МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ ДНР

ГОУ ВПО «ДОНЕЦКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет КНТ

Кафедра ПИ

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту по дисциплине

«Программирование систем с Базами Данных»

по теме: Информационная система «Нефтеперерабатывающие заводы»

|  |  |
| --- | --- |
| Руководители:  асс. каф. ПИ  Щедрин С.В.  асс. каф. ПИ  Ногтев Е.А. | Выполнил:  ст. гр. ПИ-17в  Петренко Д.А. |

Донецк – 2020Реферат

Пояснительная записка к курсовому проекту содержит: 77 страниц, 45 рисунков, 6 таблиц, 4 источника, 6 приложений.

Цель работы – закрепить практические навыки разработки реляционных баз данных (БД) и проектирования ПО, научиться осуществлять нормализацию таблиц, получить углубленные знания языка структурированных запросов (SQL), приобрести практические навыки самостоятельной разработки основных компонентов прикладного программного обеспечения.

Для достижения поставленной цели курсового проекта необходимо решить следующие задачи:

* выполнить анализ предметной области задания курсовой работы;
* осуществить проектирование модели базы данных;
* разработать программный продукт;
* осуществить тестирование спроектированного программного комплекса.

Объект исследования – информационная система лечебных заведений.

Результат работы – разработанная СУБД обеспечения и учета заказов нефтеперерабатывающих заводов на языке С#, с использованием синтаксиса PostgreSQL.

БАЗА ДАННЫХ, СУБД, POSTGRESQL, СВЯЗИ, ЗАПРОС, РЕЛЯЦИОННАЯ БАЗА ДАННЫХ

Содержание

[Введение…………………………………………………………………………...4](#_Toc40791292)

[1 описание предметной области, постановка задачи…………………………..6](#_Toc40791293)

[2 Обоснование выбора СУБД, описание возможностей СУБД……………….7](#_Toc40791294)

[3 Обоснование выбора инструментальных средств для написания клиентской части, проектирование структуры ПО……………………………11](#_Toc40791295)

[3.1 Невизуальные компоненты для работы с данным 13](#_Toc40791296)

[3.2 Визуальные компоненты для работы с данными 13](#_Toc40791297)

[3.3 Разработка шаблонов приложений для работы с шаблонами базы данных 15](#_Toc40791298)

[4 Проектирование базы данных в выбранной СУБД…………………………18](#_Toc40791299)

[4.1 Проектирование концептуальной модели БД 18](#_Toc40791300)

[4.2 Создание таблиц, доменов, индексов 19](#_Toc40791301)

[4.3 Разработка триггеров 22](#_Toc40791302)

[4.4 Проектирование запросов к базе данных 26](#_Toc40791303)

[4.5 Создание представлений и хранимых процедур, функций 27](#_Toc40791304)

[5 Разработка клиентского приложения………………………………………..51](#_Toc40791305)

[5.1 Формы и компоненты для работы с основными таблицами 51](#_Toc40791306)

[5.2 Формы и компоненты для работы со справочниками 51](#_Toc40791307)

[5.3 Формы и компоненты для отображения результатов запросов 52](#_Toc40791308)

[6 Тестирование разработанной информационной системы………………….54](#_Toc40791309)

[ЗАКЛЮЧЕНИЕ………………………………………………………………….60](#_Toc40791310)

[Перечень ссылок………………………………………………………………...61](#_Toc40791311)

[Приложение А Техническое задание…………………………………………..62](#_Toc40791312)

[Приложение Б Листниг шаблонов………………………………………….….63](#_Toc40791313)

[Приложение В Листинг серверного приложения…………..…………………68](#_Toc40791314)

[Приложение Г Листинг клиентского приложения……………………………74](#_Toc40791315)

[Приложение Д Руководство пользователя…………………………………….81](#_Toc40791316)

[Приложение Ж Руководство администратора………………………………...82](#_Toc40791317)

Введение

На сегодняшний день применение баз данных приобрело весьма важное значение для многих организаций, которые для упрощения своей работы применяют компьютерные технологии.

База данных (БД) — это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации, преимущественно больших объемов. [Базы данных](https://hostiq.ua/wiki/migrate-mysql-database/) активно используются для проектирования практически любых пользовательских приложений.

В контексте баз данных стоит рассмотреть понятие СУБД. Система управления базами данных (СУБД) — это комплекс программных средств, необходимых для создания структуры новой базы, ее наполнения, редактирования содержимого и отображения информации. Наиболее распространенными СУБД являются MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

Это СУБД примеры типа клиент-сервер, именно такие СУБД встречаются чаще всего в контексте понятия хостинга. Их особенности:

* расположение СУБД на сервере с базами данных;
* непосредственный доступ к БД;
* централизованная обработка клиентских запросов на обработку данных;
* высокий уровень надежности, доступности и безопасности;
* повышенная нагрузка на сервер.

В свою очередь, для удобства работы с СУБД используются специальные веб-приложения, которые позволяют посредством графического интерфейса выполнять администрирование сервера баз данных, запускать специальные команды, а также работать с контентом таблиц и баз данных — действия, которые при отсутствии веб-приложения подлежат выполнению средствами консоли. Примеры: phpMyAdmin используется для администрирования СУБД MySQL, pgAdmin — для PostgreSQL.

Эффективное хранение, обработка и взаимодействие с данными – главные преимущества использование БД сегодня. Один из способов повышения эффективности обработки данных — организовать их эффективное хранение и получение. Самый распространенный подход к хранению данных на сегодня — использовать реляционную базу данных.

Цель исследования состоит в формировании и закреплении навыков применения, комплекса методов управления базами данных, серверами баз данных и создания клиентских приложений для них.

Объектом называется элемент информационной системы, сведения о котором хранятся в базе данных. Иногда объект также называют сущностью (от англ. entity).

Классом объектов называют их совокупность, обладающую одинаковым набором свойств.

Атрибут - это информационное отображение свойств объекта. Каждый объект характеризуется некоторым набором атрибутов.

Ключевым элементом данных называются такой атрибут (или группа атрибутов), который позволяет определить значения других элементов-данных.

Запись данных (от англ. record) - это совокупность значений связанных элементов данных.

Первичный ключ - это атрибут (или группа атрибутов), который уникальным образом идентифицируют каждый экземпляр объекта (запись).

Вторичным ключом называется атрибут (или группа атрибутов), значение которого может повторяться для нескольких записей (экземпляров объекта). Прежде всего, вторичные ключи используются в операциях поиска записей [3].

# описание предметной области, постановка задачи

В задании курсовой работы требуется разработать базу данных учёта нефтеперерабатывающих заводов. Рассмотрим особенности данной системы, указанные в условии.

Для автоматизации учета работы нефтеперерабатывающих заводов Украины необходима следующая информация: сведения о заводах (название, город, тип собственности (государственный, частный, ЗАО, ООО, …), год начала функционирования, телефон, страны-поставщики нефти (Казахстан, Россия, Туркмения, Украина,…), виды выпускаемого топлива (вид (дизельное, бензин А76, бензин А80, А92, А95, авиационный керосин,…), годовой объем (в тоннах), цена 1 тонны)), сведения о заказах на нефтепродукты (заказчик (название предприятия, город, телефон), вид топлива, объем (в тоннах), дата заказа).

# ОБОСНОВАНИЕ ВЫБОРА СУБД, описание возможностей СУБД

PostgreSQL не просто реляционная, а объектно-реляционная СУБД. Это даёт ему некоторые преимущества над другими SQL базами данных с открытым исходным кодом, такими как MySQL, MariaDB и Firebird.

Надежность PostgreSQL является проверенным и доказанным фактом и обеспечивается следующими возможностями:

полное соответствие принципам ACID - атомарность, непротиворечивость, изолированность, сохранность данных.

Atomicity - транзакция рассматривается как единая логическая единица, все ее изменения или сохраняются целиком, или полностью откатываются.

Consistency - транзакция переводит базу данных из одного непротиворечивого состояния (на момент старта транзакции) в другое непротиворечивое состояние (на момент завершения транзакции). Непротиворечивым считается состояние базы, когда выполняются все ограничения физической и логической целостности базы данных, при этом допускается нарушение ограничений целостности в течение транзакции, но на момент завершения все ограничения целостности, как физические, так и логические, должны быть соблюдены.

Isolation - изменения данных при конкурентных транзакциях изолированы друг от друга на основе системы версионности

Durability - PostgreSQL заботится о том, что результаты успешных транзакций гарантировано сохраняются на жесткий диск вне зависимости от сбоев аппаратуры.

многоверсионность (Multiversion Concurrency Control,MVCC) используется для поддержания согласованности данных в конкурентных условиях, в то время как в традиционных базах данных используются блокировки. MVCC означает, что каждая транзакция видит копию данных (версию базы данных) на время начала транзакции, несмотря на то, что состояние базы могло уже измениться. Это защищает транзакцию от несогласованных изменений данных, которые могли быть вызваны (другой) конкурентной транзакцией, и обеспечивает изоляцию транзакций. Основной выигрыш от использования MVCC по сравнению с блокировкой заключается в том, что блокировка, которую ставит MVCC для чтения не конфликтует с блокировкой на запись, и поэтому чтение никогда не блокирует запись и наоборот.

наличие Write Ahead Logging (WAL) - общепринятый механизм протоколирования всех транзакций, что позволяет восстановить систему после возможных сбоев. Основная идея WAL состоит в том, что все изменения должны записываться в файлы на диск только после того, как эти записи журнала, описывающие эти изменения будут и гарантировано записаны на диск. Это позволяет не сбрасывать страницы данных на диск после фиксации каждой транзакции, так как мы знаем и уверены, что сможем всегда восстановить базу данных используя журнал транзакций.

Point in Time Recovery (PITR) - возможность восстановления базы данных (используя WAL) на любой момент в прошлом, что позволяет осуществлять непрерывное резервное копирование кластера PostgreSQL.

Репликация также повышает надежность PostgreSQL. Существует несколько систем репликации, например, Slony, который является свободным и самым используемым решением, поддерживает master-slaves репликацию. Ожидается, что Slony-II будет поддерживать multi-master режим.

Целостность данных является сердцем PostgreSQL. Помимо MVCC, PostgreSQL поддерживает целостность данных на уровне схемы - это внешние ключи (foreign keys), ограничения (constraints).

Модель развития PostgreSQL, которая абсолютно прозрачна для любого, так как все планы, проблемы и приоритеты открыто обсуждаются. Пользователи и разработчики находятся в постоянном диалоге через мэйлинг листы. Все предложения, патчи проходят тщательное тестирование до принятия их в программное дерево. Большое количество бета-тестеров способствует тестированию версии до релиза и вычищению мелких ошибок.

Открытость кодов PostgreSQL означает их абсолютную доступность для любого, а либеральная BSD лицензия не накладывает никаких ограничений на использование кода.

Производительность PostgreSQL основывается на использовании индексов, интеллектуальном планировщике запросов, тонкой системы блокировок, системой управления буферами памяти и кэширования, превосходной масштабируемости при конкурентной работе.

Поддержка индексов

Стандартные индексы - B-tree, hash, R-tree, GiST (обобщенное поисковое дерево)

Частичные индексы (partial indices)

Функциональные индексы

Планировщик запросов основывается на стоимости различных планов, учитывая множество факторов. Он предоставляет возможность пользователю отлаживать запросы и настраивать систему.

Система блокировок поддерживает блокировки на нижнем уровне, что позволяет сохранять высокий уровень конкурентности при защите целостности данных. Блокировка поддерживается на уровне таблиц и записей. На нижнем уровне, блокировка для общих ресурсов оптимизирована под конкретную ОС и архитектуру.

Управление буферами и кэширование используют сложные алгоритмы для поддержания эффективности использования выделенных ресурсов памяти.

Tablespaces (табличные пространства) позволяют гибкое использование дискового пространства для хранения объектов системы, что также повышает производительность и масштабируемость.

Масштабируемость основывается на описанных выше возможностях. Низкая требовательность PostgreSQL к ресурсам и гибкая система блокировок обеспечивают его шкалирование, в то время как индексы и управление буферами обеспечивают хорошую управляемость системы даже при высоких загрузках.

Расширяемость PostgreSQL означает, что пользователь может настраивать систему путем определения новых функций, агрегатов, типов,языков, индексов и операторов. Объектно-ориентированность PostgreSQL позволяет перенести логику приложения на уровень базы данных, что сильно упрощает разработку клиентов, так как вся бизнес логика находится в базе данных. Функции в PostgreSQL однозначно определяются названием, количеством и типами аргументов..

# ОБОСНОВАНИЕ выбора инструментальных средств для написания клиентской части, проектирование структуры ПО

Для реализации клиентской части курсового проекта был выбран язык программирования С# и Framework WPF. Для разработки приложения использовалась среда разработки Microsoft Visual Studio.

Когда говорят C#, нередко имеют в виду технологии платформы .NET (Windows Forms, WPF, ASP.NET, Xamarin). И, наоборот, когда говорят .NET, нередко имеют в виду C#. Однако, хотя эти понятия связаны, отождествлять их неверно. Язык C# был создан специально для работы с фреймворком .NET, однако само понятие .NET несколько шире.

Как-то Билл Гейтс сказал, что платформа .NET - это лучшее, что создала компания Microsoft. Возможно, он был прав. Фреймворк .NET представляет мощную платформу для создания приложений. Можно выделить следующие ее основные черты:

Поддержка нескольких языков. Основой платформы является общеязыковая среда исполнения Common Language Runtime (CLR), благодаря чему .NET поддерживает несколько языков: наряду с C# это также VB.NET, C++, F#, а также различные диалекты других языков, привязанные к .NET, например, Delphi.NET. При компиляции код на любом из этих языков компилируется в сборку на общем языке CIL (Common Intermediate Language) - своего рода ассемблер платформы .NET. Поэтому мы можем сделать отдельные модули одного приложения на отдельных языках.

Кроссплатформенность. .NET является переносимой платформой (с некоторыми ограничениями). Например, последняя версия платформы на данный момент .NET Core поддерживается на большинстве современных ОС Windows, MacOS, Linux. Используя различные технологии на платформе .NET, можно разрабатывать приложения на языке C# для самых разных платформ - Windows, MacOS, Linux, Android, iOS, Tizen.

Мощная библиотека классов. .NET представляет единую для всех поддерживаемых языков библиотеку классов. И какое бы приложение мы не собирались писать на C# - текстовый редактор, чат или сложный веб-сайт - так или иначе мы задействуем библиотеку классов .NET.

Разнообразие технологий. Общеязыковая среда исполнения CLR и базовая библиотека классов являются основой для целого стека технологий, которые разработчики могут задействовать при построении тех или иных приложений. Например, для работы с базами данных в этом стеке технологий предназначена технология ADO.NET и Entity Framework Core. Для построения графических приложений с богатым насыщенным интерфейсом - технология WPF и UWP, для создания более простых графических приложений - Windows Forms. Для разработки мобильных приложений - Xamarin. Для создания веб-сайтов - ASP.NET и т.д.

Также еще следует отметить такую особенность языка C# и фреймворка .NET, как автоматическая сборка мусора. А это значит, что нам в большинстве случаев не придется, в отличие от С++, заботиться об освобождении памяти. Вышеупомянутая общеязыковая среда CLR сама вызовет сборщик мусора и очистит память..

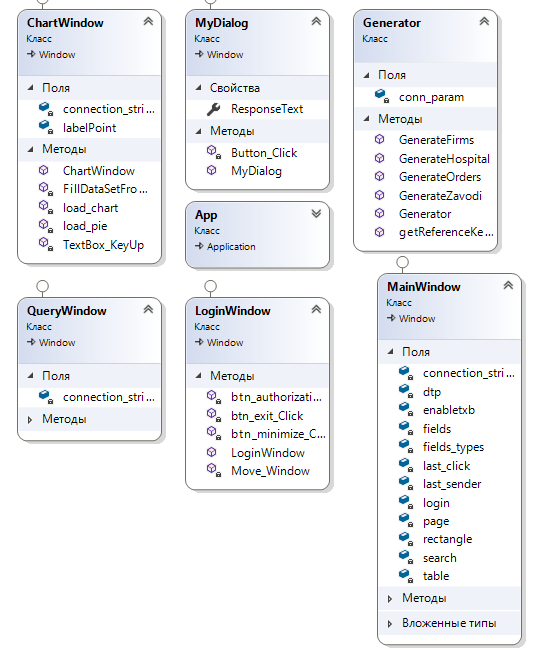


Рисунок 3.1 – Диаграмма классов

## 3.1 Невизуальные компоненты для работы с данным

Для создания проекта необходимо первым делом скачать драйвер PostgreSQL и добавить его в проект в качестве библиотеки “Npgsql”, что позволит в дальнейшем работать с базами данных.

### 3.2 Визуальные компоненты для работы с данными

Для отображения данных из таблиц был выбран элемент управления DataGridView из WindowsForms, который является достаточно гибким и позволяет автоматизировать вывод информации пользователю. Пример использования данного элемента управления для отображения таблицы из базы данных представлен на рисунках 3.2 и 3.3.



Рисунок 3.2 – Пример использования DataGridView для отображения данных таблицы



Рисунок 3.3 – Пример использования DataGridView для отображения данных таблицы (код программы)

Для отображения справочников при добавлении/редактировании записей в базе данных был использован элемент управления ComboBox, который обеспечивает удобное отображение и выбор данных из справочника. Примеры использования данного элемента управления приведен на рисунках 3.4 и 3.5.

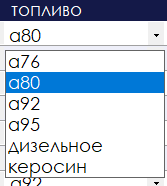


Рисунок 3.4 – Пример использования ComboBox для отображения справочников

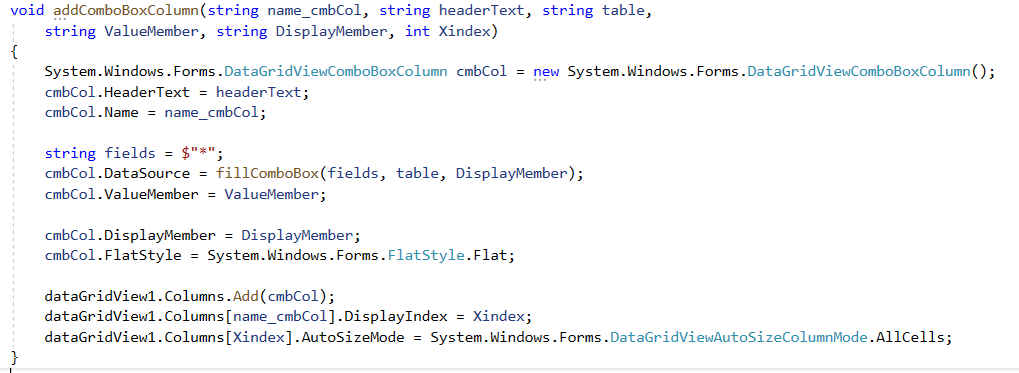


Рисунок 3.5 – Пример использования ComboBox для отображения справочников (код программы)

### 3.3 Разработка шаблонов приложений для работы с шаблонами базы данных

Данный программный продукт оснащён простым, интуитивно понятным пользовательским интерфейсом. В программе были созданы окна авторизации пользователя, работы с таблицами, работы с запросами и графиками.

Начало работы происходит в окне авторизации. Если авторизация происходит успешно, появляется окно для работы (добавление, удаление, изменение записей) с таблицами. Работа с таблицами происходит на основе разрешений конкретного пользователя.

