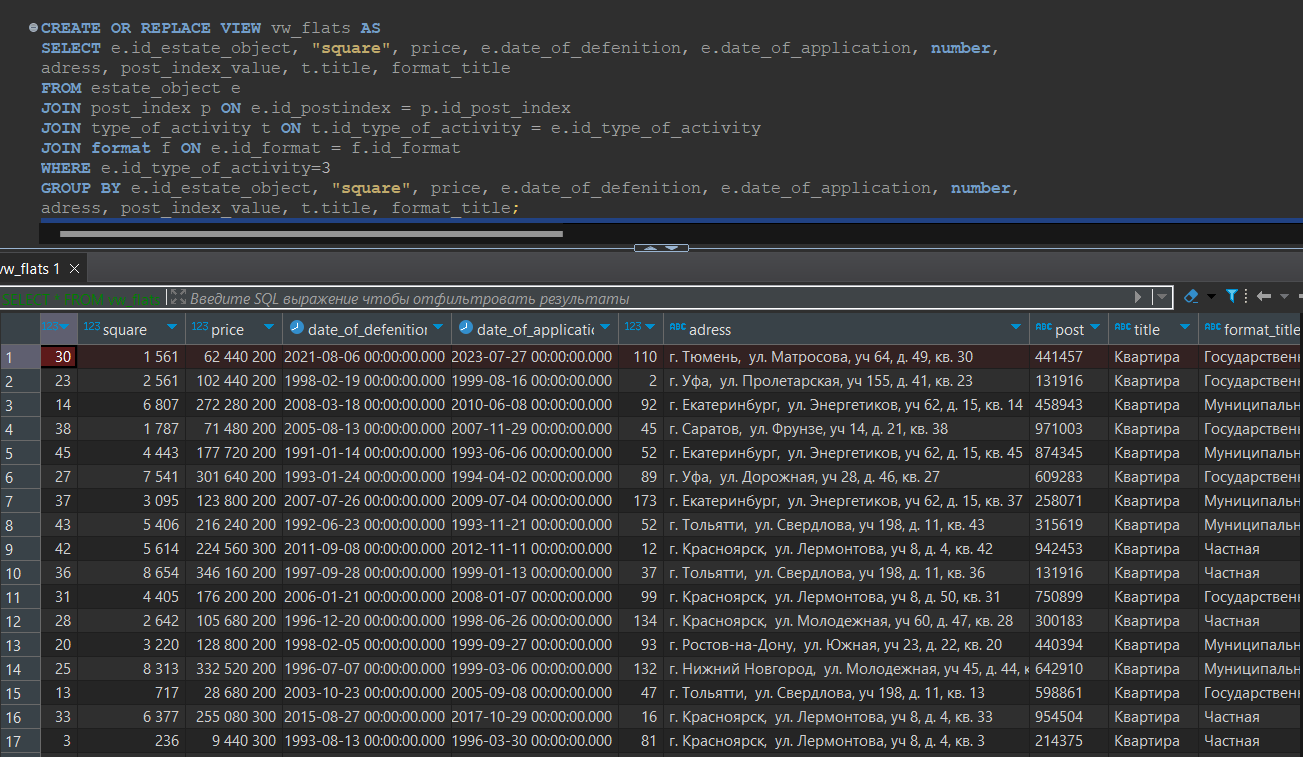


Рисунок 2.20 Создание представления vw\_flats

2.3.2 Разработка пользовательских функций

Отличительной особенностью функций в СУБД PostgreSQL является более сложный синтаксис, в табличных функциях необходимо прописывать столбцы возвращаемой таблицы. Аналогичная функция «fn\_all\_flats». Код создания и результат выполнения представлен на рисунке 2.21.

