РЕФЕРАТ

Пояснительная записка содержит 29 с., 21 рис., 2 табл., 3 источника, 3 приложения.

Объектом исследования данной работы является работа с поступлениями книжных товаров в магазины.

Цель работы – разработать информационную систему для учета поступления литературы в книжные магазины, получить знания и практической опыт работы с системами управления базами данных, методами проектирования моделей баз данных, составления запросов на языке структурированных запросов SQL.

Результатом работы является реляционная база данных, предназначенная для хранения информации, необходимой для работы с поставками товаров в магазины, разработанная в соответствии требованиям к курсовой работе. Разработанная база данных написана на языке программирования C# и использует синтаксис SQL для доступа и изменения данных.

Для разработки системы использована СУБД PostgreSQL.

ПОСТАВКИ, КНИГА, МАГАЗИНЫБ БАЗЫ ДАННЫХ, МОДЕЛЬ ДАННЫХ, VISUAL STUDIO, C#, ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ, ОТЧЕТЫ, ЗАПРОСЫ, SQL, POSTGRESQL

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc72972584)

[1 ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ 4](#_Toc72972585)

[1.1 Проблемы автоматизации 4](#_Toc72972586)

[1.1.1 Разработка информационной системы для учета поступления литературы в книжные магазины. 5](#_Toc72972587)

[1.1.2 Анализ предметной области 5](#_Toc72972588)

[1.1.3 Описание входных документов 5](#_Toc72972589)

[1.1.4 Описание выходных документов 6](#_Toc72972590)

[1.2 Назначение и функции системы 7](#_Toc72972591)

[2 ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 8](#_Toc72972592)

[2.1 Проектирование базы данных 8](#_Toc72972593)

[2.2 Физическое проектирование 9](#_Toc72972594)

[3 РАЗРАБОТКА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ 11](#_Toc72972595)

[3.1 Создание таблиц 11](#_Toc72972596)

[3.2 Схема базы данных (поддержка целостности данных) 11](#_Toc72972597)

[3.3 Разработка внешних представлений данных – меню, форм, запросов, отчетов 12](#_Toc72972598)

[Выводы 25](#_Toc72972599)

[Список источников 26](#_Toc72972600)

[Приложение А Техническое задание 27](#_Toc72972601)

[Приложение Б Руководство пользователя 28](#_Toc72972602)

[Приложение В Текст программы 29](#_Toc72972603)

# ВВЕДЕНИЕ

Целью данной курсовой работы является проектирование и создание реляционной базы данных Visual Studio для оптимизации работы с поставками книжных товаров в магазины и закрепление практических навыков разработки реляционных баз данных, проектирования моделей баз данных.

Оптимальным способом взаимодействия с базой данных является графический пользовательский интерфейс. Среда Microsoft Visual Studio предоставляет возможности для создания графических интерфейсов пользователя для операционной системы Microsoft Windows, а также других видов операционных систем.

Необходимо спроектировать модель взаимодействия пользователя с системой, разработать экранные формы в среде Visual Studio на основе Visual Studio Forms, запросы в базу данных на языке запросов SQL, спроектировать отчеты и диаграммы по статистическим показателям, разработать программу генерации тестовых данных, протестировать разработанную информационную системы на тестовых данных: частных и общих случаях, разработать эксплуатационную документацию для базы данных и информационной системы.

# ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

## Проблемы автоматизации

В процессе своей работы, сотрудники магазинов взаимодействуют с большими объемами данных. Одним из вариантов осуществления этой задачи является использование бумажных носителей данных. Существенным недостатком этого подхода является сложность поиска и изменения информации, невозможность эффективного удаленного доступа к данным.

Часть проблем «бумажного» делопроизводства решается использованием информационных систем. У данного подхода также имеется ряд недостатков: большая сложность поиска информации в хранилище, сложность обеспечения одинаковой структуры данных, сложность реализации многопользовательского доступа к хранилищу данных.

Для решения этих проблем разрабатываются автоматизированные информационные системы: программные системы, основанные на использовании структурированных хранилищ информации – баз данных. Преимуществами использования систем управления базами данных и прикладных программ, разработанных на их основе, является быстрый оптимизированный поиск по записям, одновременный доступ к данным несколькими удаленными пользователями, возможность составления сложных запросов к данным без необходимости реализации дополнительных программных средств для поиска информации.