Из окна для работы с таблицами пользователь может перейти на окно с перечнем запросов. В нём пользователь может выполнить запрос, получить результат, экспортировать результаты в excel, просмотреть некоторые итоговые запросы в виде графиков.

Рассмотрим особенности каждой формы на рисунках 3.6 - 3.10.

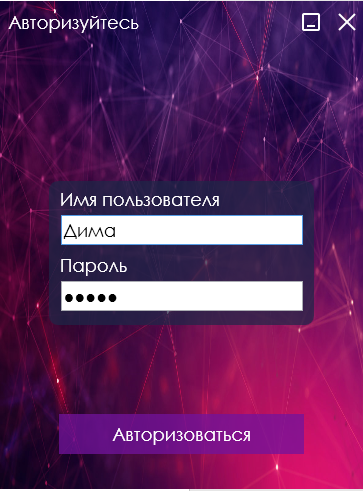


Рисунок 3.6 – Форма авторизации

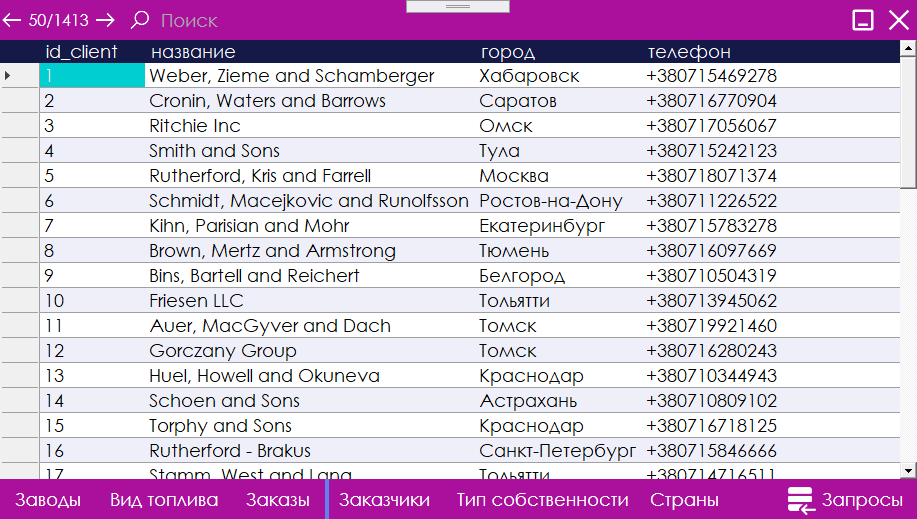


Рисунок 3.7 – Форма работы с таблицами

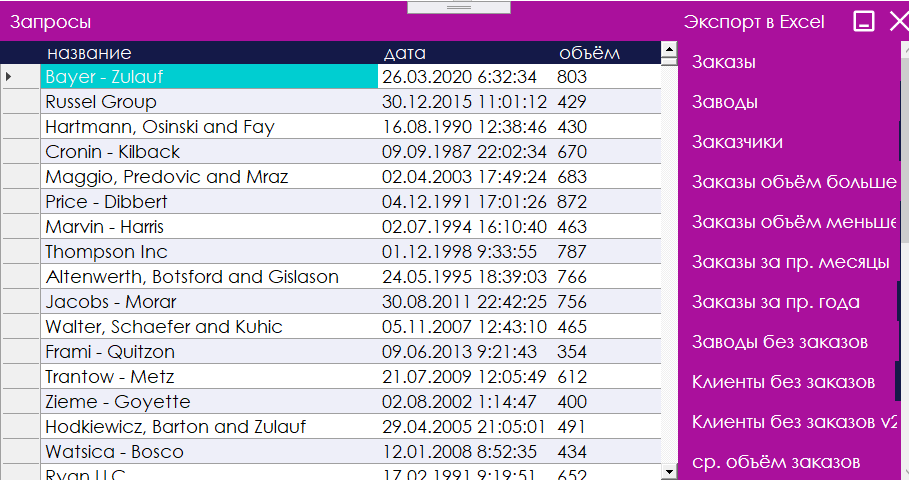


Рисунок 3.8 – Форма запросов



Рисунок 3.9 – Форма с графиками

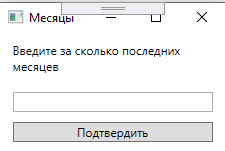


Рисунок 3.10 – Окно ввода параметров запроса

# ПРОЕКТИРОВАНИЕ БАзы данных в выбранной субд

### 4.1 Проектирование концептуальной модели БД

Процесс проектирования БД с использованием метода нормальных форм (НФ) является итерационным и заключается в последовательном переводе отношения из 1НФ в НФ более высокого порядка по определенным правилам. Каждая следующая НФ ограничивается определенным типом функциональных зависимостей и устранением соответствующих аномалий при выполнении операций над отношениями БД, а также сохранении свойств, предшествующих НФ.

Нормализация базы данных сводит к минимуму количество избыточной информации. Ее целью является сохранять данные только один раз, но в нужном месте. Нормализованная база данных исключает дублирование и многократное обслуживание данных, а также появление проблем с целостностью данных, возникающих при повторном вводе одинаковых данных. Первоначально доктором Эдгаром Коддом были определены только 3 нормальные формы. Дальнейшая разработка реляционной теории привела к появлению еще нескольких форм и на данный момент их насчитывается 6.

На практике соответствие базы данных правилам 3-ей нормальной формы вполне достаточно.

В результате выполнения нормализации базы данных (приведение в 3НФ) были выделены следующие таблицы:

1. Заказы (таблица 1.1);
2. Заводы (таблица 1.2);
3. Заказчики (таблица 1.3);
4. Страны (таблица 1.4);
5. Тип собственности (таблица 1.5);
6. вид топлива (таблица 1.6);

## Создание таблиц, доменов, индексов

Таблица 4.1 – Заказы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип | Размер | Описание |
| 1 | id\_order | Числовой | 4 | Уникальный идентификатор |
| 2 | id\_client | Числовой | 4 | Уникальный идентификатор |
| 3 | id\_fuel | Числовой | 4 | Уникальный идентификатор |
| 4 | объем | Числовой | 4 | Объем |
| 5 | дата | Дата | 4 | Дата составления заказа |
| 6 | id\_plants | Числовой | 4 | Уникальный идентификатор |
| 7 | login | Текстовый | 64 | Для Row level security |

Таблица 4.2 – Заводы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип | Размер | Описание |
| 1 | id\_plants | Числовой | 4 | Уникальный идентификатор |
| 2 | название | Текстовый | 64 | Форма выпуска |
| 3 | город | Текстовый | 64 | Город |
| 4 | id\_type | Числовой | 4 | Уникальный идентификатор |
| 5 | год | Числовой | 4 | Год создания |
| 6 | телефон | Domain | 12 | Телефон |
| 7 | id\_country | Числовой | 4 | Уникальный идентификатор |
| 8 | id\_fuel | Числовой | 4 | Уникальный идентификатор |
| 9 | объем | Числовой | 4 | Объем |
| 10 | цена | Числовой | 4 | Цена |

Таблица 4.3 – Заказчики

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип | Размер | Описание |
| 1 | id\_client | Числовой | 4 | Уникальный идентификатор |
| 2 | название | Текстовый | 64 | Группа |
| 3 | город | Текстовый | 64 | Город |
| 4 | телефон | Domain | 12 | Телефон |

Таблица 4.4 – страны

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип | Размер | Описание |
| 1 | id\_country | Числовой | 4 | Уникальный идентификатор |
| 2 | страна | Текстовый | 64 | Страна |

Таблица 4.5 – тип собственности

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип | Размер | Описание |
| 1 | id\_type | Числовой | 4 | Уникальный идентификатор |
| 2 | тип | Текстовый | 64 | Тип собственности |

Таблица 4.6 – вид топлива

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Поле | Тип | Размер | Описание |
| 1 | id\_fuel | Числовой | 4 | Уникальный идентификатор |
| 2 | топливо | Текстовый | 64 | Вид топлива |

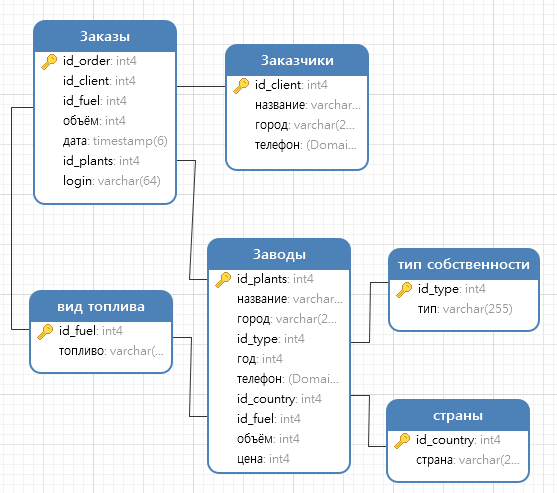


Рисунок 4.1 – Схема базы данных

В ходе разработки базы данных был создан домен phone, который представляет собой телефонный номер и имеющий проверку на корректность.

CREATE DOMAIN создаёт новый домен. Домен по сути представляет собой тип данных с дополнительными условиями (ограничивающими допустимый набор значений). Владельцем домена становится пользователь его создавший.

Если задаётся имя схемы (например, CREATE DOMAIN myschema.mydomain ...), домен создаётся в указанной схеме, в противном случае — в текущей. Имя домена должно быть уникальным среди имён типов и доменов, существующих в этой схеме.

Домены полезны для абстрагирования и вынесения общих характеристик разных полей в единое место для упрощения сопровождения. Например, в нескольких таблицах может присутствовать столбец, содержащий электронный адрес, и для всех требуются одинаковые ограничения CHECK, проверяющие синтаксис адреса. В этом случае лучше определить домен, а не задавать для каждой таблицы отдельные ограничения.

Чтобы создать домен, необходимо иметь право USAGE для нижележащего типа.



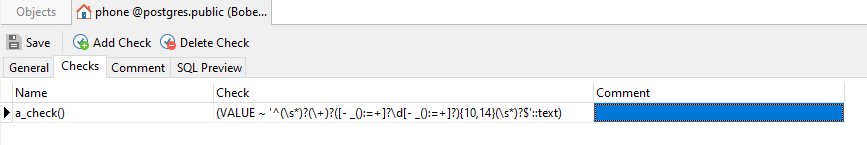


Рисунок 4.2 – Домен phone

Во время анализа запросов к базе данных возникла необходимость добавить индекс b-tree к полю дата таблицы Заказы для эффективного поиска и фильтрации данных.

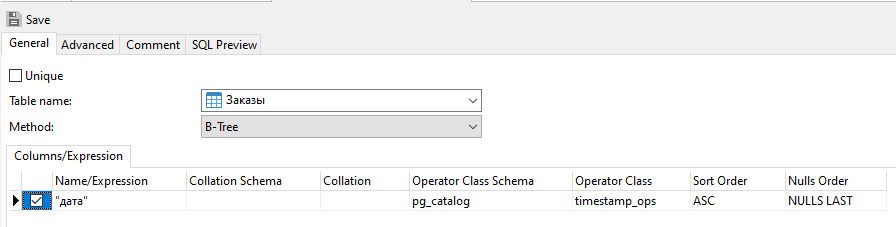


Рисунок 4.3 – Индекс поля дата таблицы Заказы

### 4.3 Разработка триггеров

Были разработаны триггеры before insert для всех таблиц и триггеры для каждого типа события.

CREATE TRIGGER создаёт новый триггер. Триггер будет связан с указанной таблицей, представлением или сторонней таблицей и будет выполнять заданную функцию имя\_функции при определённых событиях.

Триггер можно настроить так, чтобы он срабатывал до операции со строкой (до проверки ограничений и попытки выполнить INSERT, UPDATE или DELETE) или после её завершения (после проверки ограничений и выполнения INSERT, UPDATE или DELETE), либо вместо операции (при добавлении, изменении и удалении строк в представлении). Если триггер срабатывает до или вместо события, он может пропустить операцию с текущей строкой, либо изменить добавляемую строку (только для операций INSERT и UPDATE). Если триггер срабатывает после события, он «видит» все изменения, включая результат действия других триггеров.

Триггер с пометкой FOR EACH ROW вызывается один раз для каждой строки, изменяемой в процессе операции. Например, операция DELETE, удаляющая 10 строк, приведёт к срабатыванию всех триггеров ON DELETE в целевом отношении 10 раз подряд, по одному разу для каждой удаляемой строки. Триггер с пометкой FOR EACH STATEMENT, напротив, вызывается только один раз для конкретной операции, вне зависимости от того, как много строк она изменила (в частности, при выполнении операции, изменяющей ноль строк, всё равно будут вызваны все триггеры FOR EACH STATEMENT). Заметьте, что при выполнении INSERT с предложением ON CONFLICT DO UPDATE сработают оба триггера уровня операторов, для INSERT и для UPDATE.

Триггеры, срабатывающие в режиме INSTEAD OF, должны быть помечены FOR EACH ROW и могут быть определены только для представлений. Триггеры BEFORE и AFTER для представлений должны быть помечены FOR EACH STATEMENT.

Чтобы создать триггер, пользователь должен иметь право TRIGGER для этой таблицы. Также пользователь должен иметь право EXECUTE для триггерной функции.

Для удаления триггера применяется команда DROP TRIGGER.

Триггер для избранных столбцов (определённый с помощью UPDATE OF имя\_столбца) будет срабатывать, когда его столбцы перечислены в качестве целевых в списке SET команды UPDATE. Изменения, вносимые в строки триггерами BEFORE UPDATE, при этом не учитываются, поэтому значения столбцов можно изменить так, что триггер не сработает. И наоборот, при выполнении команды UPDATE ... SET x = x ... триггер для столбца x сработает, хотя значение столбца не меняется.

В триггере BEFORE условие WHEN вычисляется непосредственно перед возможным вызовом функции, поэтому проверка WHEN существенно не отличается от проверки того же условия в начале функции триггера. В частности, учтите, что строка NEW, которую видит ограничение, содержит текущие значения, возможно изменённые предыдущими триггерами. Кроме того, в триггере BEFORE условие WHEN не может проверять системные столбцы в строке NEW (например, oid), так как они ещё не установлены.

В триггере AFTER условие WHEN проверяется сразу после изменения строки, и если оно выполняется, событие запоминается, чтобы вызвать триггер в конце оператора. Если же для триггера AFTER условие WHEN не выполняется, нет необходимости запоминать событие для последующей обработки или заново перечитывать строку в конце оператора. Это приводит к значительному ускорению операторов, изменяющих множество строк, когда триггер должен срабатывать только для некоторых из них.

Триггеры уровня операторов для представления срабатывают, только если операция с представлением обрабатывается триггером уровня строк INSTEAD OF. Если операция обрабатывается правилом INSTEAD, то вместо исходного оператора, обращающегося к представлению, выполняются те операторы, что генерирует правило, поэтому вызываться будут триггеры, связанные с таблицами, к которым обращаются эти заменяющие операторы. Аналогично, для автоматически изменяемого представления выполнение операции сводится к переписыванию оператора в виде операции с базовой таблицей представления, так что срабатывать будут триггеры уровня операторов для базовой таблицы.

В PostgreSQL до версии 7.3 обязательно требовалось объявлять триггерные функции, как возвращающие фиктивный тип opaque, а не trigger. Для поддержки загрузки старых файлов экспорта БД, команда CREATE TRIGGER принимает функции с объявленным типом результата opaque, но при этом выдаётся предупреждение и тип результата меняется на trigger.

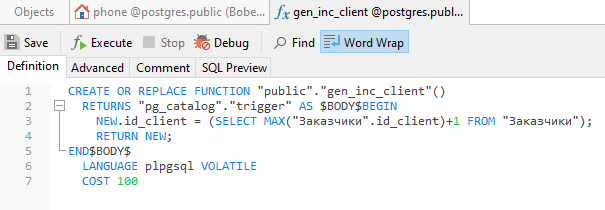


Рисунок 4.4 – Функция триггера before insert для генерации ключа

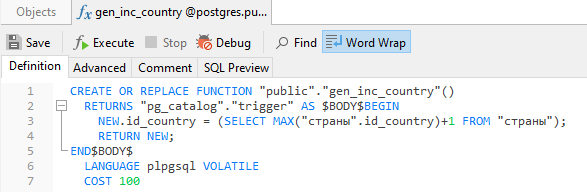


Рисунок 4.5 – Функция триггера before insert для генерации ключа

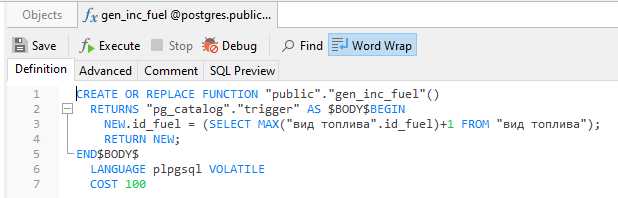


Рисунок 4.6 – Функция триггера before insert для генерации ключа



Рисунок 4.7 – Функция триггера before insert для генерации ключа



Рисунок 4.8 – Функция триггера before insert для генерации ключа

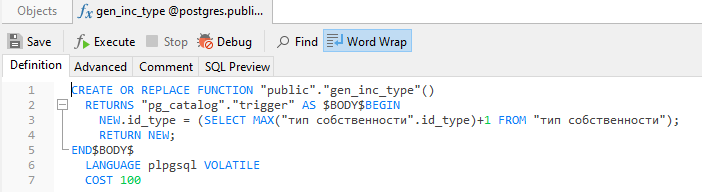


Рисунок 4.9 – Функция триггера before insert для генерации ключа

### 4.4 Проектирование запросов к базе данных

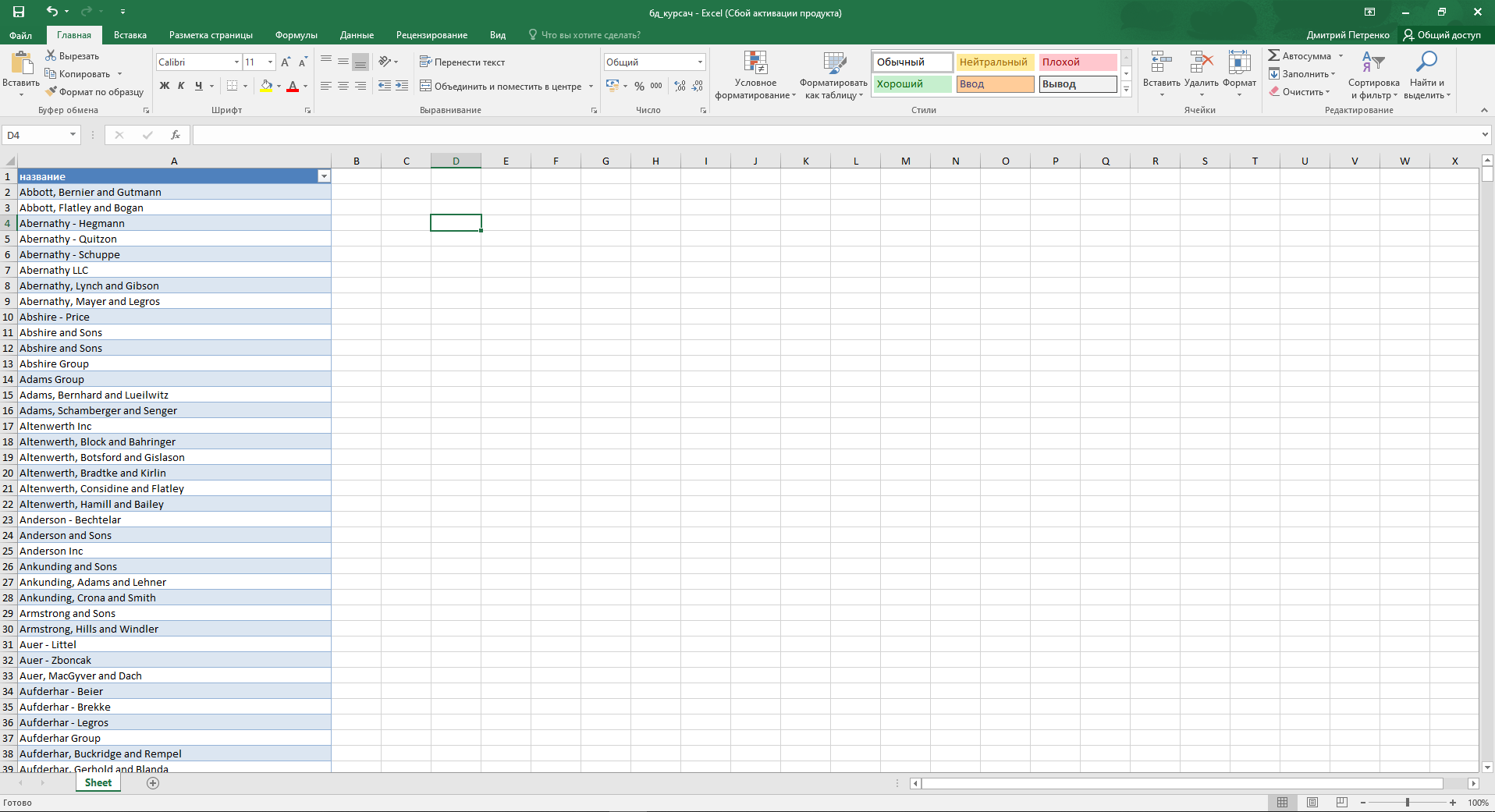
Во время обучения синтаксису SQL необходимо было разработать запросы следующих типов:

* симметричное внутреннее соединение с условием (два запроса с условием отбора по внешнему ключу, два – по датам);
* симметричное внутреннее соединение без условия (три запроса);
* левое внешнее соединение;
* правое внешнее соединение;
* запрос на запросе по принципу левого соединения;
* итоговый запрос без условия;
* итоговый запрос без условия c итоговыми данными вида: «всего», «в том числе»;
* итоговые запросы с условием на данные (по значению, по маске, с использованием индекса, без использования индекса);
* итоговый запрос с условием на группы;
* итоговый запрос с условием на данные и на группы;
* запрос на запросе по принципу итогового запроса;
* запрос с использованием объединения
* запросы с подзапросами (с использованием in, not in, case).