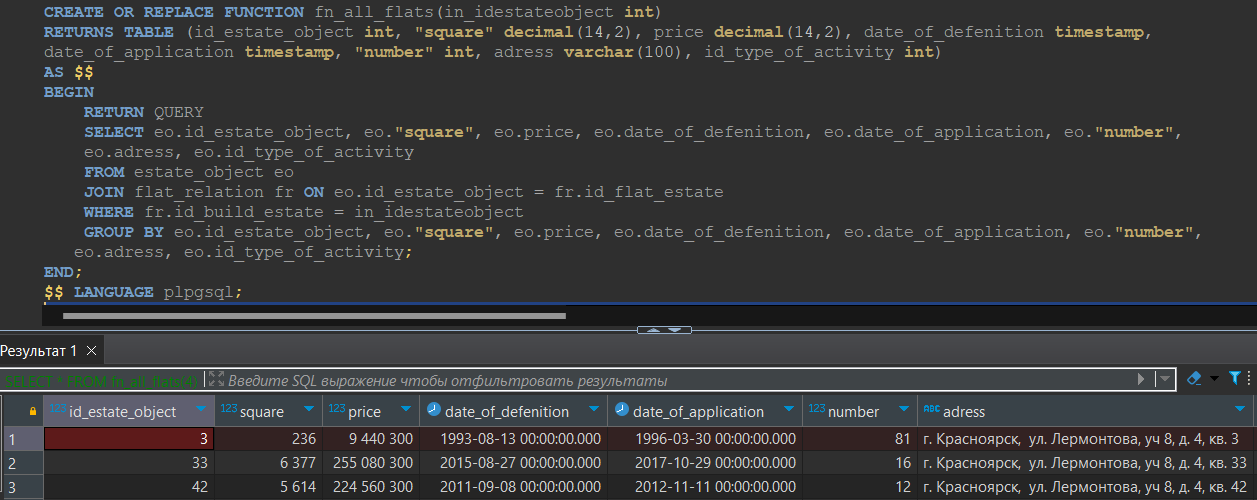


Рисунок 2.21 Создание функции fn\_all\_flats

Аналогичная функция «fn\_all\_builds». Код создания и результат выполнения представлен на рисунке 2.22.

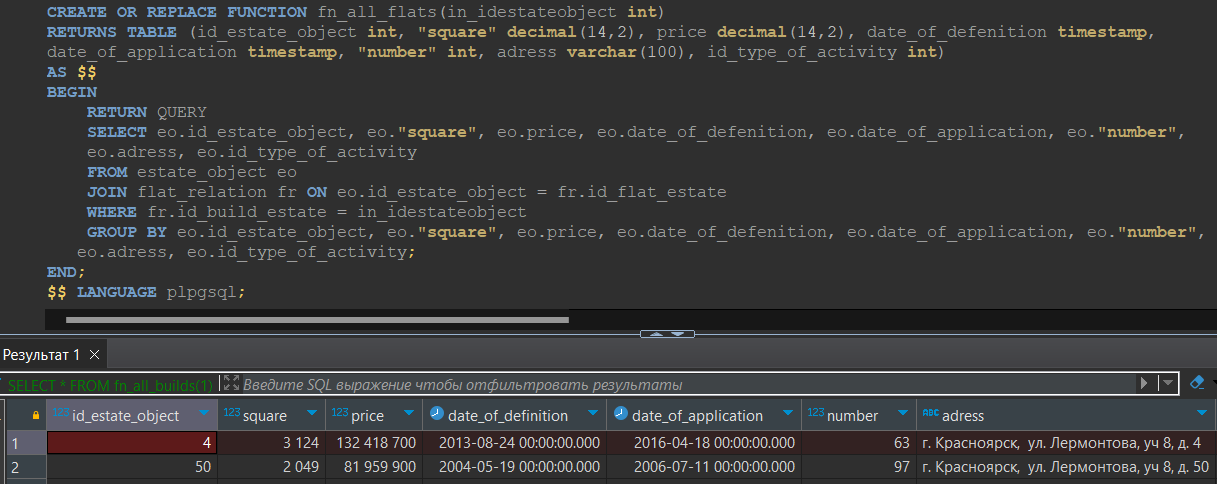


Рисунок 2.22 Создание функции fn\_all\_builds

Аналогичная функция «fn\_history\_check». Код создания и результат выполнения представлен на рисунке 2.23.