Все ранее изложенное дает нам сделать вывод о том, что использование автоматизированной информационной системы является наилучшим подходом к реализации программной информационной системы, упрощающей работу агентства по трудоустройству.

### Разработка информационной системы для учета поступления литературы в книжные магазины.

### Анализ предметной области

В работе в сфере купли-продажи взаимодействует огромное количество людей. Для выгодного сотрудничества поставщиков и магазинов была создана данная реляционная база данных.

Сотрудники магазинов должны видеть статистику поставок, а так же на её основе повышать актуальность своего товара, заказывая у поставщиков более выгодный товар.

Поставщики, смотря какой товар более пользуется спросом, будут закупать издание большим тиражом для получения большего количества прибыли. Так же, статистические данные, показанные в базе данных помогут определить более «выгодные» даты (праздники и др.) , накануне которых можно продать издания большим тиражом.

Таким образом, сотрудникам магазинам и поставщикам необходима полная информация о их сотрудничестве, для повышения качества торговли.

### Описание входных документов

Основными видами документов в разрабатываемой базе данных являются документы, описывающие информацию об поставках, книгах и магазинах.

Информация о поставке включает в себя: дата поставки, название магазина, название книги, тираж, язык перевода, цена для магазина и для поставщика, объем заказа, статус предзаказа. На основе этой информации можно вывести статистику актуальности каждой книги, самых прибыльных дат, а так же степень сотрудничества с каждым магазином.

Информация о книге содержит в себе такие данные: название, краткое описание, обложка, язык оригинала, список авторов, издательство, жанр, год издания, а также год выпуска авторами(ом).

Информация о магазине содержит такие параметры: название, год создания, район, адрес и тип собственности.

### Описание выходных документов

Выходными документами являются диаграммы, используемые для представления построенных по данным статистических показателей. Такими статистическими показателями являются проценты типов собственности всех магазинов. Другой важным показателем является количество издательств в каждом городе. От этого зависит, в каком городе будет осуществляться большее количество поставок. И третий показатель, сумма закупок магазинов и их поставщиков, а также разница в этой цене, для сохранения усредненной цены.

Кроме диаграмм, выходными документами информационной системы служат различные отчеты. Отчеты позволяют просматривать и печатать на бумажный носитель информацию в удобном для чтения человеком виде. В одном из таких отчетов необходимо показать количество магазинов, заданного типа собственности. Кроме этого, важным отчетом является список магазинов, открытых в заданном пользователем году, а так же все районы, в которых есть заданный магазин.

## Назначение и функции системы

Пользователем разрабатываемой информационной системы является сотрудник магазина, издательства или поставщика. Назначением программы является упрощение сотрудничества данных организаций за счет ускорения получения обработки и редактирования данных, а так же статистики.

Функциями разрабатываемой системы является просмотр и редактирование списка издательств и магазинов. Все данные связанные ними должны быть редактируемыми пользователями программного продукта. Пользователь разрабатываемой информационной системы может использовать разработанную систему для добавления магазинов, поставок, книг и издательств.

Информационная система должна генерировать отчеты и диаграммы, полезные для сотрудников данных предприятий.

Информационная система должна позволить пользователю выполнять запросы в базу данных для поиска информации, что сильно упростит пользователю использование и анализ данных.

# ТЕХНИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

## Проектирование базы данных

В процессе проектирования была получена реляционная модель данных. На этом этапе были выделены таблицы, определены индексы, отношения между таблицами. В процессе проектирования модель данных приводится к третьей нормальной форме.

Основными таблицами в модели данных являются таблицы «Издательство», «Поставки», «Магазин» и «Книга».

Так же в процессе проектирования были выделены такие таблицы справочники: «Район», «Тип собственности», «Язык», «Переплет», «Жанр», «Книги-авторы» и «Город». Для отображения всех связей можно воспользоваться схемой базы данных (рис. 2.1).