Запросы — это объект базы данных, который служит для извлечения данных из таблиц и предоставления их пользователю в удобном виде. Особенность запросов состоит в том, что они черпают данные из базовых таблиц и создают на их основе временную таблицу. Применение запросов позволяет избежать дублирования данных в таблицах и обеспечивает максимальную гибкость при поиске и отображении данных в базе данных.

### 4.5 Создание представлений и хранимых процедур, функций

На базе ранее представленных запросов разработаны отчёты Excel и представлены на рисунке 4.13.

Рисунок 4.13 – Отчет Excel

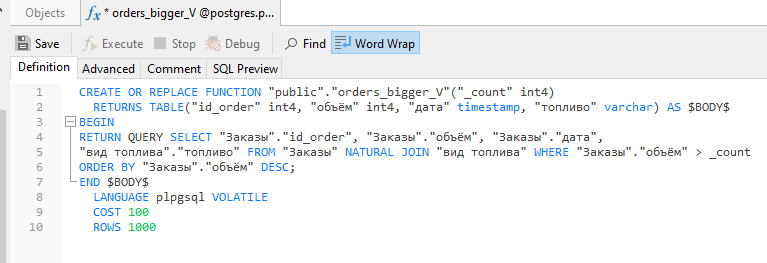




Рисунок 4.14 – Симметричное внутреннее соединение с условием отбора по внешнему ключу (Вывести заказы объем больше)

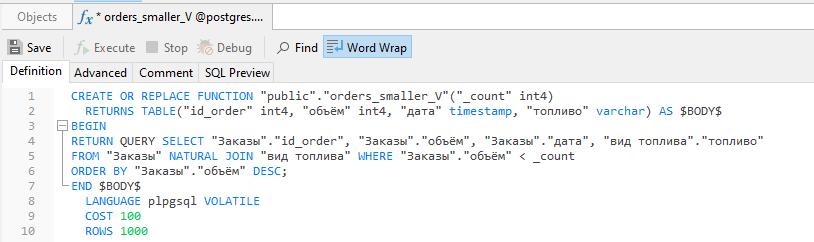


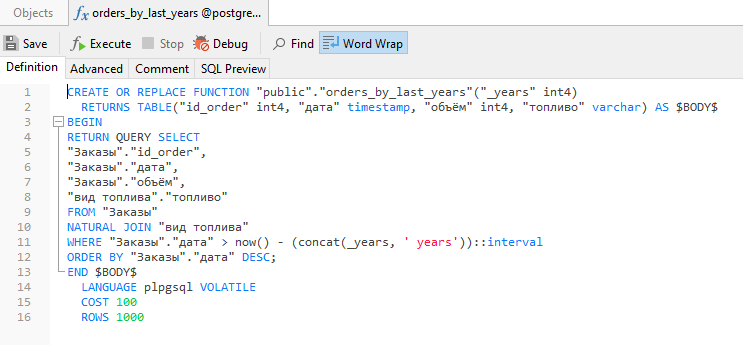


Рисунок 4.15 – Симметричное внутреннее соединение с условием отбора по внешнему ключу (Вывести заказы объем меньше)





Рисунок 4.16 – Симметричное внутреннее соединение с условием отбора по дате (Вывести заказы за последние N месяцев)



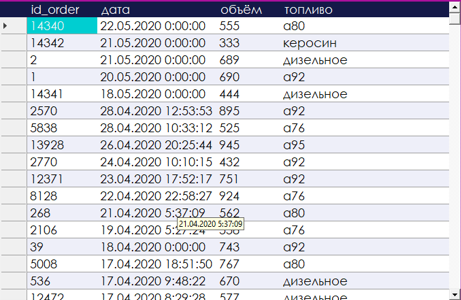


Рисунок 4.17 – Симметричное внутреннее соединение с условием отбора по дате (Вывести заказы за последние N лет)

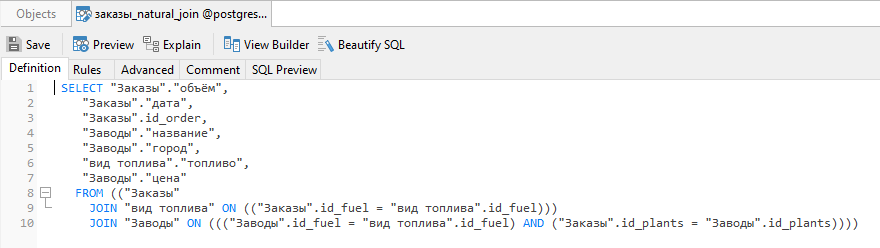
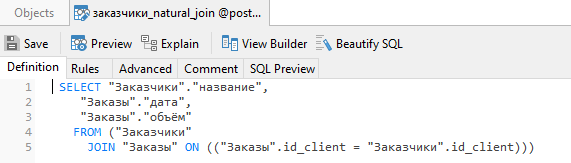




Рисунок 4.18 – Симметричное внутреннее соединение без условия (вывести информацию о заказах)



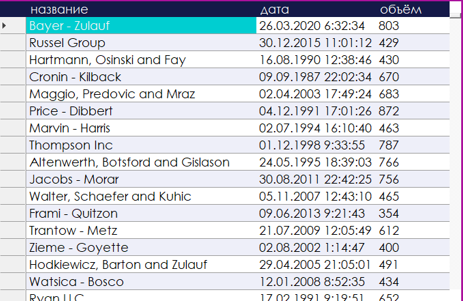
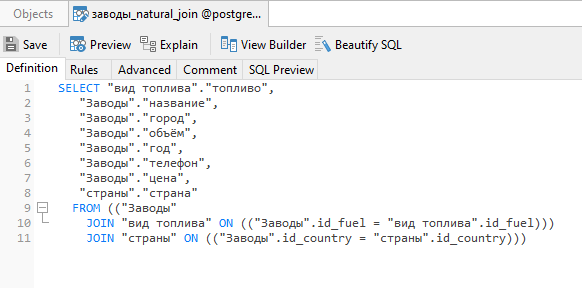


Рисунок 4.19 – Симметричное внутреннее соединение без условия (вывести информацию о заказчиках)



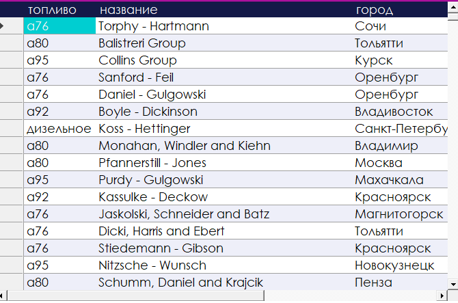
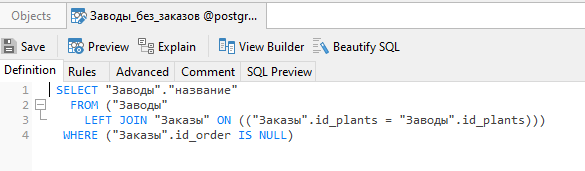


Рисунок 4.20 – Симметричное внутреннее соединение без условия (вывести информацию о заводах)



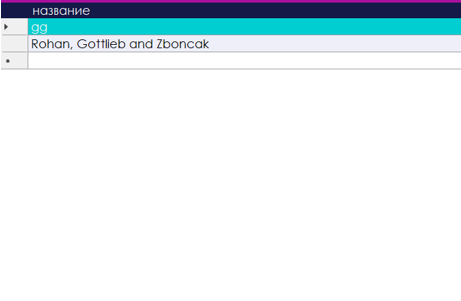
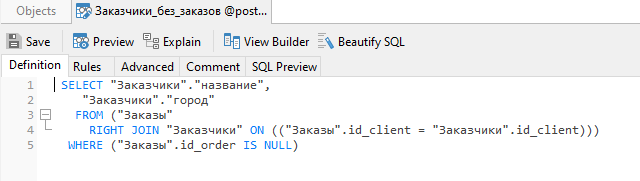


Рисунок 4.21 – Левое внешнее соединение (заводы без заказов)



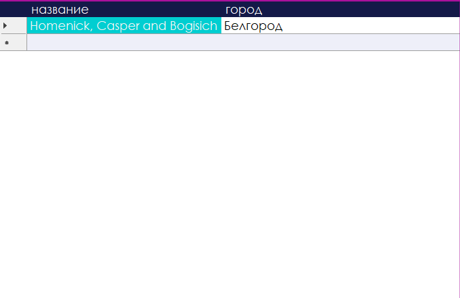
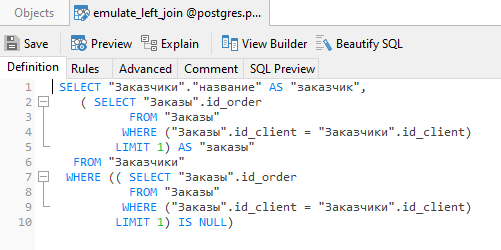


Рисунок 4.22 – Правое внешнее соединение (клиенты без заказов)



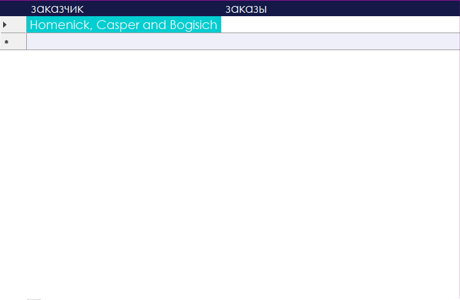
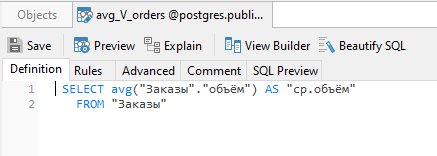


Рисунок 4.23 – Запрос на запросе по принципу левого соединения (клиенты без заказов)



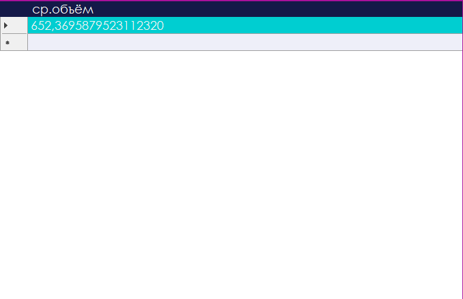
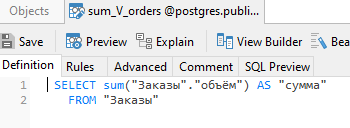


Рисунок 4.24 – Итоговый запрос без условия (Средняя объем заказов)



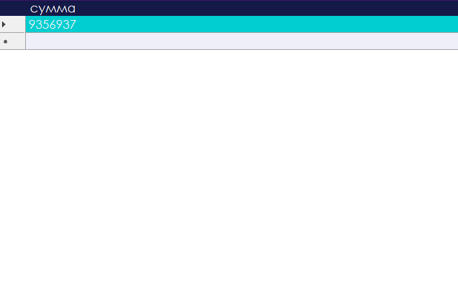


Рисунок 4.25 – Итоговый запрос без условия с итоговыми данными вида «всего», «в том числе» (Сумма заказов)



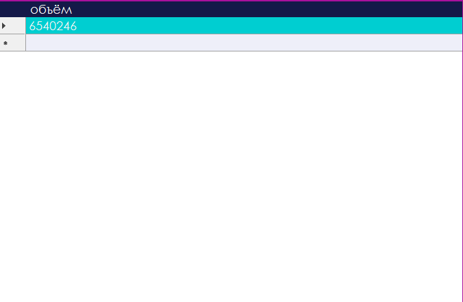


Рисунок 4.26 – Итоговый запрос с условием на данные по значению (заказы средняя цена больше N)



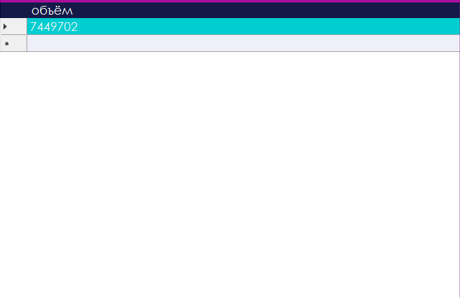
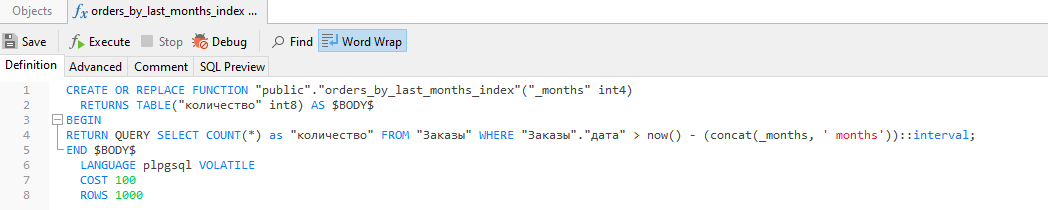
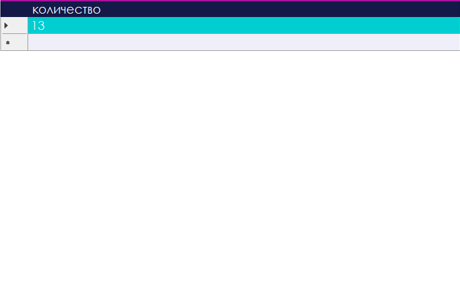


Рисунок 4.27 – Итоговый запрос с условием на данные по маске (объем топлива по маске)





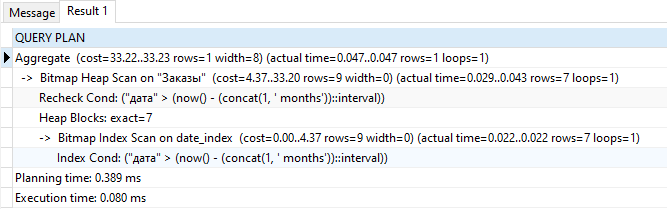
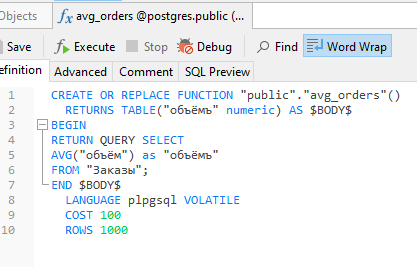
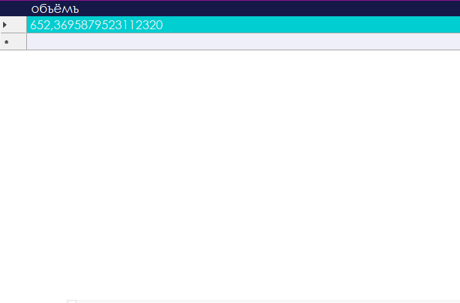


Рисунок 4.28 – Итоговый запрос с условием на данные с индексом (кол-во заказов за последние N месяцев)





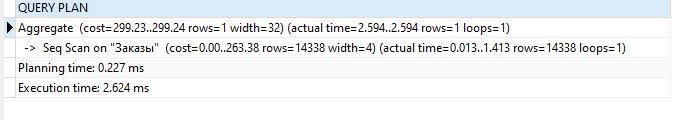
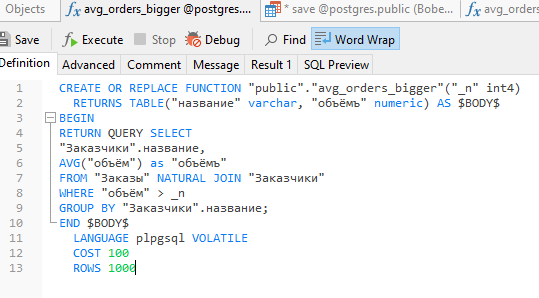


Рисунок 4.29 – Итоговый запрос с условием на данные без индекса (средние объем заказов)



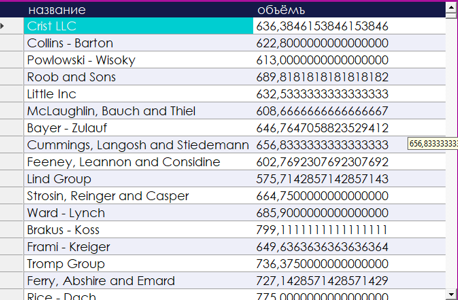
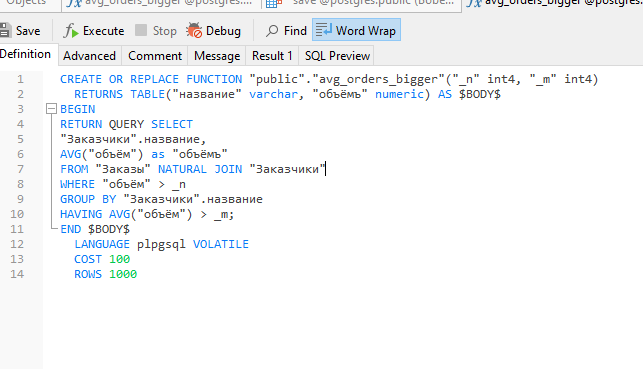