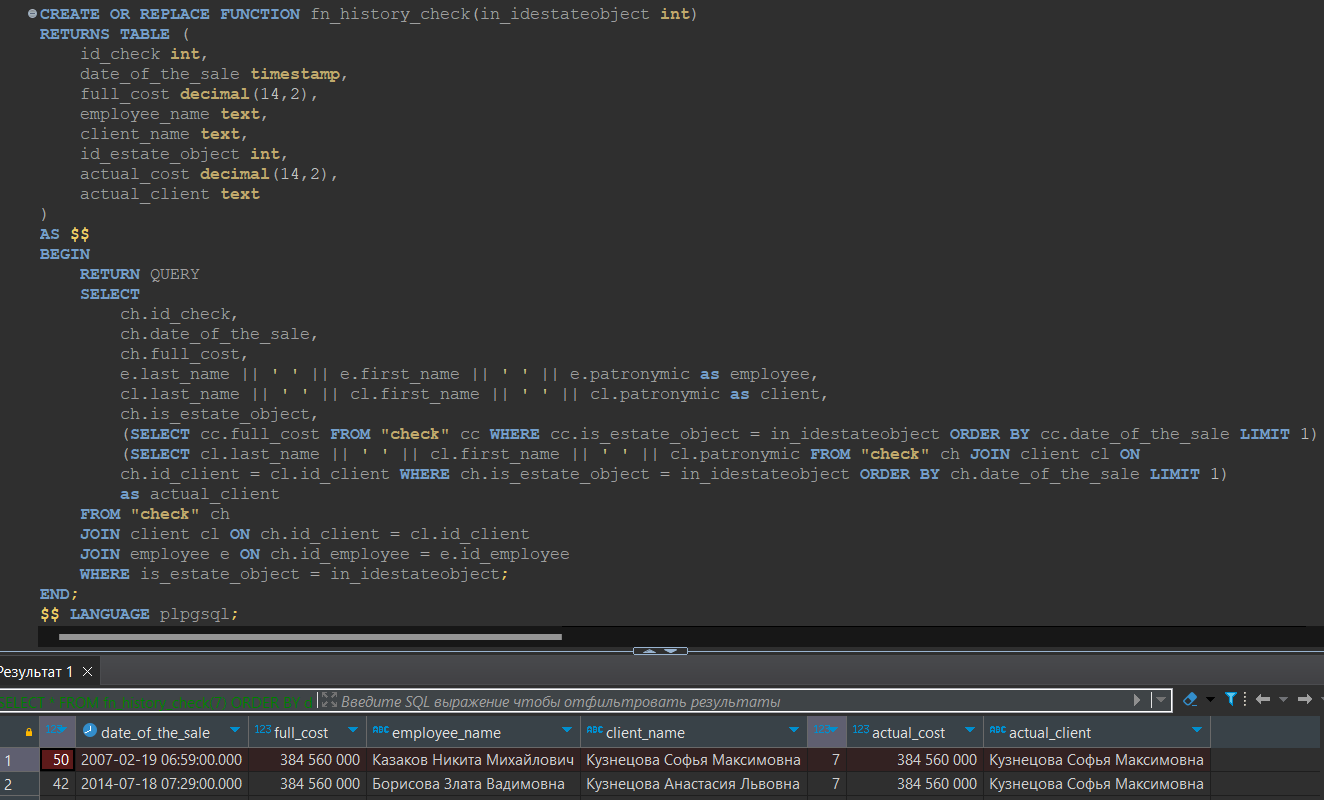


Рисунок 2.23 Создание функции fn\_history\_check

2.3.3 Разработка хранимых процедур

Синтаксис создания процедур схож с синтаксисом Microsoft SQL Server.

Аналогичная процедура «pr\_add\_common\_cost», однако синтаксис создания и использования курсора, с помощью которого выполнена процедура, отличается. Код создания и результат выполнения представлен на рисунке 2.24.