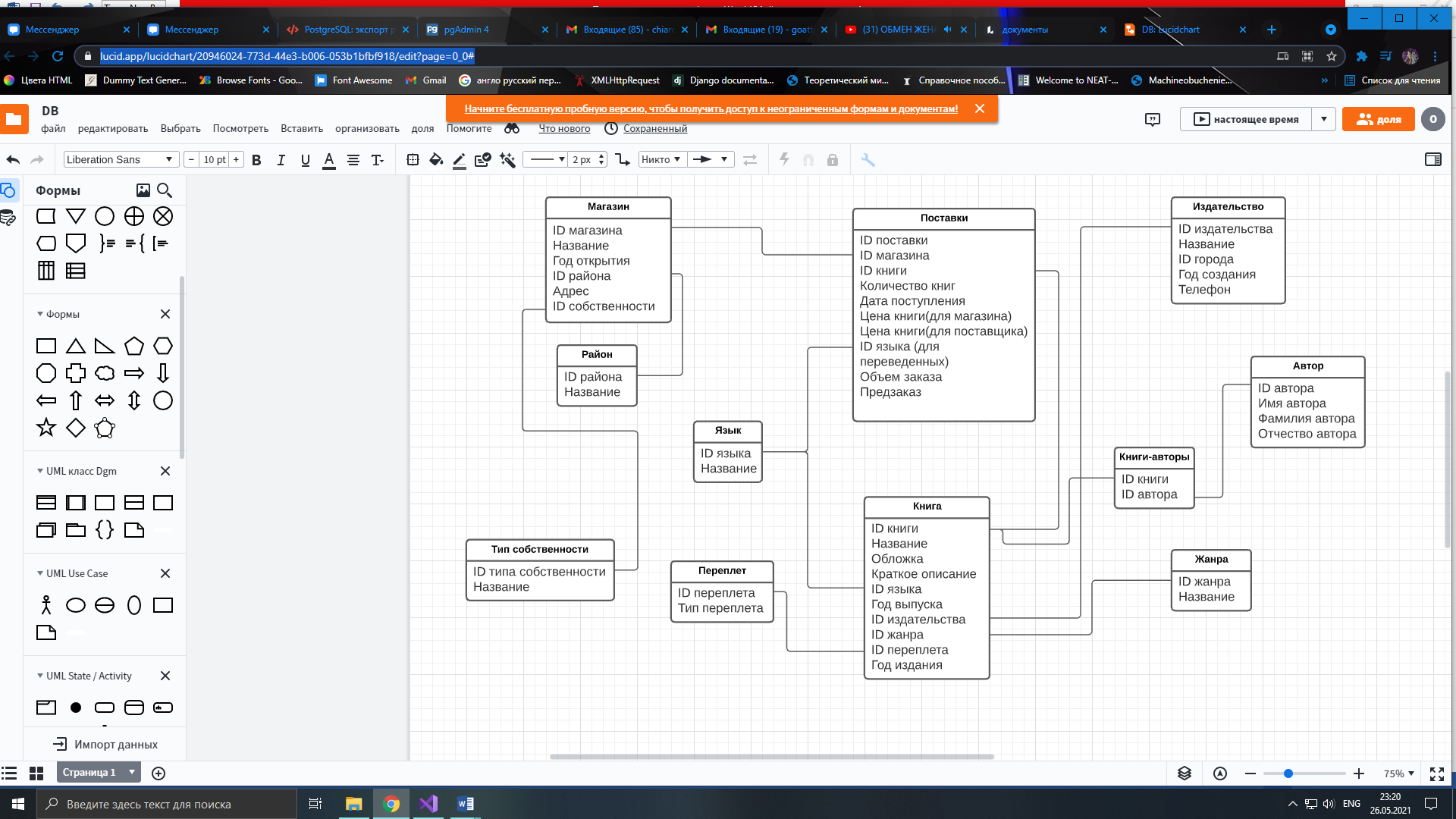


Рисунок 2.1 – Реляционная модель данных

Данная модель сформирована не зависимо от используемой системы управления реляционными базами данных. Следующим этапом проектирования является формирования физической модели данных с учетом особенностей конкурентной СУБД. В рамках данной курсовой работы – PostgreSQL.

## Физическое проектирование

На стадии физического проектирования базы данных необходимо определить структуру базы данных с учетом возможностей, особенностей и ограничений системы управления базы данных PostgreSQL.

При физическом проектировании баз данных принято названия таблиц и их полей записывать буквами латинского алфавита. Кроме этого, для удобства программирования информационной системы, связанной с базой данной, рекомендуется давать полям таблиц одинаковые имена. Рассмотрим структуру всех таблиц базы данных (таблица 2.1).