Рисунок 4.30 – Итоговый запрос с условием на группы



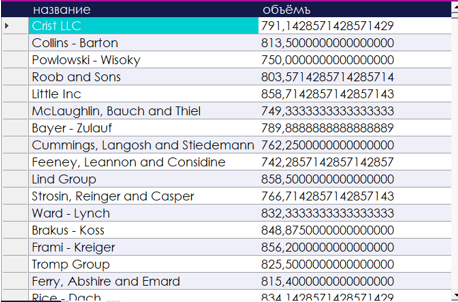
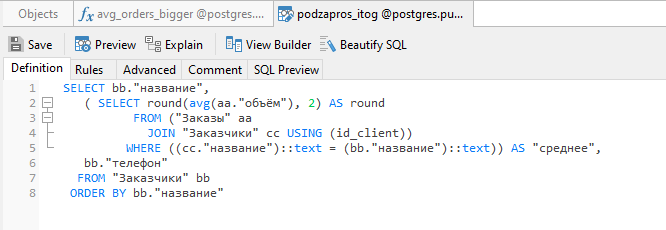


Рисунок 4.31 – Итоговый запрос с условием на данные и на группы



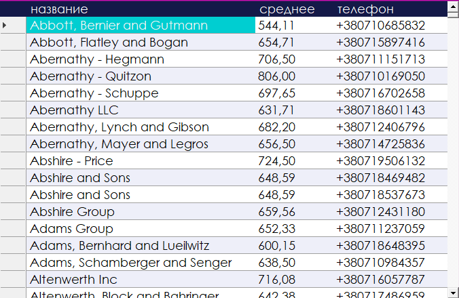
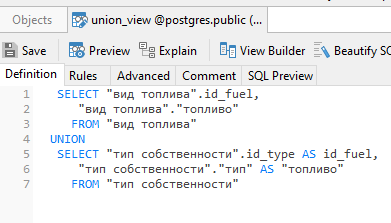


Рисунок 4.32 – Запрос на запросе по принципу итогового запроса



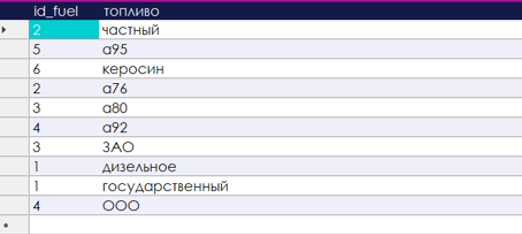
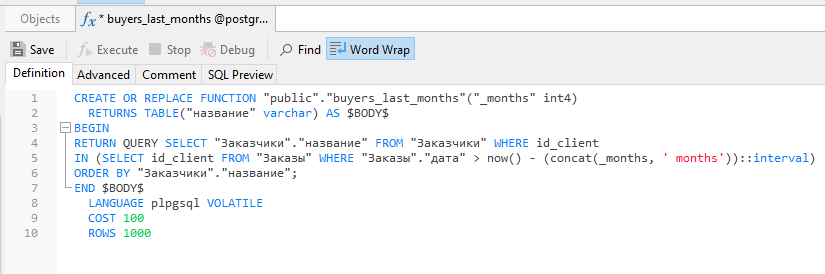


Рисунок 4.33 – Запрос с использованием объединения



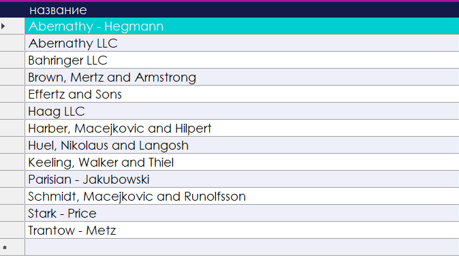
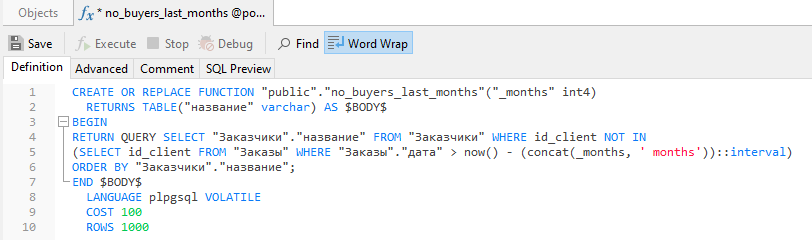


Рисунок 4.34 – Запрос с подзапросом in



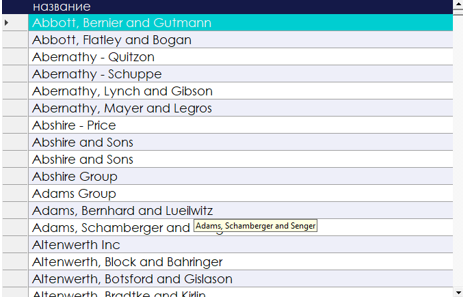
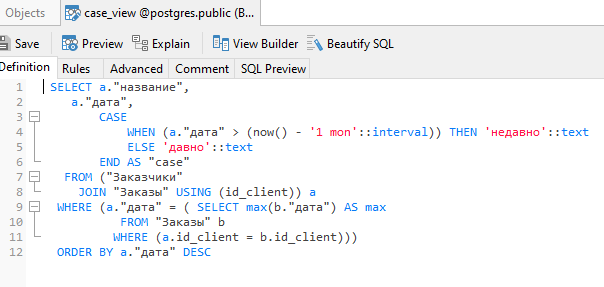


Рисунок 4.35 – Запрос с подзапросом not in



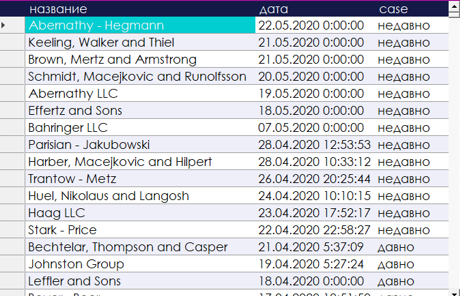


Рисунок 4.36 – Запрос с case

# Разработка клиентского приложения

### 5.1 Формы и компоненты для работы с основными таблицами



Рисунок 5.1 – форма для работы с основной таблицей

### 5.2 Формы и компоненты для работы со справочниками

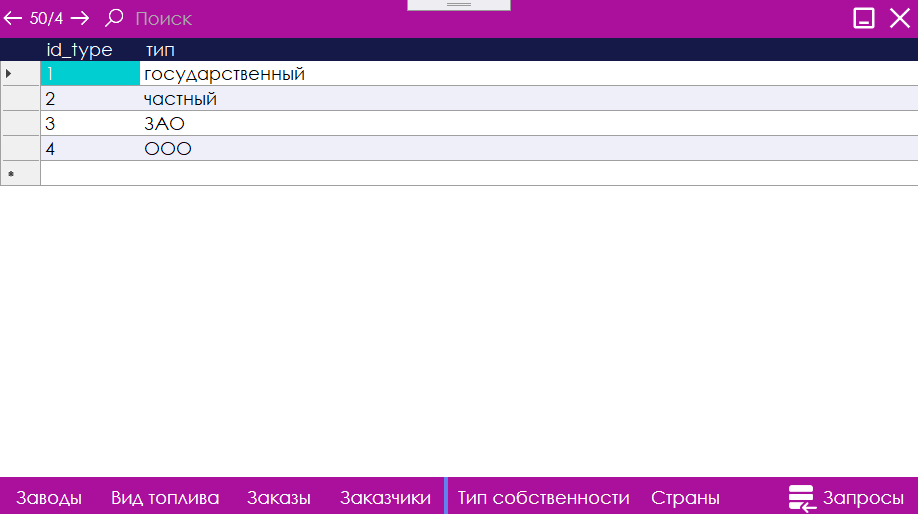
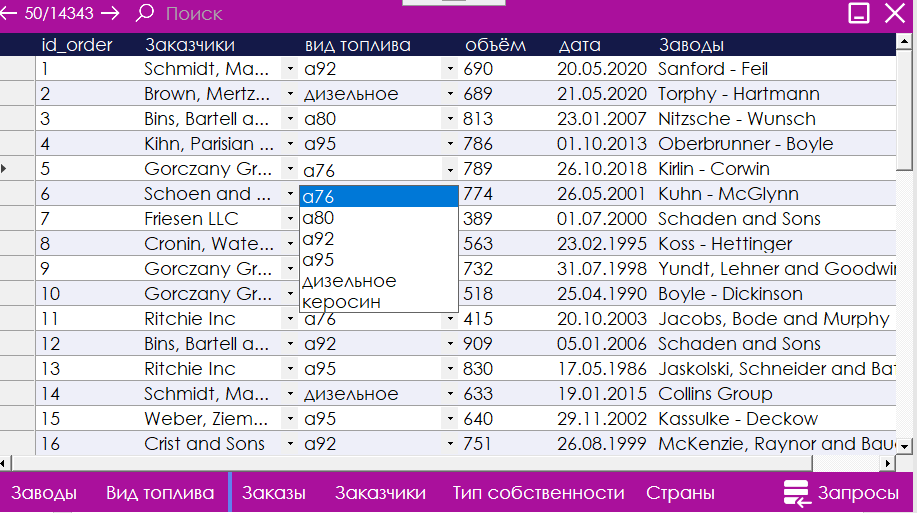


Рисунок 5.2 – форма для работы с основной таблицей

 Рисунок 5.3 – пример выпадающего списка, заполняемого по справочнику

### 5.3 Формы и компоненты для отображения результатов запросов

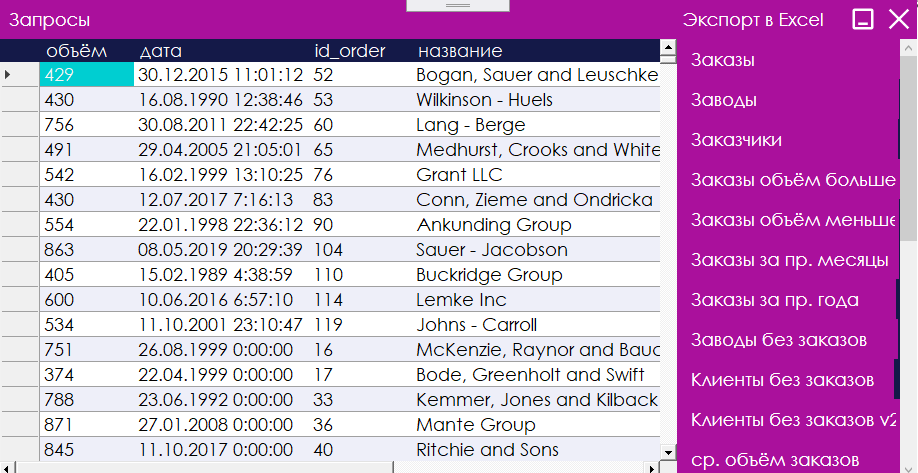


Рисунок 5.4 – форма для отображения запросов

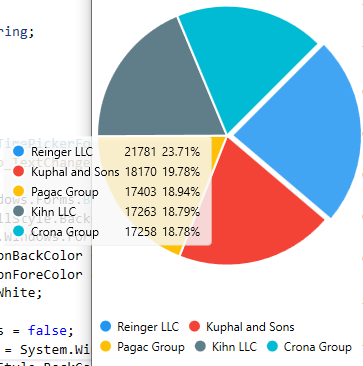


Рисунок 5.5 – круговая диаграмма Топ Заказчиков

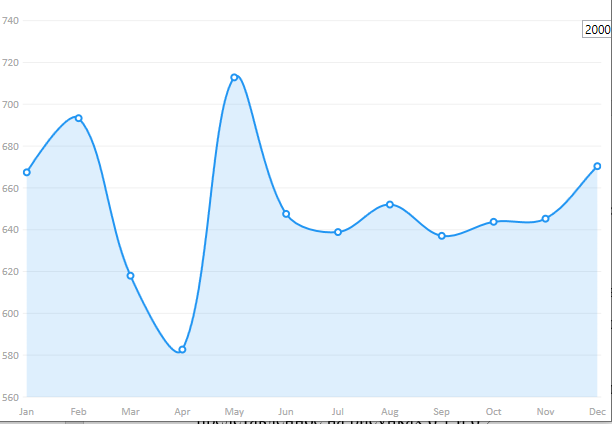


Рисунок 5.6 – график средней цены заказов по месяцам в 2000г

# Тестирование разработанной информационной системы

После создания информационной системы было проведено его тестирования. Была проверена возможность единовременной работы с данными, каскадное удаление, изменение.

При ошибочном вводе данных пользователю выводится сообщение, представленное на рисунках 6.1 и 6.2

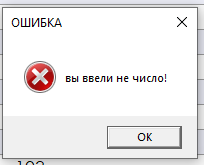


Рисунок 6.1 – Ошибка при вводе неправильных данных

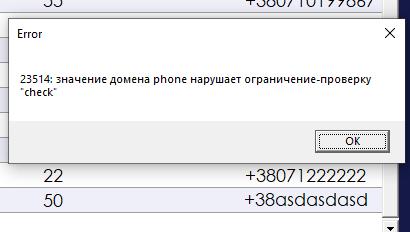


Рисунок 6.2 – Ошибка при вводе не правильного номера телефона

При вводе неправильных входных данных пользователь получает сообщение об ошибке авторизации, представленная на рисунке 6.3

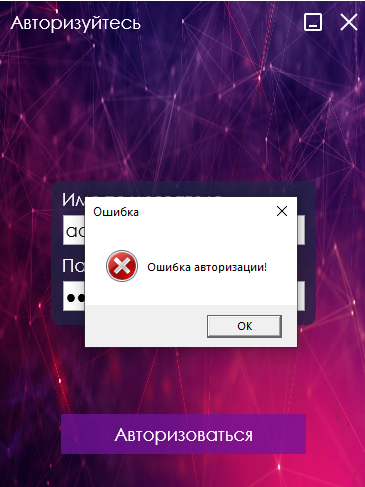


Рисунок 5.3 – Ошибка при некорректно заполненных данных

Было созданы роли: администратора и сотрудников. Администратор имеет доступ почти ко всем элементам созданной базы данных, а у сотрудников есть доступ исключительно к своим записям в таблице Заказы. Остальные сотрудники наследуются от этих ролей.

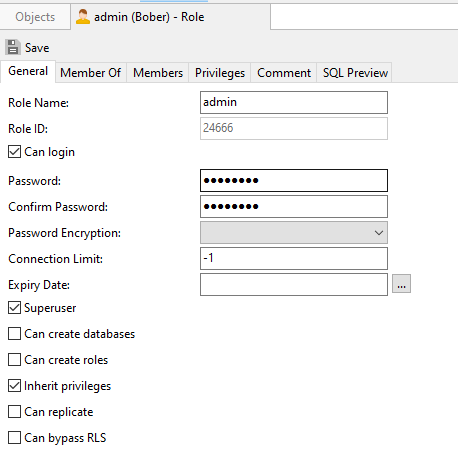


Рисунок 6.4 – пользователь admin

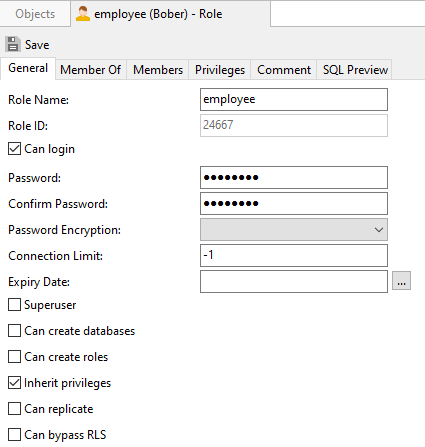


Рисунок 6.5 – пользователь employee

Для таблицы Заказы был включен модификатор ROW LEVEL SECURITY, который защищает данные в таблице на уровне строк.

На рисунке 6.6 сказано, что роль admin имеет доступ ко всем строкам таблицы Заказы, а роль employee имеет доступ только к записям, которые создал он сам. Работу данного механизма обеспечивает триггер before\_insert\_orders, который перед вставкой в таблицу заполняет поле владельца записи текущего пользователя.