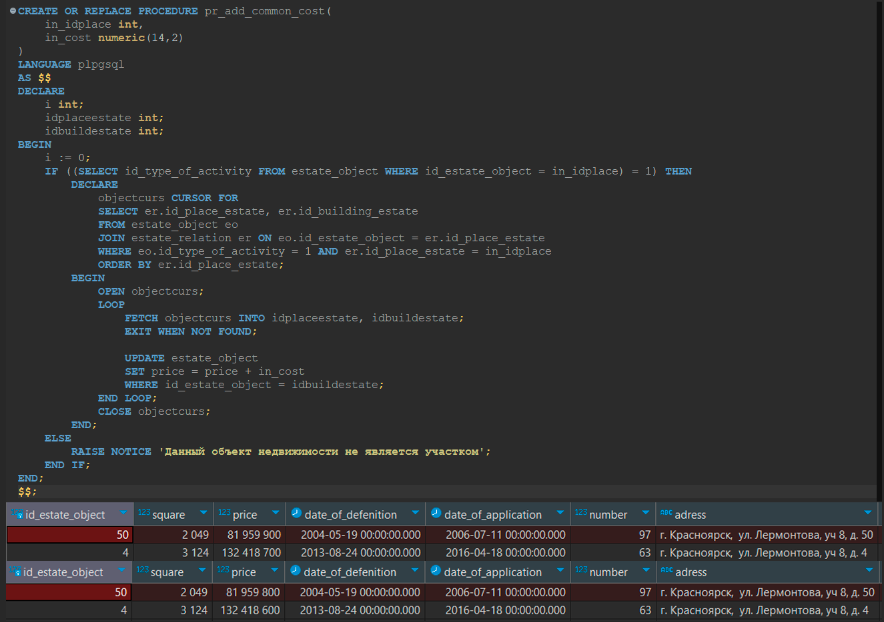


Рисунок 2.24 Создание процедуры pr\_add\_common\_cost

Аналогичная процедура «pr\_add\_common\_flat\_cost», однако синтаксис использования цикла, с помощью которого выполнена процедура, отличается. Код создания и результат выполнения представлен на рисунке 2.25.

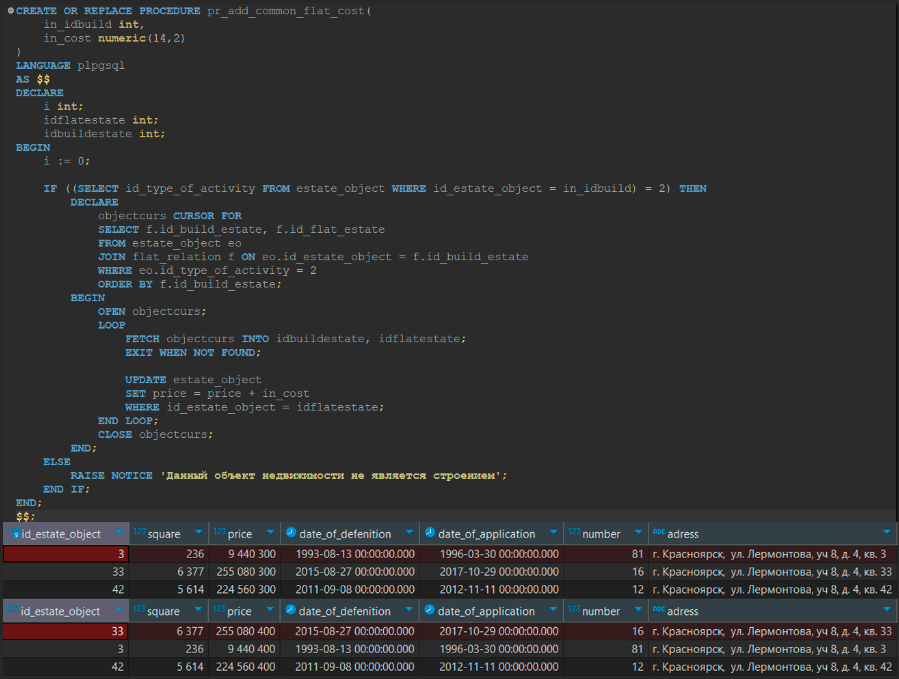


Рисунок 2.25 Создание процедуры pr\_add\_common\_flat\_cost

2.3.4 Разработка триггеров

В PostgreSQL создание триггеров сильно отличается от Microsoft SQL Server, здесь триггеры реализуются через функции, и исполняют именно их, однако сам функционал мало чем отличается.

Аналогичный триггер «tr\_difference\_place\_block», Код создания и результат выполнения представлен на рисунке 2.26.