Таблица 2.1 – Поля таблиц базы данных

| Таблица | Поле | Назначение поля |
| --- | --- | --- |
| Магазин | id | Код магазина |
| name | Название магазина |
| date\_open | Дата открытия |
| address | Адрес |
| Id\_area | Код района |
| Id\_own | Код типа собственности |
| Поставки | id | Код поставки |
| Id\_shop | Код магазина |
| Id\_book | Код книги |
| Book\_count | Количество книг |
| Date\_come | Дата поступления |
| cost | Цена для магазина |
| Def\_cost | Цена для поставщика |
| Id\_lang | Код языка |
| Size | Объем заказа |
| pre\_order | Предзаказ |
| Книга | id | Код работодателя |
| name | Название |
| Photo | Фото |
| Description | Краткое описание |
| Id\_lang | Код языка оригинала |
| Date\_public | Дата издания |
| Date\_create | Дата создания |
| Id\_style | Код жанра |
| Id\_publisher | Код издательства |
| Id\_binding | Код переплета |
| Издательство | Id | Код издательства |
| name | Название |
| Id\_city | Код города |
| Date\_create | Дата создания |
| phone | Телефон |
| Тип собственности | id | Код типа собственности |
| name | Тип собственности |
| Район | id | Код района |
| name | Название |
| Язык | id | Код языка |
| name | Название |
| Переплет | id | Код переплета |
| name | Название |
| Жанр | id | Код жанра |
| name | Название |
| Автор | id | Код автора |
| name | Имя |
| Second\_name | Фамилия |
| Last\_name | Отчество |
| Город | id | Код города |
| name | Город |
| Книги-авторы | Id\_book | Код книги |
| Id\_author | Код автора |

# РАЗРАБОТКА ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

## Создание таблиц

При создании таблиц базы данных необходимо создать индексы для оптимизации поиска данных в таблице и установления связи между таблицами. Поле однозначно идентифицирующее запись должно быть помечено как первичный ключ. Если первичный ключ – числовое поле (тип Integer). Тип этого поля может быть изменен на автоматически инкрементируемое числовое поле. При этом для каждой записи будет автоматически создаваться уникальный первичный ключ – число на 1 больше чем предыдущее сгенерированного число. Шаг изменение автоматически инкрементируемого поля может быть указан при редактировании структуры таблицы.

Для проверки правильности задаваемых данных можно задать правила для проверки значений полей. Например, в данной курсовой работе, поля, содержащие цену книги должны быть неотрицательны.

## Схема базы данных (поддержка целостности данных)

При разработке реляционной базы данных возникает проблема наличия записей в таблицах, которые содержат ключи, указывающие на белее не существующую запись в другой таблице. Это может произойти при удалении записи родительской таблицы, изменении индекса в родительской таблице или при создании записи в дочерней таблице и указании неправильного значения в поле при создании записи в дочерней таблице.

Система управления базами данных PostgreSQL содержит инструменты для поддержки целостности данных. С помощью специального интерфейса можно установить действие, выполняемое СУБД при изменении и удалении записи в родительской таблице и при создании записи в дочерней таблице.

Поскольку в разработанной модели данных все ключевые поля являются неизменяемыми (тип Integer (AutoIncrement)), необходимо задать правильное действие, выполняемое при создании и удалении полей таблицы.

Все выбранные действия, выполняемые при изменении данных показаны в таблице 3.1.

Таблица 3.1 – поддержка целостности данных

|  |  |  |
| --- | --- | --- |
| Родительская таблица | Дочерняя таблица | Удаление |
| Район | Магазин | Cascade |
| Города | Издательство | Cascade |
| Язык | Поставки | Cascade |
| Язык | Книга | Cascade |
| Жанр | Книга | Cascade |
| Переплет | Тип переплета | Cascade |
| Автор | Книги-авторы | Cascade |
| Типы собственности | Магазин | Cascade |
| Книга | Книги-авторы | Cascade |

В таблице 3.1 значение Cascade означает, что при удалении записи в родительской таблице будут удалены все зависимые записи дочерней таблицы.

## Разработка внешних представлений данных – меню, форм, запросов, отчетов

Для разработки эффективного человеко-машинного интерфейса для информационной системы необходимо определить основные функции системы и действия, производимые пользователем над данными в процессе работы с информационной системой. Существует множество методик для выявления основных требований, предъявляемых к интерфейсу системы. Одной из них является составление диаграммы прецедентов (рисунок 3.1). Диаграмма прецедентов показывает все возможные действия, совершаемые всеми пользователями информационной системы. Разрабатываемая информационная система имеет единственную категорию пользователей – сотрудников агентства по трудоустройству.