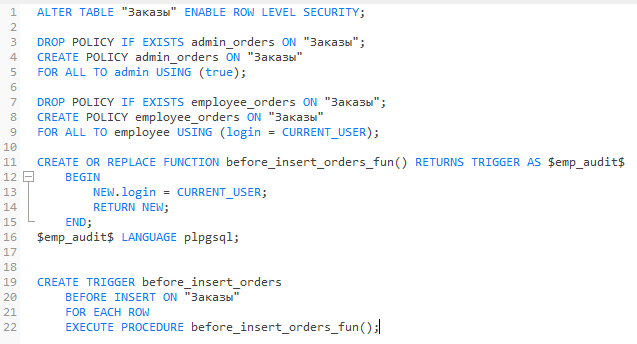


Рисунок 6.6 – ROW LEVEL SECURITY таблицы Заказы

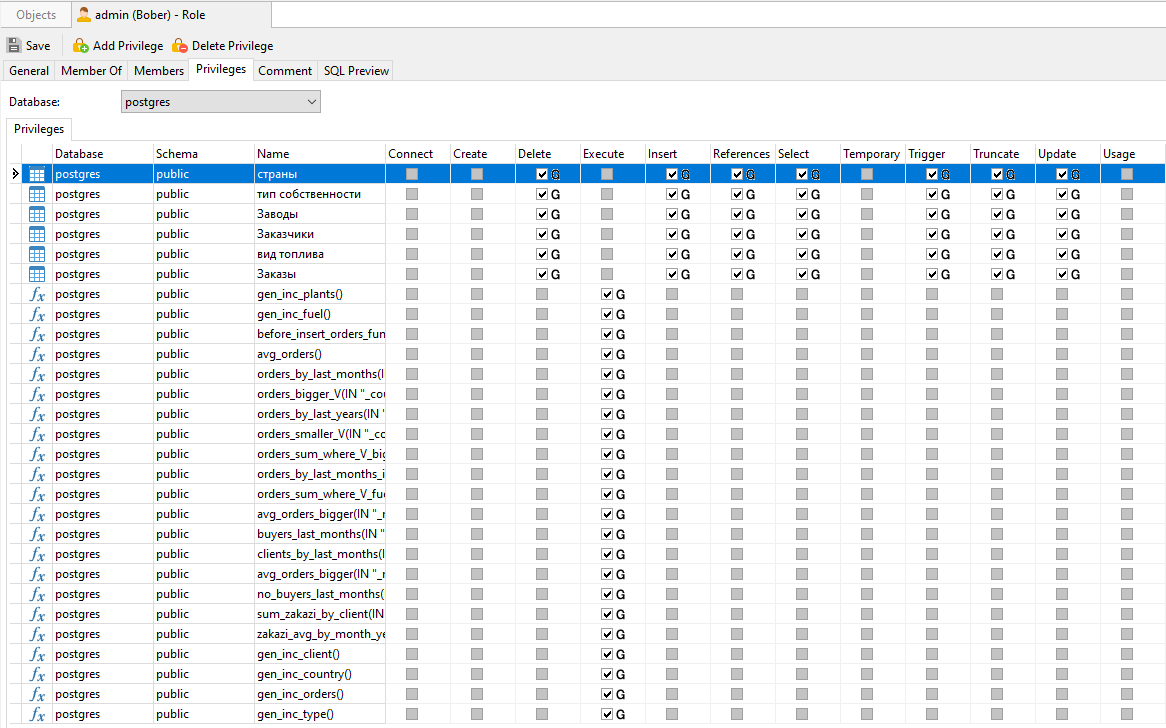


Рисунок 6.7 - Привилегии пользователя admin

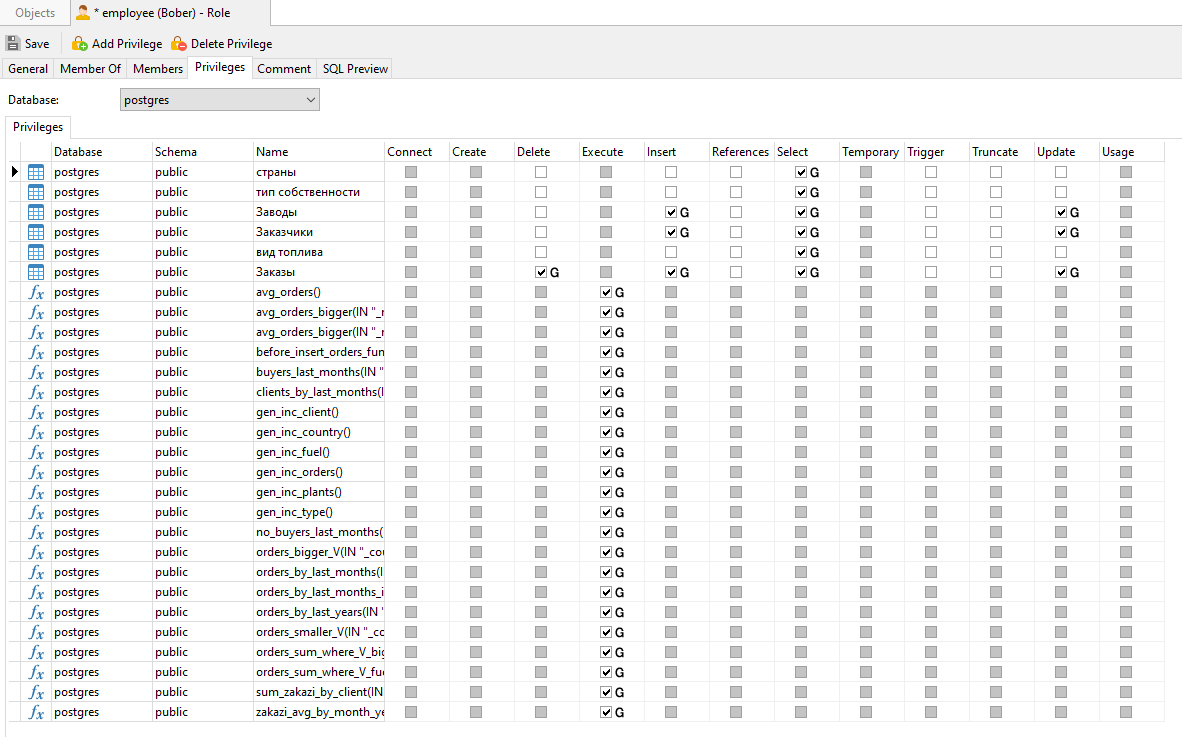


Рисунок 6.8 – привилегии пользователя employee

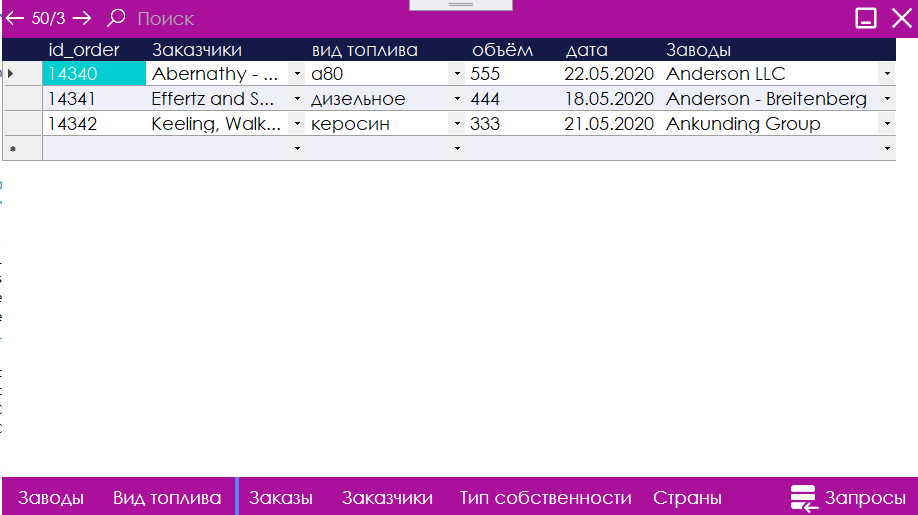


Рисунок 6.9 – Заказы созданные пользователем Дима (RLS)

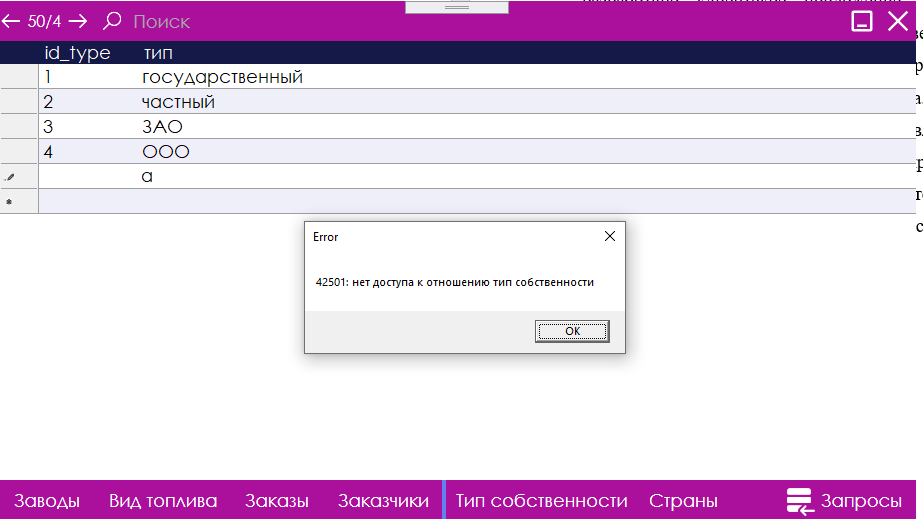


Рисунок 6.10 – Попытка пользователем Дима работы с запрещенной таблицей

ЗАКЛЮЧЕНИЕ

В течение разработки курсового проекта была изучена предметная область проекта, разработана концептуальная модель БД, были написаны основные запросы и функции для данной предметной области, было разработано клиентское приложение. Также были изучены тонкости проектирования и разработки клиент-серверных приложений.

Система обладает уникальной авторизацией, базовым уровнем защиты данных от несанкционированного доступа.

Недостатками данной программы является то, что отображение данных часто производится с задержкой по причине вычисления данных и не осуществлено разделение ролей пользователей.

В результате создания данной системы требования, изложенные в постановке задачи, выполнены.

У Постгреса множество возможностей. Созданный с использованием объектно-реляционной модели, он поддерживает сложные структуры и широкий спектр встроенных и определяемых пользователем типов данных. Он обеспечивает расширенную ёмкость данных и заслужил доверие бережным отношением к целостности данных. Возможно, вам не понадобятся все те продвинутые функции хранения данных, которые мы исследовали в этой статье, но, поскольку потребности могут быстро возрасти, есть несомненное преимущество в том, чтобы иметь всё это под рукой.

# Перечень ссылок

1. PostgreSQL [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/PostgreSQL, свободный.- Загл. с экрана.
2. Триггер [Электронный ресурс] Режим доступа: https://ru.wikipedia.org/wiki/Триггер\_(базы\_данных) , свободный.- Загл. с экрана.
3. Запрос [Электронный ресурс] Режим доступа: http://miheevag.narod.ru/db.htm , свободный.- Загл. с экрана.
4. Уотсон, К. Visual C# 2008. Базовый курс / Карли Уотсон, Кристиан Нейгел, Якоб Хаммер Педерсен, Джон Д. Рид, Морган Скиннер, Эрик Уайт// «Вильямс», 2009.– 1216c.

Приложение А Техническое задание

Приложение Б листниг ШАБЛОНОВ

 Рисунок Б.1 – Главная форма клиентского приложения

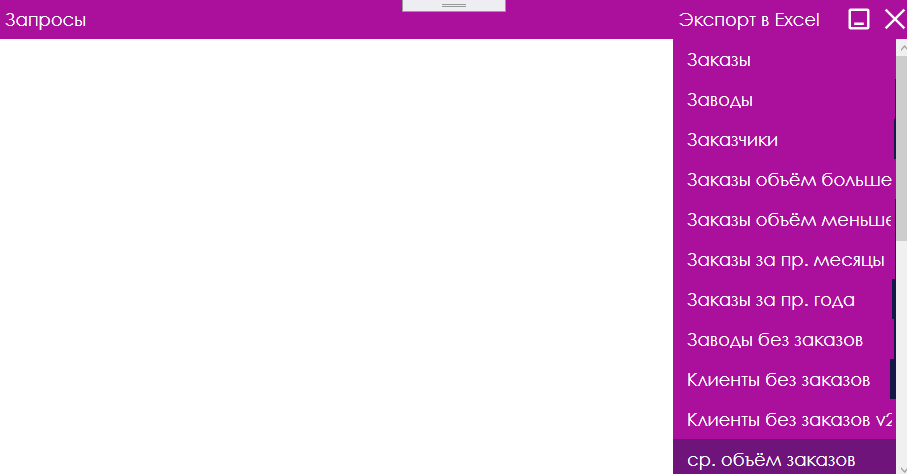


Рисунок Б.2 – Форма запросов

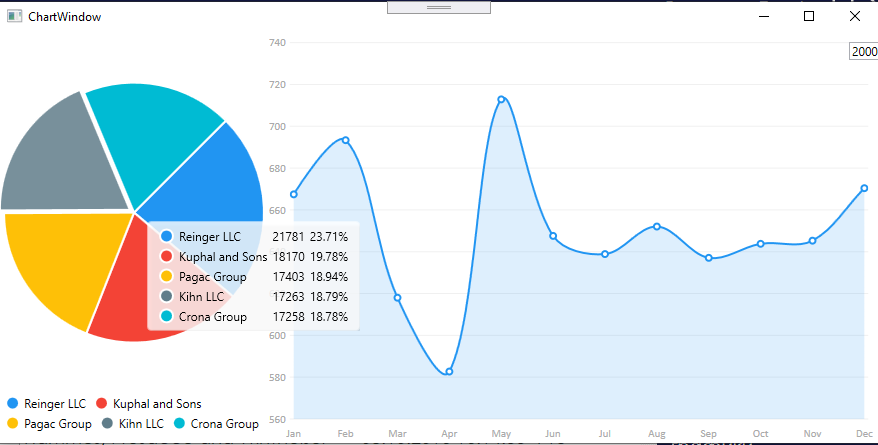


Рисунок Б.3 –Форма графиков

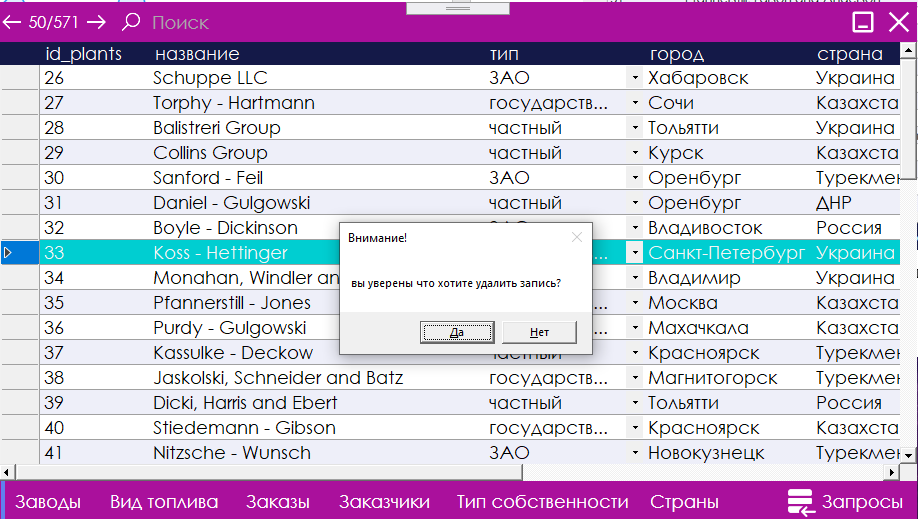


Рисунок Б.4– Всплывающее окно при удалении

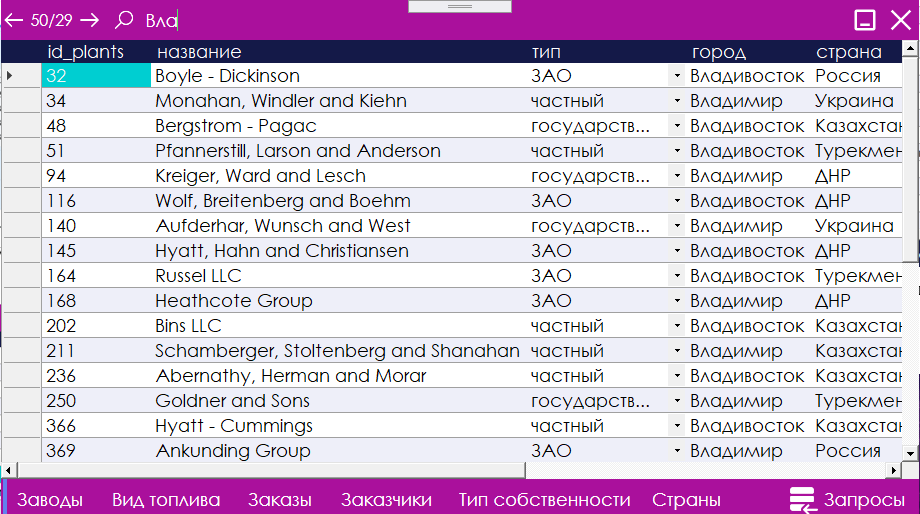


Рисунок Б.5 – Фильтрация данных

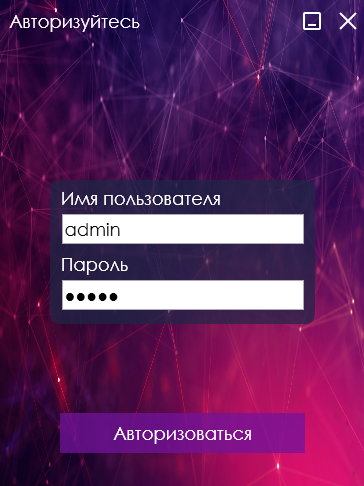


Рисунок Б.6 – Форма Login

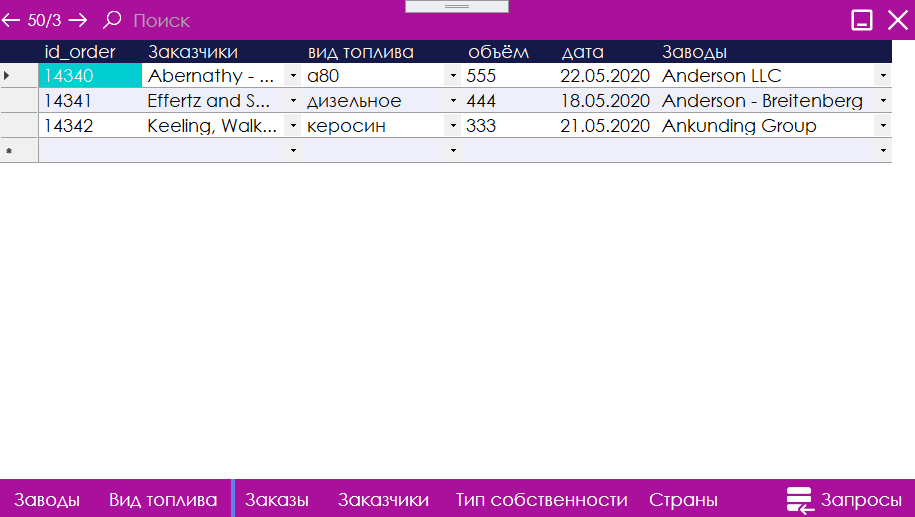


Рисунок Б.7 – Заказы, созданные пользователем Дима (RLS)

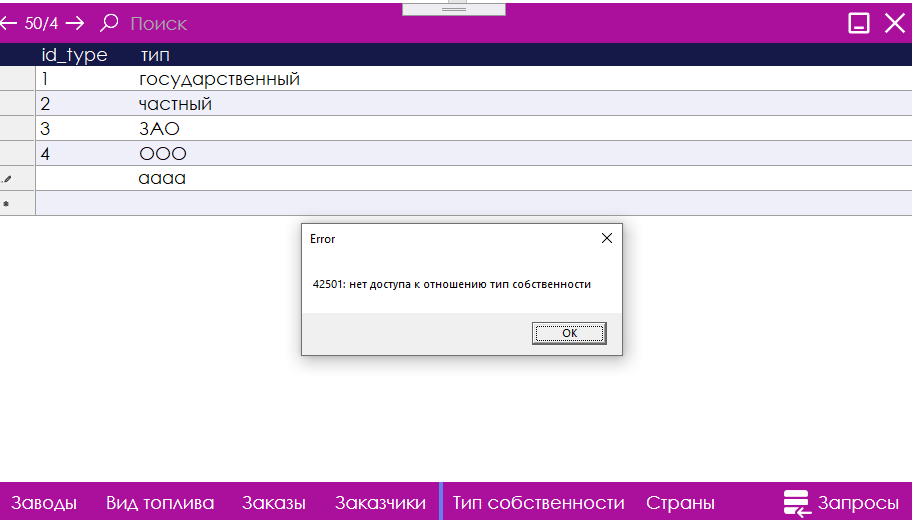


Рисунок Б.8 – Попытка пользователем Дима работы с запрещенной таблицей

# Приложение В ЛИСТИНГ СЕРВЕРНОГО ПРИЛОЖЕНИЯ

/\*

Navicat Premium Data Transfer

Source Server : Bober

Source Server Type : PostgreSQL

Source Server Version : 100011

Source Host : localhost:5432

Source Catalog : postgres

Source Schema : public

Target Server Type : PostgreSQL

Target Server Version : 100011

File Encoding : 65001

Date: 20/05/2020 18:45:35

\*/

-- ----------------------------

-- Sequence structure for Заводы\_id\_seq

-- ----------------------------

DROP SEQUENCE IF EXISTS "public"."Заводы\_id\_seq";

CREATE SEQUENCE "public"."Заводы\_id\_seq"

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

CACHE 1;

-- ----------------------------

-- Sequence structure for Заказчики\_id\_seq

-- ----------------------------

DROP SEQUENCE IF EXISTS "public"."Заказчики\_id\_seq";

CREATE SEQUENCE "public"."Заказчики\_id\_seq"

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

CACHE 1;

-- ----------------------------

-- Sequence structure for Заказы\_id\_seq

-- ----------------------------

DROP SEQUENCE IF EXISTS "public"."Заказы\_id\_seq";

CREATE SEQUENCE "public"."Заказы\_id\_seq"

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

CACHE 1;

-- ----------------------------

-- Sequence structure for вид топлива\_id\_seq

-- ----------------------------

DROP SEQUENCE IF EXISTS "public"."вид топлива\_id\_seq";

CREATE SEQUENCE "public"."вид топлива\_id\_seq"

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

CACHE 1;

-- ----------------------------

-- Sequence structure for страны\_id\_seq

-- ----------------------------

DROP SEQUENCE IF EXISTS "public"."страны\_id\_seq";