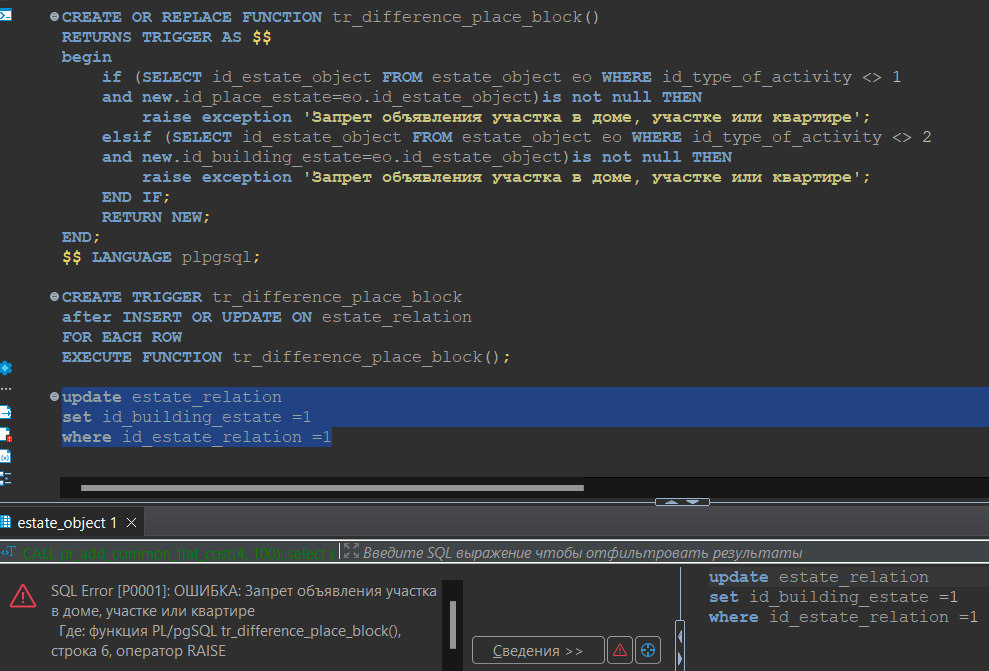


Рисунок 2.26 Создание триггера tr\_difference\_place\_block

Аналогичный триггер «tr\_difference\_place\_block». Код создания и результат выполнения представлен на рисунке 2.27.

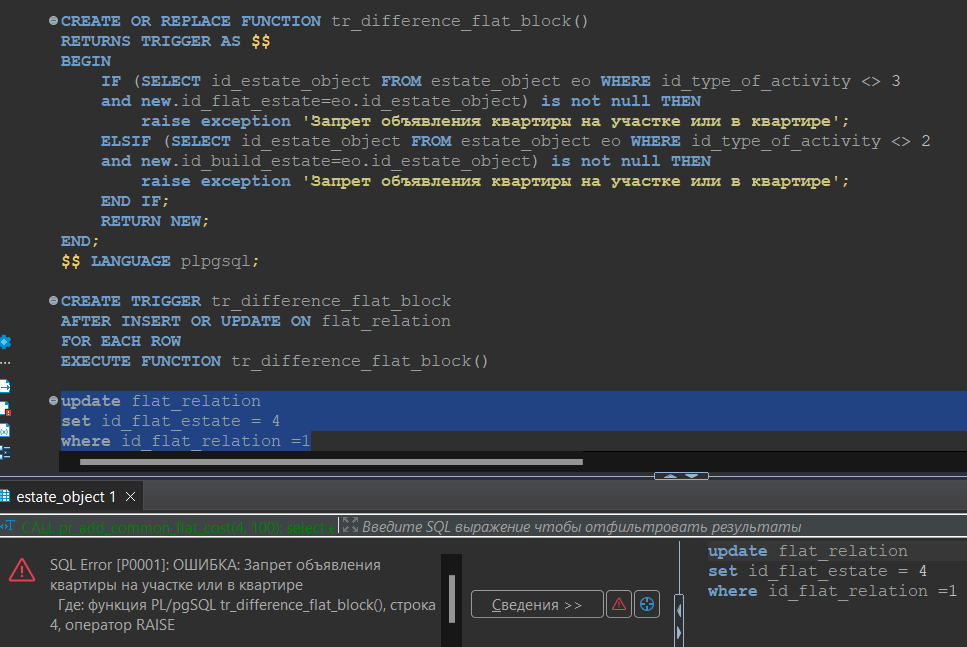


Рисунок 2.27 Создание триггера tr\_difference\_flat\_block

Аналогичный триггер «tr\_is\_legal\_entity». Код создания и результат выполнения представлен на рисунке 2.28.