На основании построенной диаграммы можно выделить необходимые формы информационной системы: список поставок, магазинов, книг, издательств, авторизация, выполнение запросов.

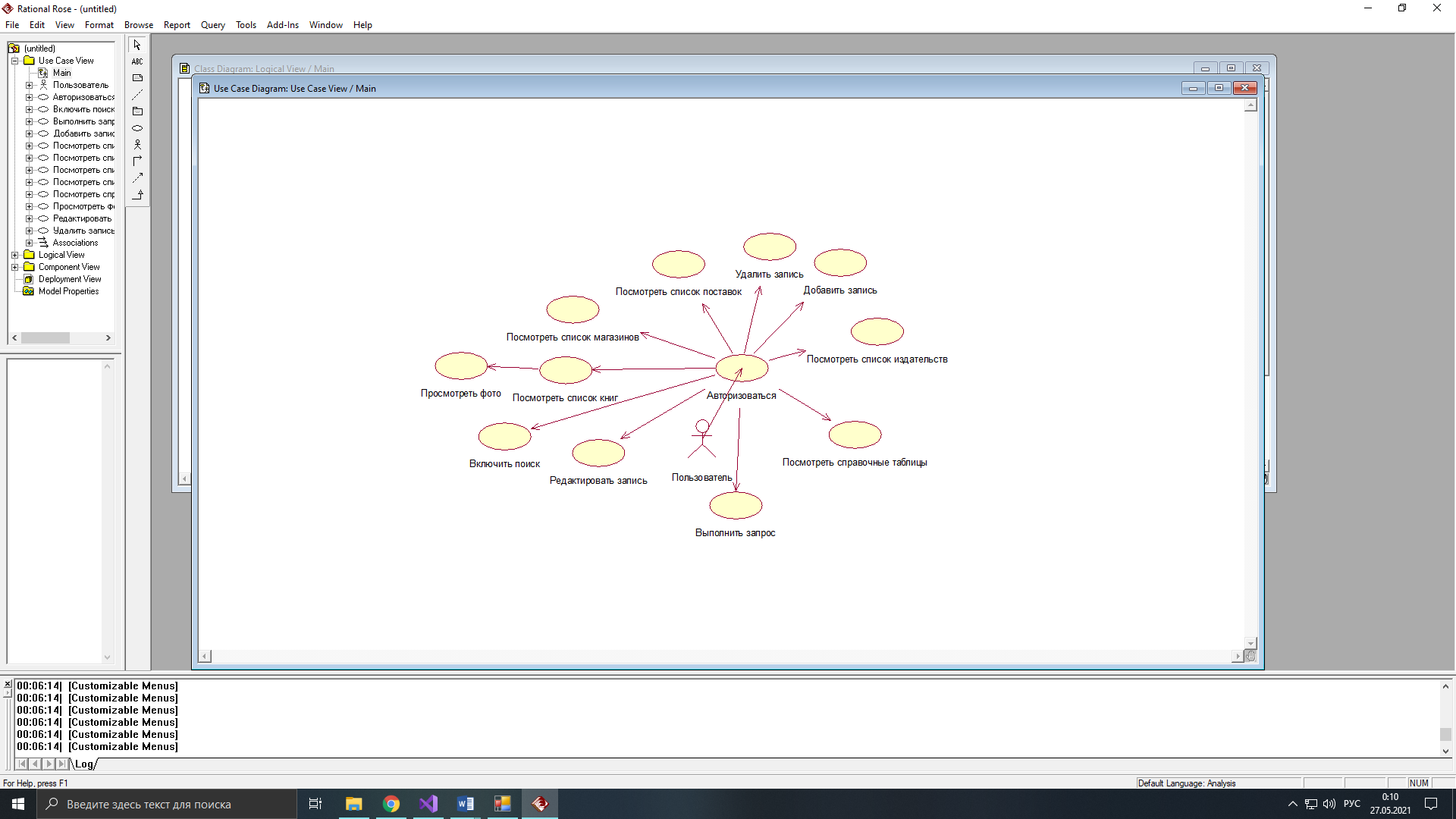


Рисунок 3.1 – Диаграмма прецедентов.