CREATE SEQUENCE "public"."страны\_id\_seq"

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

CACHE 1;

-- ----------------------------

-- Sequence structure for тип собственности\_id\_seq

-- ----------------------------

DROP SEQUENCE IF EXISTS "public"."тип собственности\_id\_seq";

CREATE SEQUENCE "public"."тип собственности\_id\_seq"

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

CACHE 1;

-- ----------------------------

-- Table structure for Заводы

-- ----------------------------

DROP TABLE IF EXISTS "public"."Заводы";

CREATE TABLE "public"."Заводы" (

"id\_plants" int4 NOT NULL GENERATED ALWAYS AS IDENTITY (

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

),

"название" varchar(255) COLLATE "pg\_catalog"."default" NOT NULL,

"город" varchar(255) COLLATE "pg\_catalog"."default" NOT NULL,

"id\_type" int4 NOT NULL,

"год" int4 NOT NULL,

"телефон" "public"."phone" COLLATE "pg\_catalog"."default" NOT NULL,

"id\_country" int4 NOT NULL,

"id\_fuel" int4 NOT NULL,

"объём" int4 NOT NULL,

"цена" int4 NOT NULL

)

;

-- ----------------------------

-- Table structure for Заказчики

-- ----------------------------

DROP TABLE IF EXISTS "public"."Заказчики";

CREATE TABLE "public"."Заказчики" (

"id\_client" int4 NOT NULL GENERATED ALWAYS AS IDENTITY (

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

),

"название" varchar(255) COLLATE "pg\_catalog"."default" NOT NULL,

"город" varchar(255) COLLATE "pg\_catalog"."default" NOT NULL,

"телефон" "public"."phone" COLLATE "pg\_catalog"."default" NOT NULL

)

;

-- ----------------------------

-- Table structure for Заказы

-- ----------------------------

DROP TABLE IF EXISTS "public"."Заказы";

CREATE TABLE "public"."Заказы" (

"id\_order" int4 NOT NULL GENERATED ALWAYS AS IDENTITY (

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

),

"id\_client" int4 NOT NULL,

"id\_fuel" int4 NOT NULL,

"объём" int4 NOT NULL,

"дата" timestamp(6) NOT NULL,

"id\_plants" int4 NOT NULL,

"login" varchar(64) COLLATE "pg\_catalog"."default" NOT NULL DEFAULT 'admin'::character varying

)

;

-- ----------------------------

-- Table structure for вид топлива

-- ----------------------------

DROP TABLE IF EXISTS "public"."вид топлива";

CREATE TABLE "public"."вид топлива" (

"id\_fuel" int4 NOT NULL GENERATED ALWAYS AS IDENTITY (

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

),

"топливо" varchar(255) COLLATE "pg\_catalog"."default" NOT NULL

)

;

-- ----------------------------

-- Table structure for страны

-- ----------------------------

DROP TABLE IF EXISTS "public"."страны";

CREATE TABLE "public"."страны" (

"id\_country" int4 NOT NULL GENERATED ALWAYS AS IDENTITY (

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

),

"страна" varchar(255) COLLATE "pg\_catalog"."default" NOT NULL

)

;

-- ----------------------------

-- Table structure for тип собственности

-- ----------------------------

DROP TABLE IF EXISTS "public"."тип собственности";

CREATE TABLE "public"."тип собственности" (

"id\_type" int4 NOT NULL GENERATED ALWAYS AS IDENTITY (

INCREMENT 1

MINVALUE 1

MAXVALUE 2147483647

START 1

),

"тип" varchar(255) COLLATE "pg\_catalog"."default" NOT NULL

)

;

-- ----------------------------

-- Function structure for avg\_orders

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."avg\_orders"();

CREATE OR REPLACE FUNCTION "public"."avg\_orders"()

RETURNS TABLE("объёмъ" numeric) AS $BODY$

BEGIN

RETURN QUERY SELECT

AVG("объём") as "объёмъ"

FROM "Заказы";

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for avg\_orders\_bigger

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."avg\_orders\_bigger"("\_n" int4, "\_m" int4);

CREATE OR REPLACE FUNCTION "public"."avg\_orders\_bigger"("\_n" int4, "\_m" int4)

RETURNS TABLE("объёмъ" numeric) AS $BODY$

BEGIN

RETURN QUERY SELECT

AVG("объём") as "объёмъ"

FROM "Заказы"

WHERE "объём" > \_n

HAVING AVG("объём") > \_m;

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for avg\_orders\_bigger

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."avg\_orders\_bigger"("\_n" int4);

CREATE OR REPLACE FUNCTION "public"."avg\_orders\_bigger"("\_n" int4)

RETURNS TABLE("объёмъ" numeric) AS $BODY$

BEGIN

RETURN QUERY SELECT

AVG("объём") as "объёмъ"

FROM "Заказы"

WHERE "объём" > \_N;

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for before\_insert\_orders\_fun

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."before\_insert\_orders\_fun"();

CREATE OR REPLACE FUNCTION "public"."before\_insert\_orders\_fun"()

RETURNS "pg\_catalog"."trigger" AS $BODY$

BEGIN

NEW.login = CURRENT\_USER;

RETURN NEW;

END;

$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

-- ----------------------------

-- Function structure for buyers\_last\_months

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."buyers\_last\_months"("\_months" int4);

CREATE OR REPLACE FUNCTION "public"."buyers\_last\_months"("\_months" int4)

RETURNS TABLE("название" varchar) AS $BODY$

BEGIN

RETURN QUERY SELECT "Заказчики"."название" FROM "Заказчики" WHERE id\_client IN (SELECT id\_client FROM "Заказы" WHERE "Заказы"."дата" > now() - (concat(\_months, ' months'))::interval)

ORDER BY "Заказчики"."название";

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for clients\_by\_last\_months

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."clients\_by\_last\_months"("\_months" int4);

CREATE OR REPLACE FUNCTION "public"."clients\_by\_last\_months"("\_months" int4)

RETURNS TABLE("название" varchar) AS $BODY$

BEGIN

RETURN QUERY SELECT "Заказчики"."название" FROM "Заказчики" WHERE id\_client IN (SELECT id\_client FROM "Заказы" WHERE "Заказы"."дата" > now() - (concat(\_months, ' months'))::interval)

ORDER BY "Заказчики"."название";

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for gen\_inc\_client

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."gen\_inc\_client"();

CREATE OR REPLACE FUNCTION "public"."gen\_inc\_client"()

RETURNS "pg\_catalog"."trigger" AS $BODY$BEGIN

NEW.id\_client = (SELECT MAX("Заказчики".id\_client)+1 FROM "Заказчики");

RETURN NEW;

END$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

-- ----------------------------

-- Function structure for gen\_inc\_country

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."gen\_inc\_country"();

CREATE OR REPLACE FUNCTION "public"."gen\_inc\_country"()

RETURNS "pg\_catalog"."trigger" AS $BODY$BEGIN

NEW.id\_country = (SELECT MAX("страны".id\_country)+1 FROM "страны");

RETURN NEW;

END$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

-- ----------------------------

-- Function structure for gen\_inc\_fuel

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."gen\_inc\_fuel"();

CREATE OR REPLACE FUNCTION "public"."gen\_inc\_fuel"()

RETURNS "pg\_catalog"."trigger" AS $BODY$BEGIN

NEW.id\_fuel = (SELECT MAX("вид топлива".id\_fuel)+1 FROM "вид топлива");

RETURN NEW;

END$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

-- ----------------------------

-- Function structure for gen\_inc\_orders

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."gen\_inc\_orders"();

CREATE OR REPLACE FUNCTION "public"."gen\_inc\_orders"()

RETURNS "pg\_catalog"."trigger" AS $BODY$BEGIN

NEW.id\_orders = (SELECT MAX("Заказы".id\_orders)+1 FROM "Заказы");

RETURN NEW;

END$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

-- ----------------------------

-- Function structure for gen\_inc\_plants

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."gen\_inc\_plants"();

CREATE OR REPLACE FUNCTION "public"."gen\_inc\_plants"()

RETURNS "pg\_catalog"."trigger" AS $BODY$BEGIN

NEW.id\_plants = (SELECT MAX("Заводы".id\_plants)+1 FROM "Заводы");

RETURN NEW;

END$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

-- ----------------------------

-- Function structure for gen\_inc\_type

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."gen\_inc\_type"();

CREATE OR REPLACE FUNCTION "public"."gen\_inc\_type"()

RETURNS "pg\_catalog"."trigger" AS $BODY$BEGIN

NEW.id\_type = (SELECT MAX("тип собственности".id\_type)+1 FROM "тип собственности");

RETURN NEW;

END$BODY$

LANGUAGE plpgsql VOLATILE

COST 100;

-- ----------------------------

-- Function structure for no\_buyers\_last\_months

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."no\_buyers\_last\_months"("\_months" int4);

CREATE OR REPLACE FUNCTION "public"."no\_buyers\_last\_months"("\_months" int4)

RETURNS TABLE("название" varchar) AS $BODY$

BEGIN

RETURN QUERY SELECT "Заказчики"."название" FROM "Заказчики" WHERE id\_client NOT IN (SELECT id\_client FROM "Заказы" WHERE "Заказы"."дата" > now() - (concat(\_months, ' months'))::interval)

ORDER BY "Заказчики"."название";

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for orders\_bigger\_V

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."orders\_bigger\_V"("\_count" int4);

CREATE OR REPLACE FUNCTION "public"."orders\_bigger\_V"("\_count" int4)

RETURNS TABLE("id\_order" int4, "объём" int4, "дата" timestamp, "топливо" varchar) AS $BODY$

BEGIN

RETURN QUERY SELECT "Заказы"."id\_order", "Заказы"."объём", "Заказы"."дата", "вид топлива"."топливо" FROM "Заказы" NATURAL JOIN "вид топлива" WHERE "Заказы"."объём" > \_count

ORDER BY "Заказы"."объём" DESC;

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for orders\_by\_last\_months

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."orders\_by\_last\_months"("\_months" int4);

CREATE OR REPLACE FUNCTION "public"."orders\_by\_last\_months"("\_months" int4)

RETURNS TABLE("id\_order" int4, "дата" timestamp, "объём" int4, "топливо" varchar) AS $BODY$

BEGIN

RETURN QUERY SELECT

"Заказы"."id\_order",

"Заказы"."дата",

"Заказы"."объём",

"вид топлива"."топливо"

FROM "Заказы"

NATURAL JOIN "вид топлива"

WHERE "Заказы"."дата" > now() - (concat(\_months, ' months'))::interval

ORDER BY "Заказы"."дата" DESC;

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for orders\_by\_last\_months\_index

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."orders\_by\_last\_months\_index"("\_months" int4);

CREATE OR REPLACE FUNCTION "public"."orders\_by\_last\_months\_index"("\_months" int4)

RETURNS TABLE("количество" int8) AS $BODY$

BEGIN

RETURN QUERY SELECT COUNT(\*) as "количество" FROM "Заказы" WHERE "Заказы"."дата" > now() - (concat(\_months, ' months'))::interval;

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for orders\_by\_last\_years

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."orders\_by\_last\_years"("\_years" int4);

CREATE OR REPLACE FUNCTION "public"."orders\_by\_last\_years"("\_years" int4)

RETURNS TABLE("id\_order" int4, "дата" timestamp, "объём" int4, "топливо" varchar) AS $BODY$

BEGIN

RETURN QUERY SELECT

"Заказы"."id\_order",

"Заказы"."дата",

"Заказы"."объём",

"вид топлива"."топливо"

FROM "Заказы"

NATURAL JOIN "вид топлива"

WHERE "Заказы"."дата" > now() - (concat(\_years, ' years'))::interval

ORDER BY "Заказы"."дата" DESC;

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for orders\_smaller\_V

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."orders\_smaller\_V"("\_count" int4);

CREATE OR REPLACE FUNCTION "public"."orders\_smaller\_V"("\_count" int4)

RETURNS TABLE("id\_order" int4, "объём" int4, "дата" timestamp, "топливо" varchar) AS $BODY$

BEGIN

RETURN QUERY SELECT "Заказы"."id\_order", "Заказы"."объём", "Заказы"."дата", "вид топлива"."топливо" FROM "Заказы" NATURAL JOIN "вид топлива" WHERE "Заказы"."объём" < \_count

ORDER BY "Заказы"."объём" DESC;

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for orders\_sum\_where\_V\_bigger

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."orders\_sum\_where\_V\_bigger"("\_n" int4);

CREATE OR REPLACE FUNCTION "public"."orders\_sum\_where\_V\_bigger"("\_n" int4)

RETURNS TABLE("объём" int8) AS $BODY$

BEGIN

RETURN QUERY SELECT sum("Заказы"."объём") FROM "Заказы" WHERE "Заказы"."объём" > \_n;

END$BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for orders\_sum\_where\_V\_fuel\_like

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."orders\_sum\_where\_V\_fuel\_like"("\_fuel" varchar);

CREATE OR REPLACE FUNCTION "public"."orders\_sum\_where\_V\_fuel\_like"("\_fuel" varchar)

RETURNS TABLE("объём" int8) AS $BODY$

BEGIN

RETURN QUERY SELECT sum("Заказы"."объём") FROM "Заказы" NATURAL JOIN "вид топлива" WHERE "вид топлива"."топливо" like concat('%', \_fuel,'%');

END$BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for sum\_zakazi\_by\_client

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."sum\_zakazi\_by\_client"("\_count" int8);

CREATE OR REPLACE FUNCTION "public"."sum\_zakazi\_by\_client"("\_count" int8)

RETURNS TABLE("Клиент" varchar, "сумма" int8) AS $BODY$

BEGIN

--Последние заказы фирмы ' название фирмы'

RETURN QUERY SELECT "Заказчики"."название" as "Клиент", SUM(объём) as "сумма" FROM "Заказчики"

NATURAL JOIN "Заказы"

GROUP BY название

HAVING SUM(объём) > \_count

ORDER BY "сумма" DESC;

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- Function structure for zakazi\_avg\_by\_month\_year

-- ----------------------------

DROP FUNCTION IF EXISTS "public"."zakazi\_avg\_by\_month\_year"("\_month" int4, "\_year" int4);

CREATE OR REPLACE FUNCTION "public"."zakazi\_avg\_by\_month\_year"("\_month" int4, "\_year" int4)

RETURNS TABLE("ср.цена" numeric) AS $BODY$

BEGIN

RETURN QUERY SELECT ROUND(AVG("Заказы"."объём"), 2) as "ср.цена" FROM "Заказы"

WHERE EXTRACT(YEAR from дата) = \_year AND EXTRACT(MONTH from дата) = \_month;

END $BODY$

LANGUAGE plpgsql VOLATILE

COST 100

ROWS 1000;

-- ----------------------------

-- View structure for Заказчики\_без\_заказов

-- ----------------------------

DROP VIEW IF EXISTS "public"."Заказчики\_без\_заказов";

CREATE VIEW "public"."Заказчики\_без\_заказов" AS SELECT "Заказчики"."название",

"Заказчики"."город"

FROM ("Заказы"

RIGHT JOIN "Заказчики" ON (("Заказы".id\_client = "Заказчики".id\_client)))

WHERE ("Заказы".id\_order IS NULL);

-- ----------------------------

-- View structure for Заводы\_без\_заказов

-- ----------------------------

DROP VIEW IF EXISTS "public"."Заводы\_без\_заказов";

CREATE VIEW "public"."Заводы\_без\_заказов" AS SELECT "Заводы"."название"

FROM ("Заводы"

LEFT JOIN "Заказы" ON (("Заказы".id\_plants = "Заводы".id\_plants)))

WHERE ("Заказы".id\_order IS NULL);

-- ----------------------------

-- View structure for emulate\_left\_join

-- ----------------------------

DROP VIEW IF EXISTS "public"."emulate\_left\_join";

CREATE VIEW "public"."emulate\_left\_join" AS SELECT "Заказчики"."название" AS "заказчик",

( SELECT "Заказы".id\_order

FROM "Заказы"

WHERE ("Заказы".id\_client = "Заказчики".id\_client)

LIMIT 1) AS "заказы"

FROM "Заказчики"

WHERE (( SELECT "Заказы".id\_order

FROM "Заказы"

WHERE ("Заказы".id\_client = "Заказчики".id\_client)

LIMIT 1) IS NULL);

-- ----------------------------

-- View structure for podzapros\_itog

-- ----------------------------

DROP VIEW IF EXISTS "public"."podzapros\_itog";

CREATE VIEW "public"."podzapros\_itog" AS SELECT bb."название",

( SELECT round(avg(aa."объём"), 2) AS round

FROM ("Заказы" aa

JOIN "Заказчики" cc USING (id\_client))

WHERE ((cc."название")::text = (bb."название")::text)) AS "среднее",

bb."телефон"

FROM "Заказчики" bb

ORDER BY bb."название";

-- ----------------------------

-- View structure for union\_view

-- ----------------------------

DROP VIEW IF EXISTS "public"."union\_view";

CREATE VIEW "public"."union\_view" AS SELECT "вид топлива".id\_fuel,

"вид топлива"."топливо"

FROM "вид топлива"

UNION

SELECT "тип собственности".id\_type AS id\_fuel,

"тип собственности"."тип" AS "топливо"

FROM "тип собственности";

-- ----------------------------

-- View structure for заказы\_natural\_join

-- ----------------------------

DROP VIEW IF EXISTS "public"."заказы\_natural\_join";

CREATE VIEW "public"."заказы\_natural\_join" AS SELECT "Заказы"."объём",

"Заказы"."дата",

"Заказы".id\_order,

"Заводы"."название",

"Заводы"."город",

"вид топлива"."топливо",

"Заводы"."цена"

FROM (("Заказы"

JOIN "вид топлива" ON (("Заказы".id\_fuel = "вид топлива".id\_fuel)))

JOIN "Заводы" ON ((("Заводы".id\_fuel = "вид топлива".id\_fuel) AND ("Заказы".id\_plants = "Заводы".id\_plants))));

-- ----------------------------

-- View structure for заводы\_natural\_join

-- ----------------------------

DROP VIEW IF EXISTS "public"."заводы\_natural\_join";

CREATE VIEW "public"."заводы\_natural\_join" AS SELECT "вид топлива"."топливо",

"Заводы"."название",

"Заводы"."город",

"Заводы"."объём",

"Заводы"."год",

"Заводы"."телефон",

"Заводы"."цена",

"страны"."страна"

FROM (("Заводы"

JOIN "вид топлива" ON (("Заводы".id\_fuel = "вид топлива".id\_fuel)))

JOIN "страны" ON (("Заводы".id\_country = "страны".id\_country)));

-- ----------------------------

-- View structure for заказчики\_natural\_join

-- ----------------------------

DROP VIEW IF EXISTS "public"."заказчики\_natural\_join";

CREATE VIEW "public"."заказчики\_natural\_join" AS SELECT "Заказчики"."название",

"Заказы"."дата",

"Заказы"."объём"

FROM ("Заказчики"

JOIN "Заказы" ON (("Заказы".id\_client = "Заказчики".id\_client)));

-- ----------------------------

-- View structure for avg\_V\_orders

-- ----------------------------

DROP VIEW IF EXISTS "public"."avg\_V\_orders";

CREATE VIEW "public"."avg\_V\_orders" AS SELECT avg("Заказы"."объём") AS "ср.объём"

FROM "Заказы";

-- ----------------------------

-- View structure for sum\_V\_orders

-- ----------------------------

DROP VIEW IF EXISTS "public"."sum\_V\_orders";