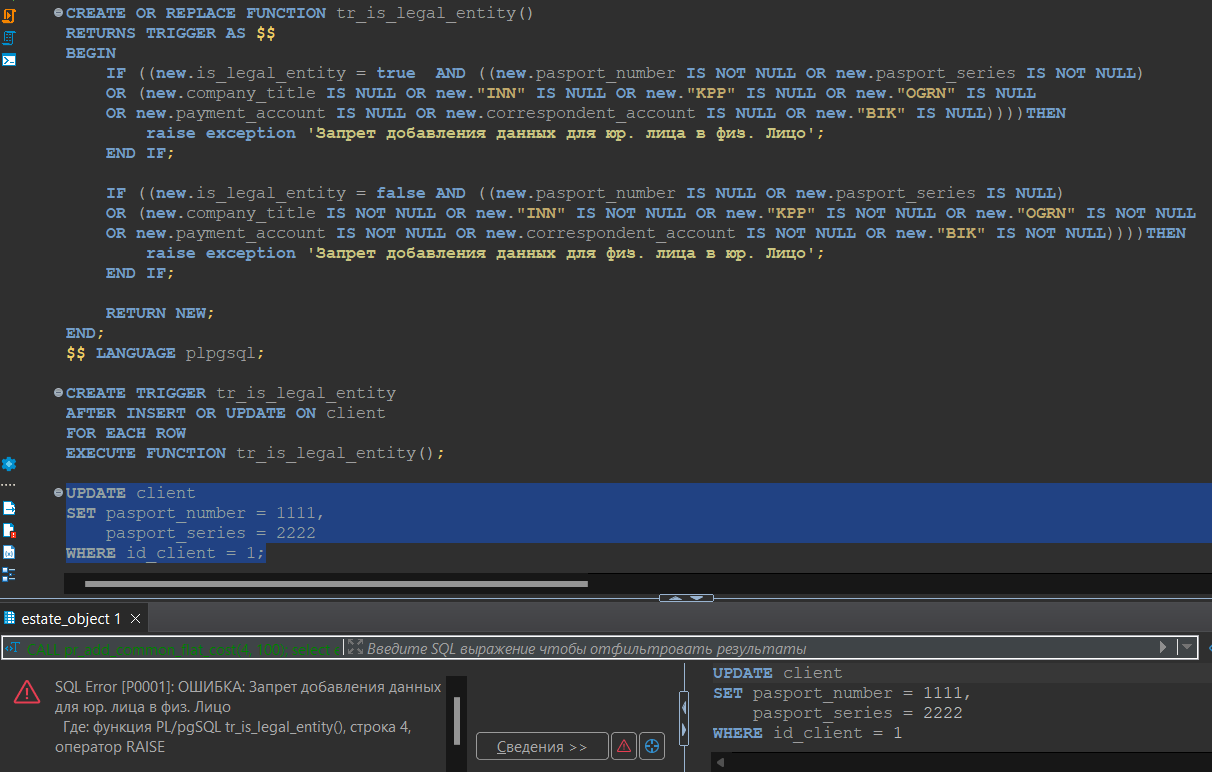


Рисунок 2.28 Создание триггера tr\_is\_legal\_entity

Триггер «tr\_auto\_full\_cost» был создан иначе чем в Microsoft SQL Server, так как в PostgreSQL нельзя использовать инструкцию INSTEAD OF, пришлось воспользоваться BEFORE, таким образом код оказался более оптимальным. Код создания и результат выполнения представлен на рисунке 2.29.