Навигация между формами может осуществляться с помощью главного меню. Пунктами главного меню наивысшего уровня являются: «Основные таблицы» (открытие основных таблиц для изменения, просмотра и удаления записей), «Справочники» (редактирование записей в справочниках базы данных), «Запросы» (выполнение запросов в базу данных в соответствии с требованиями курсовой работы), «Отчеты» (просмотр графиков и построение отчетов).

В соответствии требованиям к курсовой работе, первая форма, которую видит пользователь при запуске программной системы является форма авторизации (рисунок 3.2). На этой форме пользователь системы может указать пользователя и ввести пароль.

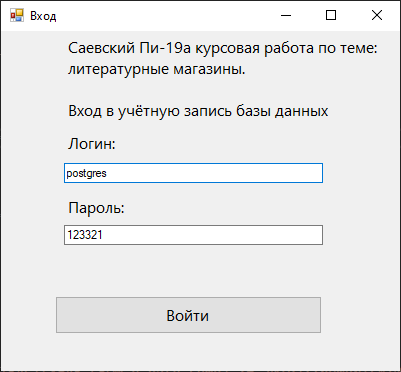


Рисунок 3.2 – Форма авторизации

После авторизации пользователь получает доступ ко всем функциям информационной системы. Устанавливается главное меню приложения (рисунок 3.3). Из данной формы можно просмотреть все основные таблицы (рисунок 3.4), а также редактировать и добавлять записи (рисунок 3.5 – 3.6).