CREATE VIEW "public"."sum\_V\_orders" AS SELECT sum("Заказы"."объём") AS "сумма"

FROM "Заказы";

-- ----------------------------

-- View structure for case\_view

-- ----------------------------

DROP VIEW IF EXISTS "public"."case\_view";

CREATE VIEW "public"."case\_view" AS SELECT a."название",

a."дата",

CASE

WHEN (a."дата" > (now() - '1 mon'::interval)) THEN 'недавно'::text

ELSE 'давно'::text

END AS "case"

FROM ("Заказчики"

JOIN "Заказы" USING (id\_client)) a

WHERE (a."дата" = ( SELECT max(b."дата") AS max

FROM "Заказы" b

WHERE (a.id\_client = b.id\_client)))

ORDER BY a."дата" DESC;

-- ----------------------------

-- Alter sequences owned by

-- ----------------------------

ALTER SEQUENCE "public"."Заводы\_id\_seq"

OWNED BY "public"."Заводы"."id\_plants";

SELECT setval('"public"."Заводы\_id\_seq"', 626, true);

ALTER SEQUENCE "public"."Заказчики\_id\_seq"

OWNED BY "public"."Заказчики"."id\_client";

SELECT setval('"public"."Заказчики\_id\_seq"', 1415, true);

ALTER SEQUENCE "public"."Заказы\_id\_seq"

OWNED BY "public"."Заказы"."id\_order";

SELECT setval('"public"."Заказы\_id\_seq"', 14343, true);

ALTER SEQUENCE "public"."вид топлива\_id\_seq"

OWNED BY "public"."вид топлива"."id\_fuel";

SELECT setval('"public"."вид топлива\_id\_seq"', 8, true);

ALTER SEQUENCE "public"."страны\_id\_seq"

OWNED BY "public"."страны"."id\_country";

SELECT setval('"public"."страны\_id\_seq"', 8, true);

ALTER SEQUENCE "public"."тип собственности\_id\_seq"

OWNED BY "public"."тип собственности"."id\_type";

SELECT setval('"public"."тип собственности\_id\_seq"', 7, true);

-- ----------------------------

-- Uniques structure for table Заводы

-- ----------------------------

ALTER TABLE "public"."Заводы" ADD CONSTRAINT "Заводы\_название\_key" UNIQUE ("название");

-- ----------------------------

-- Primary Key structure for table Заводы

-- ----------------------------

ALTER TABLE "public"."Заводы" ADD CONSTRAINT "Заводы\_pkey" PRIMARY KEY ("id\_plants");

-- ----------------------------

-- Primary Key structure for table Заказчики

-- ----------------------------

ALTER TABLE "public"."Заказчики" ADD CONSTRAINT "Заказчики\_pkey" PRIMARY KEY ("id\_client");

-- ----------------------------

-- Indexes structure for table Заказы

-- ----------------------------

CREATE INDEX "date\_index" ON "public"."Заказы" USING btree (

"дата" "pg\_catalog"."timestamp\_ops" ASC NULLS LAST

);

-- ----------------------------

-- Triggers structure for table Заказы

-- ----------------------------

CREATE TRIGGER "before\_insert\_orders" BEFORE INSERT ON "public"."Заказы"

FOR EACH ROW

EXECUTE PROCEDURE "public"."before\_insert\_orders\_fun"();

-- ----------------------------

-- Primary Key structure for table Заказы

-- ----------------------------

ALTER TABLE "public"."Заказы" ADD CONSTRAINT "Заказы\_pkey" PRIMARY KEY ("id\_order");

-- ----------------------------

-- Primary Key structure for table вид топлива

-- ----------------------------

ALTER TABLE "public"."вид топлива" ADD CONSTRAINT "вид топлива\_pkey" PRIMARY KEY ("id\_fuel");

-- ----------------------------

-- Primary Key structure for table страны

-- ----------------------------

ALTER TABLE "public"."страны" ADD CONSTRAINT "страны\_pkey" PRIMARY KEY ("id\_country");

-- ----------------------------

-- Primary Key structure for table тип собственности

-- ----------------------------

ALTER TABLE "public"."тип собственности" ADD CONSTRAINT "тип собственности\_pkey" PRIMARY KEY ("id\_type");

-- ----------------------------

-- Foreign Keys structure for table Заводы

-- ----------------------------

ALTER TABLE "public"."Заводы" ADD CONSTRAINT "Заводы\_id\_country\_fkey" FOREIGN KEY ("id\_country") REFERENCES "public"."страны" ("id\_country") ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE "public"."Заводы" ADD CONSTRAINT "Заводы\_id\_fuel\_fkey" FOREIGN KEY ("id\_fuel") REFERENCES "public"."вид топлива" ("id\_fuel") ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE "public"."Заводы" ADD CONSTRAINT "Заводы\_id\_type\_fkey" FOREIGN KEY ("id\_type") REFERENCES "public"."тип собственности" ("id\_type") ON DELETE NO ACTION ON UPDATE NO ACTION;

-- ----------------------------

-- Foreign Keys structure for table Заказы

-- ----------------------------

ALTER TABLE "public"."Заказы" ADD CONSTRAINT "Заказы\_id\_client\_fkey" FOREIGN KEY ("id\_client") REFERENCES "public"."Заказчики" ("id\_client") ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE "public"."Заказы" ADD CONSTRAINT "Заказы\_id\_fuel\_fkey" FOREIGN KEY ("id\_fuel") REFERENCES "public"."вид топлива" ("id\_fuel") ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE "public"."Заказы" ADD CONSTRAINT "Заказы\_id\_plants\_fkey" FOREIGN KEY ("id\_plants") REFERENCES "public"."Заводы" ("id\_plants") ON DELETE NO ACTION ON UPDATE NO ACTION;

Приложение Г ЛИСТИНГ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ

Код

public partial class LoginWindow : Window

{

public LoginWindow()

{

InitializeComponent();

}

private void Move\_Window(object sender, MouseButtonEventArgs e)

{

if (e.ChangedButton == MouseButton.Left)

this.DragMove();

}

private void btn\_exit\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void btn\_minimize\_Click(object sender, RoutedEventArgs e)

{

this.WindowState = WindowState.Minimized;

}

private void btn\_authorization\_Click(object sender, RoutedEventArgs e)

{

string database = "course2020";

string port = "5432";

string host = "127.0.0.1";

string login = txb\_login.Text;

string password = txb\_password.Password;

string connection\_string = $"Server={host}; Port={port}; User Id={login}; Password={password}; Database={database}";

NpgsqlConnection connection = new NpgsqlConnection(connection\_string);

try

{

connection.Open();

}

catch (Npgsql.PostgresException ex)

{

MessageBox.Show("Ошибка авторизации!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}//#FF3E59A0

MainWindow window = new MainWindow(connection\_string, login);

window.Show();

this.Close();

}

} public partial class QueryWindow : Window

{

string connection\_string;

public QueryWindow(string connection\_string)

{

InitializeComponent();

this.connection\_string = connection\_string;

dataGridView1.BorderStyle = System.Windows.Forms.BorderStyle.None;

dataGridView1.AlternatingRowsDefaultCellStyle.BackColor = System.Drawing.Color.FromArgb(238, 239, 249);

dataGridView1.CellBorderStyle = System.Windows.Forms.DataGridViewCellBorderStyle.SingleHorizontal;

dataGridView1.DefaultCellStyle.SelectionBackColor = System.Drawing.Color.DarkTurquoise;

dataGridView1.DefaultCellStyle.SelectionForeColor = System.Drawing.Color.WhiteSmoke;

dataGridView1.BackgroundColor = System.Drawing.Color.White;

dataGridView1.EnableHeadersVisualStyles = false;

dataGridView1.ColumnHeadersBorderStyle = System.Windows.Forms.DataGridViewHeaderBorderStyle.None;

dataGridView1.ColumnHeadersDefaultCellStyle.BackColor = System.Drawing.Color.FromArgb(20, 25, 72);

dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = System.Drawing.Color.White;

}

private void Move\_Window(object sender, MouseButtonEventArgs e)

{

if (e.ChangedButton == MouseButton.Left)

this.DragMove();

}

private void btn\_exit\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void move\_rect(object sender)

{

//rect.Margin = new Thickness((sender as Button).Margin.Left, (sender as Button).Margin.Top, 0, 0);

//rect.Height = (sender as Button).Height;

}

private void btn\_minimize\_Click(object sender, RoutedEventArgs e)

{

this.WindowState = WindowState.Minimized;

}

void FillGridFromDataSet(DataSet dataSet)

{

try

{

dataGridView1.Columns.Clear();

for (int i = 0; i < dataSet.Tables[0].Columns.Count; i++)

{

var col = dataSet.Tables[0].Columns[i];

dataGridView1.Columns.Add(col.ColumnName, col.Caption);

if (i == 0) dataGridView1.Columns[col.ColumnName].ReadOnly = true;

}

for (int i = 0; i < dataSet.Tables[0].Rows.Count; i++)

dataGridView1.Rows.Add(dataSet.Tables[0].Rows[i].ItemArray);

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

dataGridView1.Columns[i].AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells;

if (i == dataGridView1.Columns.Count - 1)

dataGridView1.Columns[i].AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeColumnMode.Fill;

}

}

catch (Exception e)

{

}

}

DataSet FillDataSetFromDB(string sql)

{

try

{

NpgsqlConnection connection = new NpgsqlConnection(connection\_string);

var select = sql;

var dataAdapter = new NpgsqlDataAdapter(select, connection);

var dataSet = new DataSet();

dataAdapter.Fill(dataSet);

connection.Close();

return dataSet;

}

catch (Exception e)

{

MessageBox.Show(e.Message, "Error");

return null;

}

}

private void btn\_orders\_Click(object sender, RoutedEventArgs e)

{

string sql = "SELECT \* FROM Заказы\_naturaljoin";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_orders\_by\_firm\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("Фирма", "Введите фирму");

if (dialog.ShowDialog() == true)

{

string sql = $"SELECT \* FROM zakazi\_by\_firm('{dialog.ResponseText}')";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

}

private void btn\_hospitals\_Click(object sender, RoutedEventArgs e)

{

string sql = "SELECT \* FROM Заводы\_naturaljoin";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_orders\_by\_hospital\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("Больница", "Введите Номер заводы");

if (dialog.ShowDialog() == true)

{

try

{

string sql = $"SELECT \* FROM zakazi\_by\_hospital({dialog.ResponseText})";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

catch (Exception ee)

{

MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void btn\_orders\_by\_last\_months\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("Месяцы", "Введите за сколько последних месяцев вернуть заказы");

if (dialog.ShowDialog() == true)

{

try

{

string sql = $"SELECT \* FROM zakazi\_by\_last\_months({dialog.ResponseText})";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

catch (Exception ee)

{

MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void btn\_orders\_by\_last\_years\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("Месяцы", "Введите за сколько последних лет вернуть заказы");

if (dialog.ShowDialog() == true)

{

try

{

string sql = $"SELECT \* FROM zakazi\_by\_last\_years({dialog.ResponseText})";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

catch (Exception ee)

{

MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void btn\_hospoitals\_wiithout\_orders\_Click(object sender, RoutedEventArgs e)

{

string sql = "SELECT \* FROM view\_Заводы\_без\_заказов";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_firms\_without\_orders\_Click(object sender, RoutedEventArgs e)

{

string sql = "SELECT \* FROM view\_фирмы\_без\_заказов";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_firms\_without\_orders\_left\_join\_Click(object sender, RoutedEventArgs e)

{

string sql = "SELECT \* FROM view\_emulate\_left\_join";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_avg\_orders\_by\_hospiatl\_Click(object sender, RoutedEventArgs e)

{

string sql = "SELECT \* FROM средняя\_стоимость\_заказа\_для\_кажд";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_sum\_orders\_Click(object sender, RoutedEventArgs e)

{

string sql = "SELECT \* FROM сумма\_заказов";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_avg\_bigger\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("Цена", "Введите больше чего");

if (dialog.ShowDialog() == true)

{

try

{

string sql = $"SELECT \* FROM zakazi\_avg\_bigger({dialog.ResponseText})";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

catch (Exception ee)

{

MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void btn\_firms\_in\_country\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("Страна", "Введите маску страны");

if (dialog.ShowDialog() == true)

{

string sql = $"SELECT \* FROM firms\_country\_like('{dialog.ResponseText}')";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

}

private void btn\_count\_firms\_by\_last\_monthes\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("Месяцы", "Введите за сколько последних месяцев вернуть заказы");

if (dialog.ShowDialog() == true)

{

try

{

string sql = $"SELECT \* FROM zakazi\_count\_by\_last\_month({dialog.ResponseText})";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

catch (Exception ee)

{

MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void btn\_sum\_zakazi\_by\_firm\_where\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("количество", "Введите количество товаров для данных");

if (dialog.ShowDialog() == true)

{

try

{

string sql = $"SELECT \* FROM sum\_zakazi\_by\_firm\_where({dialog.ResponseText})";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

catch (Exception ee)

{

MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void btn\_sum\_zakazi\_by\_firm\_where2\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("количество", "Введите количество товаров для данных");

var dialog2 = new MyDialog("количество", "Введите количество товаров для группы");

if (dialog.ShowDialog() == true && dialog2.ShowDialog() == true)

{

try

{

string sql = $"SELECT \* FROM sum\_zakazi\_by\_firm\_where({dialog.ResponseText}, {dialog2.ResponseText})";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

catch (Exception ee)

{

MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void btn\_itog\_podzapros\_Click(object sender, RoutedEventArgs e)

{

string sql = $"SELECT \* FROM podzapros\_itog()";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_union\_Click(object sender, RoutedEventArgs e)

{

string sql = $"SELECT \* FROM union\_view";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_in\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("Месяцы", "Введите интервал в месяцах");

if (dialog.ShowDialog() == true)

{

try

{

string sql = $"SELECT \* FROM hospitals\_by\_last\_months({dialog.ResponseText})";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

catch (Exception ee)

{

MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void btn\_not\_in\_Click(object sender, RoutedEventArgs e)

{

var dialog = new MyDialog("Месяцы", "Введите интервал в месяцах");

if (dialog.ShowDialog() == true)

{

try

{

string sql = $"SELECT \* FROM no\_hospitals\_by\_last\_months({dialog.ResponseText})";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

catch (Exception ee)

{

MessageBox.Show("Вы ввели не число", "Error", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

private void btn\_case\_Click(object sender, RoutedEventArgs e)

{

string sql = $"SELECT \* FROM case\_view";

FillGridFromDataSet(FillDataSetFromDB(sql));

}

private void btn\_excel\_Click(object sender, RoutedEventArgs e)

{

//Creating DataTable

DataTable dt = new DataTable();

//Adding the Columns

foreach (System.Windows.Forms.DataGridViewColumn column in dataGridView1.Columns)

{

dt.Columns.Add(column.HeaderText);

}

//Adding the Rows

foreach (System.Windows.Forms.DataGridViewRow row in dataGridView1.Rows)

{

dt.Rows.Add();

foreach (System.Windows.Forms.DataGridViewCell cell in row.Cells)

{

if (cell.Value != null)

dt.Rows[dt.Rows.Count - 1][cell.ColumnIndex] = cell.Value.ToString();

}

}

SaveFileDialog saveFileDialog1 = new SaveFileDialog();

saveFileDialog1.Filter = "xlsx files (\*.xlsx)|\*.xlsx|All files (\*.\*)|\*.\*";

saveFileDialog1.FilterIndex = 1;

saveFileDialog1.RestoreDirectory = true;

if (saveFileDialog1.ShowDialog() == true)

{

using (XLWorkbook wb = new XLWorkbook())

{

wb.Worksheets.Add(dt, "Sheet");

wb.SaveAs(saveFileDialog1.FileName);

MessageBox.Show("Сохранено");

}

}

}

private void btn\_charts\_Click(object sender, RoutedEventArgs e)

{

ChartWindow cw = new ChartWindow(connection\_string);

cw.Show();

}

}}

using System;

using System.Collections.Generic;

using System.Data;

using System.Drawing;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using Npgsql;

namespace BDcource2020

{

/// <summary>

/// Interaction logic for MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

string connection\_string;

string login;

public MainWindow(string connection\_string, string login)

{

InitializeComponent();

this.connection\_string = connection\_string;

this.login = login;

set\_TextBox(tb\_search, "Search");

dataGridView1.Controls.Add(dtp);

dtp.Visible = false;

dtp.Format = System.Windows.Forms.DateTimePickerFormat.Custom;

dtp.TextChanged += new EventHandler(dtp\_TextChange);

dataGridView1.BorderStyle = System.Windows.Forms.BorderStyle.None;

dataGridView1.AlternatingRowsDefaultCellStyle.BackColor = Color.FromArgb(238, 239, 249);

dataGridView1.CellBorderStyle = System.Windows.Forms.DataGridViewCellBorderStyle.SingleHorizontal;

dataGridView1.DefaultCellStyle.SelectionBackColor = Color.DarkTurquoise;

dataGridView1.DefaultCellStyle.SelectionForeColor = Color.WhiteSmoke;

dataGridView1.BackgroundColor = Color.White;

dataGridView1.EnableHeadersVisualStyles = false;

dataGridView1.ColumnHeadersBorderStyle = System.Windows.Forms.DataGridViewHeaderBorderStyle.None;

dataGridView1.ColumnHeadersDefaultCellStyle.BackColor = Color.FromArgb(20, 25, 72);

dataGridView1.ColumnHeadersDefaultCellStyle.ForeColor = Color.White;

btn\_hospital\_Click(btn\_hospital, null);

new Generator();

}

private void Move\_Window(object sender, MouseButtonEventArgs e)

{

if (e.ChangedButton == MouseButton.Left)

this.DragMove();

}

private void set\_TextBox(TextBox tb, string text = "")

{

var converter = new System.Windows.Media.BrushConverter();

tb.Text = text;

tb.GotKeyboardFocus += new KeyboardFocusChangedEventHandler(tb\_GotKeyboardFocus);

tb.LostKeyboardFocus += new KeyboardFocusChangedEventHandler(tb\_LostKeyboardFocus);

}

bool enabletxb = false;

private void tb\_GotKeyboardFocus(object sender, KeyboardFocusChangedEventArgs e)

{

if (sender is TextBox)

{

//If nothing has been entered yet.

var converter = new System.Windows.Media.BrushConverter();

if (enabletxb == false)

{

((TextBox)sender).Text = "";

((TextBox)sender).Foreground = System.Windows.Media.Brushes.White;

}

}

}

private void tb\_LostKeyboardFocus(object sender, KeyboardFocusChangedEventArgs e)

{

if (sender is TextBox)

{

enabletxb = false;

//If nothing was entered, reset default text.

if (((TextBox)sender).Text.Trim().Equals(""))

{

var converter = new System.Windows.Media.BrushConverter();

((TextBox)sender).Foreground = (System.Windows.Media.Brush)converter.ConvertFromString("#FFAAAAAA");//K

if (((TextBox)sender).Name == "tb\_search")

((TextBox)sender).Text = "Search";

else

((TextBox)sender).Text = "DefaultText";

}

}

}

void FillGridFromDataSet(DataSet dataSet)

{

try

{

dataGridView1.Columns.Clear();

for (int i = 0; i < dataSet.Tables[0].Columns.Count; i++)

{

var col = dataSet.Tables[0].Columns[i];

dataGridView1.Columns.Add(col.ColumnName, col.Caption);

if(i==0) dataGridView1.Columns[col.ColumnName].ReadOnly = true;