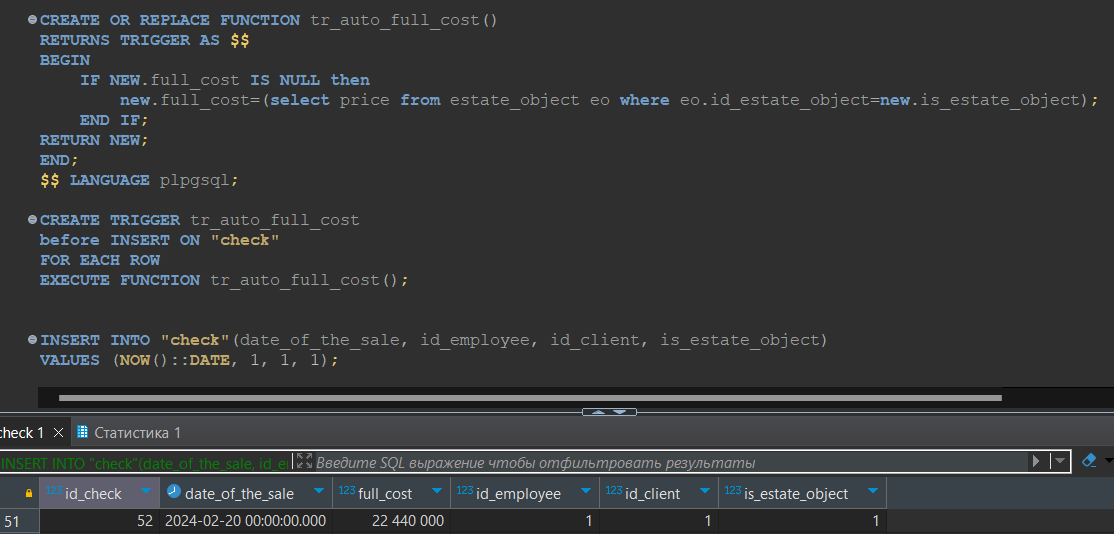


Рисунок 2.29 Создание триггера tr\_auto\_full\_cost

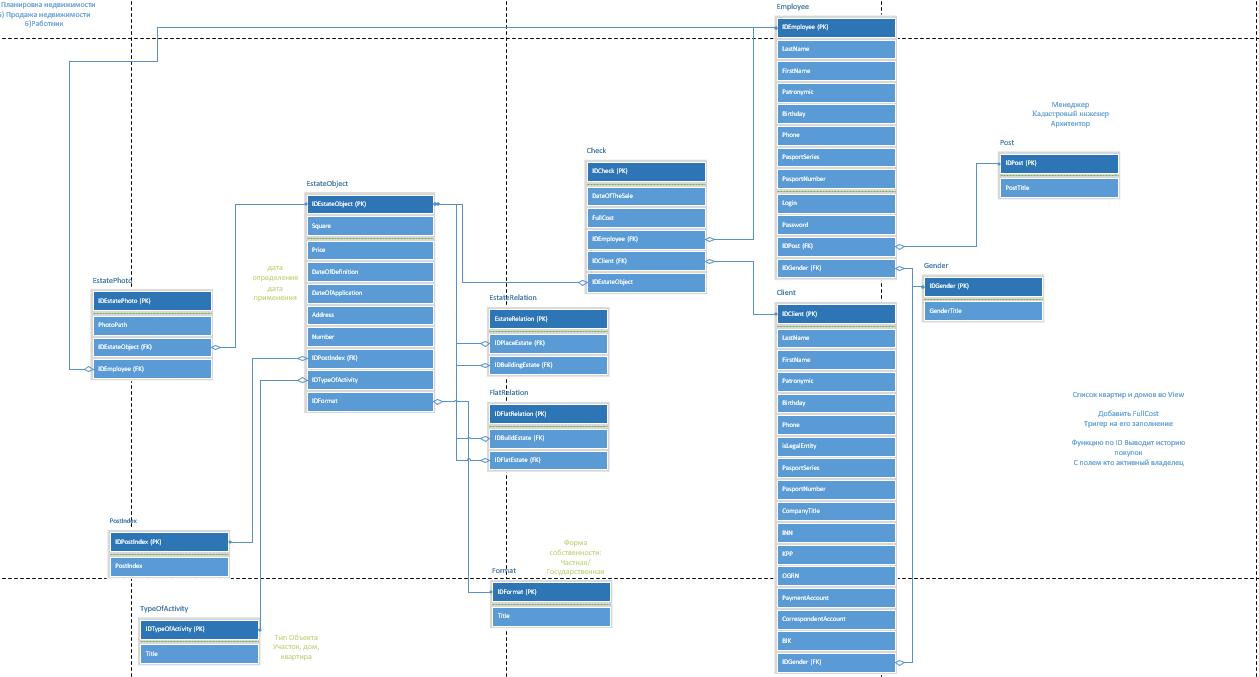
**ЗАКЛЮЧЕНИЕ**

Таким образом, проектирование и разработка базы данных для учета планируемых и реализованных объектов недвижимости позволяет эффективно управлять данными, сокращать временные и финансовые затраты на процессы учета и анализа информации. Благодаря использованию специализированных инструментов и технологий мы успешно достигли поставленных целей и создали функциональные и надёжные базы данных, способные эффективно поддерживать процессы учета и анализа информации об объектах недвижимости. Новая система учета поможет оптимизировать работу с данными, улучшить качество принимаемых решений и увеличить эффективность деятельности в сфере недвижимости. Объекты баз данных помогут избежать попадания не корректной информации в таблицы, что способствует безопасности и целостности данных. Также объекты помогут быстро и эффективно производить выборку нужных данных, кроме того, помогут проводить администрирование таблиц. Разработка базы данных, а также объектов в Microsoft SQL Server позволила системе быть быстрой и эффективной, однако более медленная база данных на PostgreSQL позволила более удобно реализовать необходимые объекты. При разработке и реализации баз данных были получены теоретические и практические знания и умения в сфере создания баз данных и их объектов, также были получены знания по городскому развитию и учёту объектов недвижимости. Все поставленные цели были достигнуты, однако я считаю, что данную систему можно доработать, чтобы её можно было ввести в настоящую эксплуатацию.

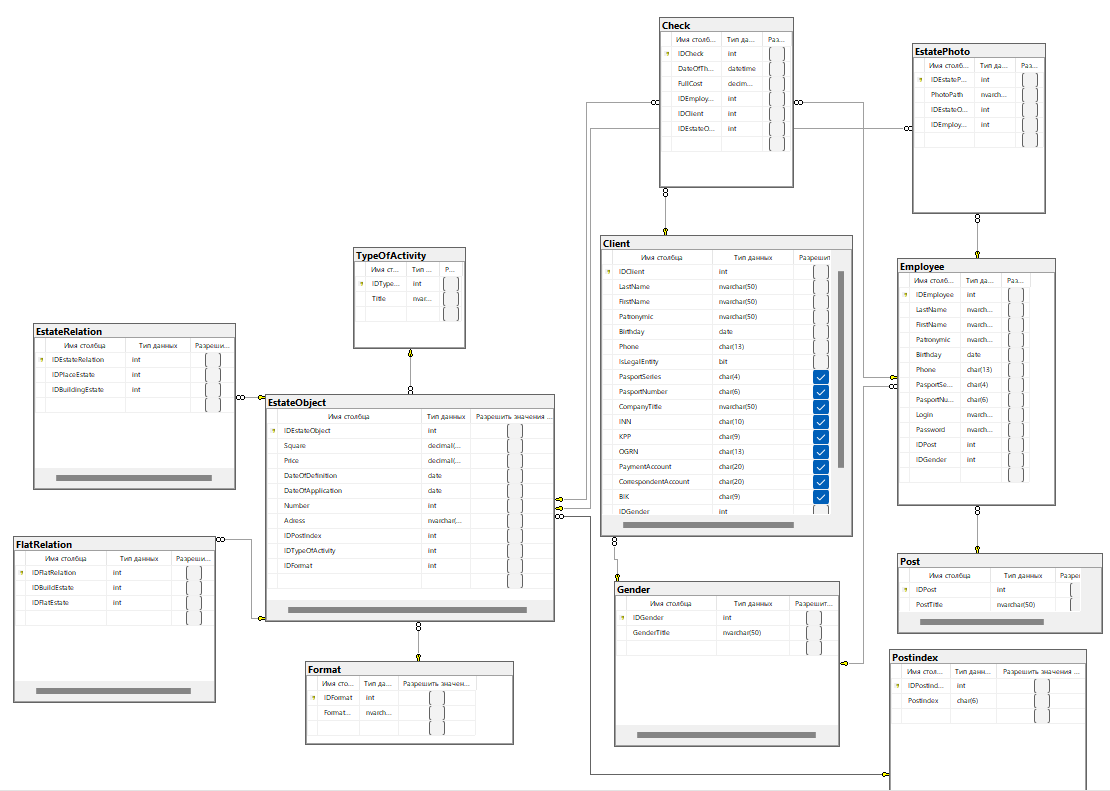
**СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ**

1. Федоров Г.Н., Разработка, администрирование и защита баз данных: учебник. Федорова Г.Н., Академия, 2021 г;
2. Волк В.К., Базы данных. Проектирование, программирование, управление и администрирование. Волк В.К., Лань, 2021 г;
3. Карпова Т. С., Базы данных: модели, разработка, реализация: учебное пособие. Карпова Т. С., ИНТУИТ, 2022 г;  
   Электронные ресурсы
4. Росреестр / Публичная кадастровая карта: <https://pkk.rosreestr.ru/#/search/55.75158999311336,37.61697544168833/19/@5w3tr1rm2> (Дата обращения 10.02.2024)
5. Росреестр / Кадастровый учёт и регистрация прав: <https://rosreestr.gov.ru/activity/okazanie-gosudarstvennykh-uslug/kadastrovyy-uchet-i-ili-registratsiya-prav-/> (Дата обращения 10.02.2024)
6. Иннотер / Статья «Городское планирование, цифровой двойник города» <https://innoter.com/articles/gorodskoe-planirovanie> (Дата обращения 10.02.2024)
7. КодНет / Статья «Объекты и концепции базы данных» <http://www.codenet.ru/db/interbase/ibsql/objs.php> (Дата обращения 10.02.2024)

Приложение 1. Диаграмма базы данных в Microsoft Visio



Приложение 2. Диаграмма базы данных в Microsoft SQL Server



Приложение 3. Диаграмма базы данных в PostgreSQL