}

for (int i = 0; i < dataSet.Tables[0].Rows.Count; i++)

dataGridView1.Rows.Add(dataSet.Tables[0].Rows[i].ItemArray);

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

dataGridView1.Columns[i].AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells;

if (i == dataGridView1.Columns.Count - 1)

dataGridView1.Columns[i].AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeColumnMode.Fill;

}

}

catch (Exception e)

{

}

}

DataSet FillDataSetFromDB(string sql)

{

try

{

NpgsqlConnection connection = new NpgsqlConnection(connection\_string);

var select = sql;

var dataAdapter = new NpgsqlDataAdapter(select, connection);

var dataSet = new DataSet();

dataAdapter.Fill(dataSet);

connection.Close();

return dataSet;

}

catch (Exception e)

{

MessageBox.Show(e.Message, "Error");

last\_click(last\_sender, null);

return null;

}

}

void execSql(string sql)

{

try

{

NpgsqlConnection connection = new NpgsqlConnection(connection\_string);

connection.Open();

NpgsqlCommand com = new NpgsqlCommand(sql, connection);

com.ExecuteNonQuery();

connection.Close();

}

catch (Exception e)

{

MessageBox.Show(e.Message, "Error");

last\_click(last\_sender, null);

}

}

public DataTable fillComboBox(string fields, string table, string display\_member)

{

NpgsqlConnection connection = new NpgsqlConnection(connection\_string);

var select = "SELECT " + fields + " FROM \"" + table + "\" ORDER BY " + display\_member;

var dataAdapter = new NpgsqlDataAdapter(select, connection);

var dataTable = new DataTable();

dataAdapter.Fill(dataTable);

connection.Close();

return dataTable;

}

/// <summary>

/// добавить ComboBox в таблицу

/// </summary>

/// <param name="name\_cmbCol">имя колонки в таблице</param>

/// <param name="headerText">отображение колонки</param>

/// <param name="table">из какой таблицы брать данные</param>

/// <param name="ValueMember">какой член table брать за индекс</param>

/// <param name="DisplayMember">какой член table показывать</param>

/// <param name="Xindex">каким по порядку разместить</param>

void addComboBoxColumn(string name\_cmbCol, string headerText, string table,

string ValueMember, string DisplayMember, int Xindex)

{

System.Windows.Forms.DataGridViewComboBoxColumn cmbCol = new System.Windows.Forms.DataGridViewComboBoxColumn();

cmbCol.HeaderText = headerText;

cmbCol.Name = name\_cmbCol;

string fields = $"\*";

cmbCol.DataSource = fillComboBox(fields, table, DisplayMember);

cmbCol.ValueMember = ValueMember;

cmbCol.DisplayMember = DisplayMember;

cmbCol.FlatStyle = System.Windows.Forms.FlatStyle.Flat;

dataGridView1.Columns.Add(cmbCol);

dataGridView1.Columns[name\_cmbCol].DisplayIndex = Xindex;

dataGridView1.Columns[Xindex].AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells;

}

void setComboBoxColumn(string column, string column\_with\_id)

{

foreach (System.Windows.Forms.DataGridViewRow row in dataGridView1.Rows)

{

row.Cells[column].Value = row.Cells[column\_with\_id].Value;

}

}

public void StretchLastColumn()

{

var lastColIndex = dataGridView1.Columns.Count - 1;

var lastCol = dataGridView1.Columns[lastColIndex];

lastCol.AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeColumnMode.Fill;

}

void fix\_cmb\_width()

{

if (dataGridView1.Columns.Count != 0)

{

dataGridView1.Columns[0].ReadOnly = true;

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

var sub = dataGridView1.Columns[i].Name.Substring(0, 3);

if (sub == "cmb")

{

dataGridView1.Columns[i].Width = 160;

}

else if (sub == "chk")

{

}

else if (dataGridView1.Columns[i].Name == "дата")

{

dataGridView1.Columns[i].Width = 130;

}

else

{

dataGridView1.Columns[i].AutoSizeMode = System.Windows.Forms.DataGridViewAutoSizeColumnMode.AllCells;

}

}

}

}

string generate\_where()

{

string where = "";

List<string> likes = new List<string>();

for (int i = 0; i < fields.Length; i++)

if (fields\_types[i] == "string")

likes.Add($"{fields[i]} LIKE '%{search}%'");

if (likes.Count == 0 || search == "Search" || search == "")

where = "1=1";

else

{

bool first = true;

for (int i = 0; i < likes.Count; i++)

{

if (first)

{

where += likes[i];

first = false;

}

else

where += " OR " + likes[i];

}

}

return where;

}

string generate\_select(string[] fields, string table)

{

string where = generate\_where();

string result = "SELECT ";

foreach(string field in fields){

result += "\"" +field + "\",";

}

result = result.Remove(result.Length - 1);

result += $"FROM \"{table}\" ";

result += $"WHERE {where} ";

result += $"ORDER BY {fields[0]} ";

result += $"LIMIT {(page + 1) \* 50} ";

result += $"OFFSET {page \* 50} ";

int count = getCountRowsTable(table, where);

status\_text.Text = " " + ((page + 1) \* 50).ToString();

status\_text.Text += "/" + count + " ";

return result;

}

private void btn\_exit\_Click(object sender, RoutedEventArgs e)

{

this.Close();

}

private void move\_rect(object sender)

{

rect.Margin = new Thickness((sender as Button).Margin.Left, (sender as Button).Margin.Top, 0, 0);

rect.Height = (sender as Button).Height;

}

private void btn\_minimize\_Click(object sender, RoutedEventArgs e)

{

this.WindowState = WindowState.Minimized;

}

string[] fields;

string[] fields\_types;

string table;

delegate void btn\_click(object sender, RoutedEventArgs e);

btn\_click last\_click;

object last\_sender;

private void btn\_hospital\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_hospital\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Заводы";

fields = new string[]

{

"id\_hospital", "номер", "id\_type", "id\_district", "год создания", "число мест", "количество врачей", "телефон"

};

fields\_types = new string[]

{

"int", "int", "int", "int", "int", "int", "int", "string"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

dataGridView1.Columns["id\_type"].Visible = false;

dataGridView1.Columns["id\_district"].Visible = false;

addComboBoxColumn("cmb\_type", "тип", "Типы заводы", "id\_type", "тип", 2);

addComboBoxColumn("cmb\_district", "район", "Районы города", "id\_district", "район", 3);

setComboBoxColumn("cmb\_type", "id\_type");

setComboBoxColumn("cmb\_district", "id\_district");

fix\_cmb\_width();

}

private void btn\_orders\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_orders\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Заказы";

fields = new string[]

{

"id\_order", "id\_firm", "id\_group", "id\_form", "дата", "количество", "цена", "id\_hospital"

};

fields\_types = new string[]

{

"int", "int", "int", "int", "string", "int", "int", "int"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

dataGridView1.Columns["id\_firm"].Visible = false;

dataGridView1.Columns["id\_group"].Visible = false;

dataGridView1.Columns["id\_form"].Visible = false;

dataGridView1.Columns["id\_hospital"].Visible = false;

addComboBoxColumn("cmb\_firm", "Фирма", "Фирма производитель", "id\_firm", "название", 8);

addComboBoxColumn("cmb\_group", "Группа", "Фармокологическая группа", "id\_group", "группа", 2);

addComboBoxColumn("cmb\_form", "Форма", "Форма выпуска", "id\_form", "форма", 3);

addComboBoxColumn("cmb\_hospital", "Больница", "Заводы", "id\_hospital", "номер", 7);

setComboBoxColumn("cmb\_firm", "id\_firm");

setComboBoxColumn("cmb\_group", "id\_group");

setComboBoxColumn("cmb\_form", "id\_form");

setComboBoxColumn("cmb\_hospital", "id\_hospital");

foreach (System.Windows.Forms.DataGridViewRow row in dataGridView1.Rows)

{

if (row.Cells["дата"].Value != null && row.Cells["дата"].Value.ToString() != "")

row.Cells["дата"].Value = row.Cells["дата"].Value.ToString().Substring(0, 10);

}

fix\_cmb\_width();

}

private void btn\_firms\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_firms\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Фирма производитель";

fields = new string[]

{

"id\_firm", "название", "id\_type", "страна", "год"

};

fields\_types = new string[]

{

"int", "string", "int", "string", "int"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

dataGridView1.Columns["id\_type"].Visible = false;

addComboBoxColumn("cmb\_type", "Тип", "Тип собственности", "id\_type", "тип", 2);

setComboBoxColumn("cmb\_type", "id\_type");

fix\_cmb\_width();

}

private void btn\_departs\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_departs\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Отделения";

fields = new string[]

{

"id\_depart", "отделение"

};

fields\_types = new string[]

{

"int", "string"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

fix\_cmb\_width();

StretchLastColumn();

}

private void btn\_jiraf\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_jiraf\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Жираф";

fields = new string[]

{

"id\_jiraf", "id\_hospital", "id\_depart"

};

fields\_types = new string[]

{

"int", "int", "int"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

dataGridView1.Columns["id\_hospital"].Visible = false;

dataGridView1.Columns["id\_depart"].Visible = false;

addComboBoxColumn("cmb\_hospital", "Больница", "Заводы", "id\_hospital", "номер", 1);

addComboBoxColumn("cmb\_depart", "Отделение", "Отделения", "id\_depart", "отделение", 2);

setComboBoxColumn("cmb\_hospital", "id\_hospital");

setComboBoxColumn("cmb\_depart", "id\_depart");

fix\_cmb\_width();

StretchLastColumn();

}

private void btn\_type\_hosp\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_type\_hosp\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Типы заводы";

fields = new string[]

{

"id\_type", "тип"

};

fields\_types = new string[]

{

"int", "string"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

fix\_cmb\_width();

StretchLastColumn();

}

private void btn\_districts\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_districts\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Районы города";

fields = new string[]

{

"id\_district", "район"

};

fields\_types = new string[]

{

"int", "string"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

fix\_cmb\_width();

StretchLastColumn();

}

private void btn\_form\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_form\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Форма выпуска";

fields = new string[]

{

"id\_form", "форма"

};

fields\_types = new string[]

{

"int", "string"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

fix\_cmb\_width();

StretchLastColumn();

}

private void btn\_type\_sobs\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_type\_sobs\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Тип собственности";

fields = new string[]

{

"id\_type", "тип"

};

fields\_types = new string[]

{

"int", "string"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

fix\_cmb\_width();

StretchLastColumn();

}

private void btn\_farmo\_group\_Click(object sender, RoutedEventArgs e)

{

last\_click = btn\_type\_sobs\_Click;

last\_sender = sender;

move\_rect(sender);

table = "Фармокологическая группа";

fields = new string[]

{

"id\_group", "группа"

};

fields\_types = new string[]

{

"int", "string"

};

FillGridFromDataSet(FillDataSetFromDB(generate\_select(fields, table)));

fix\_cmb\_width();

StretchLastColumn();

}

private void btn\_back\_Click(object sender, RoutedEventArgs e)

{

move\_rect(sender);

QueryWindow window = new QueryWindow(connection\_string);

window.Show();

move\_rect(last\_sender);

}

bool readStrFromCell(System.Windows.Forms.DataGridViewCell cell, out string r)

{

bool result = true;

if (cell.Value != null)

{

r = cell.Value.ToString();

}

else

{

r = "";

result = false;

}

return result;

}

bool readIntFromCell(System.Windows.Forms.DataGridViewCell cell, out string i)

{

int temp = 0;

i = "null";

if (cell.Value != null)

{

if (int.TryParse(cell.Value.ToString(), out temp))

{

i = cell.Value.ToString();

}

else if (cell.Value.ToString() != "")

{

return false;

}

}

else

{

return false;

}

return true;

}

private void dataGridView1\_CellValueChanged(object sender, System.Windows.Forms.DataGridViewCellEventArgs e)

{

var row = dataGridView1.Rows[e.RowIndex];

string name = dataGridView1.Columns[e.ColumnIndex].Name;

if (name.Length > 3 && name.Substring(0, 3) == "cmb")

{

row.Cells["id" + name.Substring(3)].Value = row.Cells[name].Value;

return;

}

string sql = "";

if (row.Cells[fields[0]].Value == null)//insert

{

string values\_string = "";

for(int i=1;i<fields.Length; i++)

{

string temp= "";

if (fields\_types[i] == "int")

{

bool result = readIntFromCell(row.Cells[fields[i]], out temp);

if (result == false && !fields[i].StartsWith("id"))

{

MessageBox.Show("вы ввели не число!", "ОШИБКА", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

values\_string += $"{temp}";

}

else if (fields\_types[i] == "string")

{

readStrFromCell(row.Cells[fields[i]], out temp);

values\_string += $"'{temp}'";

}

if (temp.Equals("null") || temp.Trim(' ').Equals(""))

return;

if (i != fields.Length-1)

values\_string += ", ";

}

sql = $"insert into \"{table}\"(";

for (int i = 1; i< fields.Length; i++){

if (i != fields.Length - 1)

{

sql += $"\"{fields[i]}\", ";

}

else

{

sql += $"\"{fields[i]}\"";

sql += ") VALUES (";

}

}

sql += values\_string;

sql += ")";

execSql(sql);

last\_click(last\_sender, null);

}

else//update

{

sql = $" update \"{table}\" set ";

for (int i = 1; i < fields.Length; i++)

{

string temp = "";

if (fields\_types[i] == "int")

{

bool result = readIntFromCell(row.Cells[fields[i]], out temp);

if (result == false && !fields[i].StartsWith("id"))

{

MessageBox.Show("вы ввели не число!", "ОШИБКА", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

else if (fields\_types[i] == "string")

{

readStrFromCell(row.Cells[fields[i]], out temp);

temp = $"'{temp}'";

}

if (i != fields.Length - 1)

sql += $"\"{fields[i]}\" = {temp}, ";

else

sql += $"\"{fields[i]}\" = {temp} ";

}

sql += $" where \"{fields[0]}\" = {row.Cells[fields[0]].Value.ToString()}";

execSql(sql);

//last\_click(last\_sender, null);

}

}

System.Windows.Forms.DateTimePicker dtp = new System.Windows.Forms.DateTimePicker();

Rectangle rectangle;

private void dtp\_TextChange(object sender, EventArgs e)

{

dataGridView1.CurrentCell.Value = dtp.Text.ToString();

dtp.Visible = false;

}

private void dataGridView1\_ColumnWidthChanged(object sender, System.Windows.Forms.DataGridViewColumnEventArgs e)

{

dtp.Visible = false;

}

private void dataGridView1\_CellClick(object sender, System.Windows.Forms.DataGridViewCellEventArgs e)

{

if (e.RowIndex != -1 && e.ColumnIndex != -1)

{

if (table == "Заказы")

{

if (dataGridView1.Columns[e.ColumnIndex].Name == "дата")

{

rectangle = dataGridView1.GetCellDisplayRectangle(e.ColumnIndex, e.RowIndex, true);

dtp.Size = new System.Drawing.Size(rectangle.Width, rectangle.Height);

dtp.Location = new System.Drawing.Point(rectangle.X, rectangle.Y);

dtp.Visible = true;

}

}

}

}

private void dataGridView1\_UserDeletingRow(object sender, System.Windows.Forms.DataGridViewRowCancelEventArgs e)

{

string sql = $"delete from {table} where \"{fields[0]}\" = {e.Row.Cells[fields[0]].Value.ToString()}";

if (MessageBox.Show("вы уверены что хотите удалить запись?", "Внимание!", MessageBoxButton.YesNo) == MessageBoxResult.No)

{

e.Cancel = true;

return;

}

execSql(sql);

}

private int getCountRowsTable(string table, string where="1=1")

{

NpgsqlConnection connection = new NpgsqlConnection(connection\_string);

connection.Open();

var dataAdapter = new NpgsqlDataAdapter(

string.Format("SELECT COUNT(\*) FROM \"{0}\" WHERE {1}", table, where),

connection);

var dataTable = new DataTable();

dataAdapter.Fill(dataTable);

connection.Close();

return int.Parse(dataTable.Rows[0][0].ToString());

}

string search = "";

int page = 0;

private void tb\_search\_KeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Enter)

{

search = tb\_search.Text;

last\_click(last\_sender, null);

}

}

private void btn\_left\_Click(object sender, RoutedEventArgs e)

{

page = Math.Max(0, --page);

last\_click(last\_sender, null);

}

private void btn\_right\_Click(object sender, RoutedEventArgs e)

{

page++;

last\_click(last\_sender, null);

}

}

}

# Приложение Д РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

При запуске программы запускается окно авторизации. Если ввести правильные данные авторизации, программа пропустит пользователя в окно с таблицами.

Для того, чтобы удалить запись в таблице, необходимо выбрать её(нажать, чтобы она стала выделенной) и нажать на клавиатуре кнопку «Delete», если же вы хотите удалить группу записей необходимо зажать «Ctrl» и нажать на записи( чтобы они стали выделенными), а после нажать на клавиатуре кнопку «Delete». Если нажатие на кнопку «Delete» было ошибочно, то сперва у Вас спросят: “Вы действительно хотите удалить запись?”, если нажмёте «Да», то записи удалятся, «Нет» - удаление не произойдёт.

Для того, чтобы добавить запись в таблицу, необходимо заполнить последнюю строчку в таблице. Если формат данных будет соблюден, запись добавится в базу данных.

Для того, чтобы редактировать запись в таблице, необходимо выбрать ее в таблице, нажать на ячейку в нужной колонке два раза и ввести данные.

Для поиска, на основной форме необходимо ввести в поле текст и поиск произведётся автоматически.

Для отображения графика, нужно нажать на меню «Графики», которое находится вверху главной формы. В открывшейся форме нажать на кнопку «Кнопочка», возле каждого графика.

Для того, чтобы экспортировать результат запроса в Excel, необходимо зайти на форму с запросами и нажать «Экспортировать в Excel» и выбрать путь сохранения файла.

Для просмотра графиков в окне с запросами необходимо в самом низу нажать на кнопку с графиками.

# ПРИЛОЖЕНИЕ Ж РУКОВОДСТВО АДМИНИСТРАТОРА

При запуске программы перед пользователем появляется меню авторизации. Корректно заполним логин и пароль в форме авторизации и нажать войти, то перед вами откроется меню с таблицами.

Для того, чтобы удалить одну запись в таблице, необходимо выбрать её (нажать, чтобы она стала выделенной) и нажать на клавиатуре кнопку «Delete», если же вы хотите удалить группу записей необходимо зажать «Ctrl» и нажать на записи (чтобы они стали выделенными), а после нажать на клавиатуре кнопку «Delete». Если нажатие на кнопку «Delete» было ошибочно, то сперва у Вас спросят: “Вы действительно хотите удалить запись?”, если нажмёте «Да», то записи удалятся, «Нет» - удаление не произойдёт.

Для того, чтобы добавить запись в таблицу, необходимо заполнить последнюю строчку в таблице. Если формат данных будет соблюден, запись добавится в базу данных.

Для того, чтобы редактировать запись в таблице, необходимо выбрать ее в таблице, нажать на ячейку в нужной колонке два раза и ввести данные.

Для поиска, на основной форме необходимо ввести в поле текст и поиск произведётся автоматически.

Для отображения графика, нужно нажать на меню «Графики», которое находится внизу формы запросов.

Для того, чтобы экспортировать результат запроса в Excel, необходимо зайти на форму с запросами и нажать «Экспортировать в Excel» и выбрать путь сохранения файла, а так же задать имя файлу.

Для просмотра графиков в окне с запросами необходимо в самом низу нажать на кнопку с графиками.