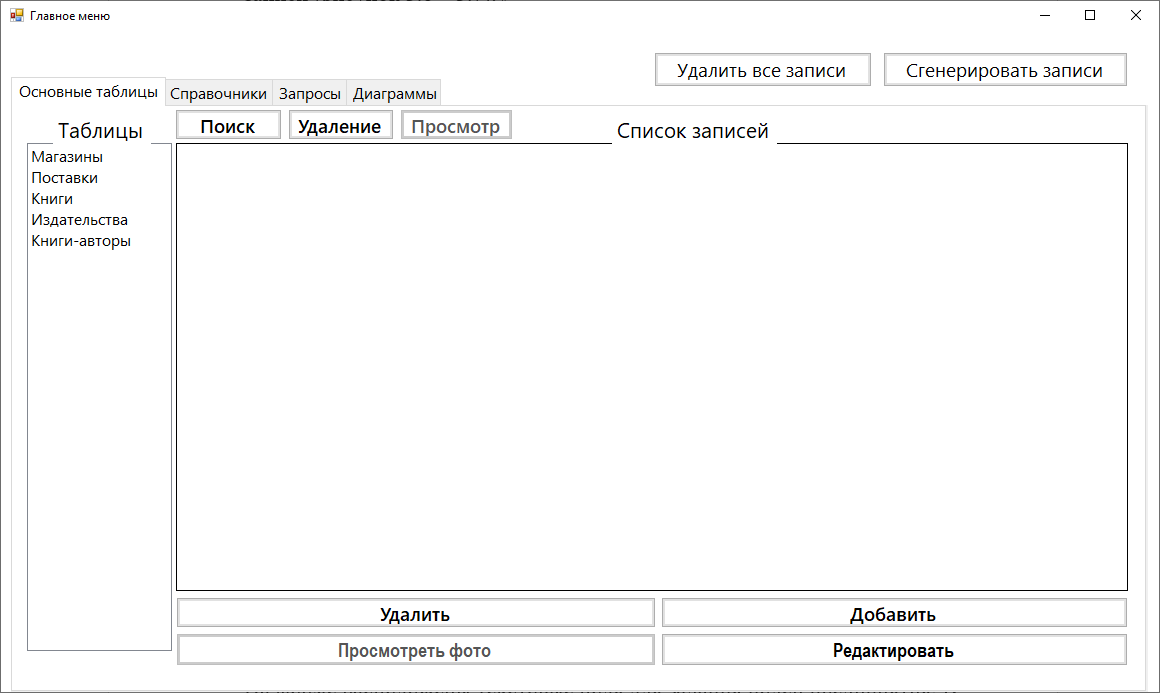


Рисунок 3.3 – Главное меню

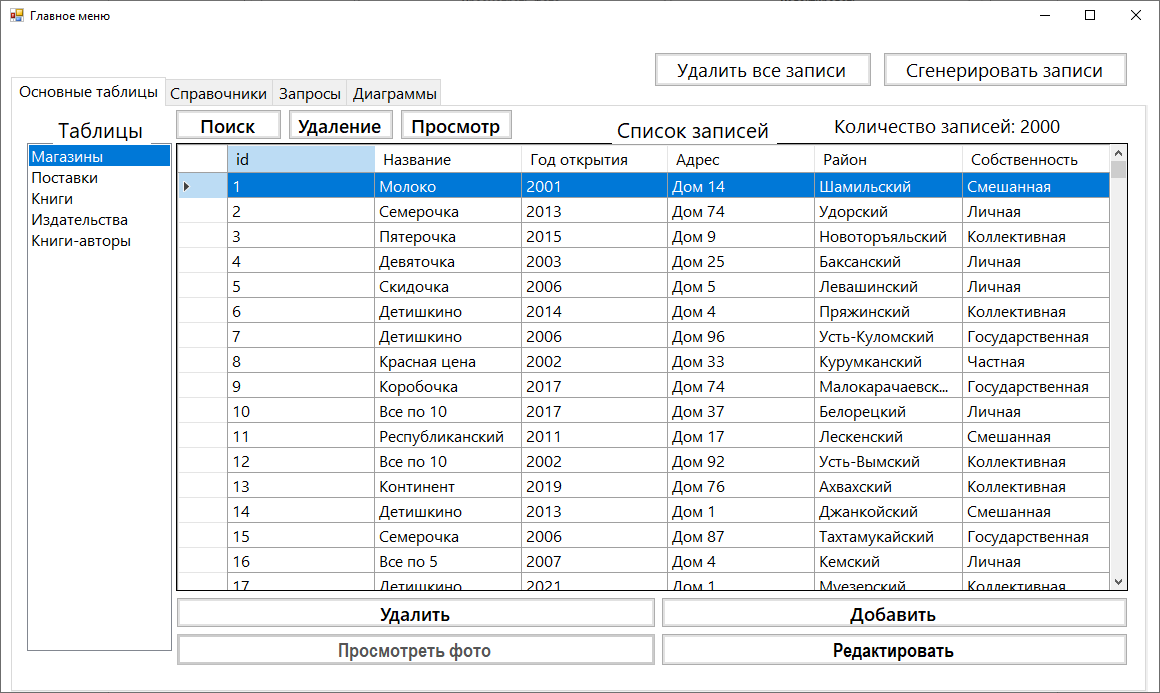


Рисунок 3.4 – Форма «Магазины»

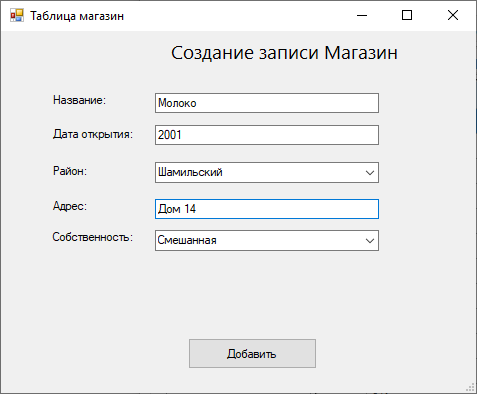


Рисунок 3.5 – Форма редактирования магазина

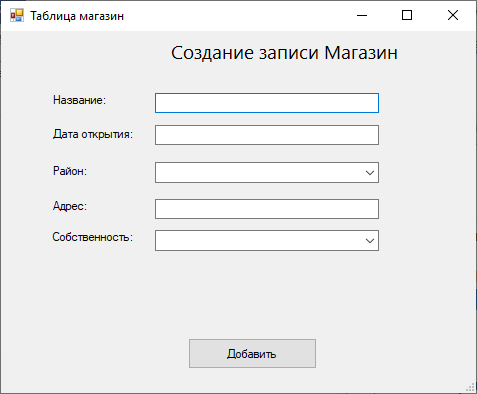


Рисунок 3.6 – Форма добавления магазина

Для поиска информации в базе данных с помощью сложных запросов, используются SQL-запросы. Пользователь информационной системы может выполнить запросы, в соответствии с требованиями к курсовой работе с помощью формы «Запросы» (рисунок 3.7).

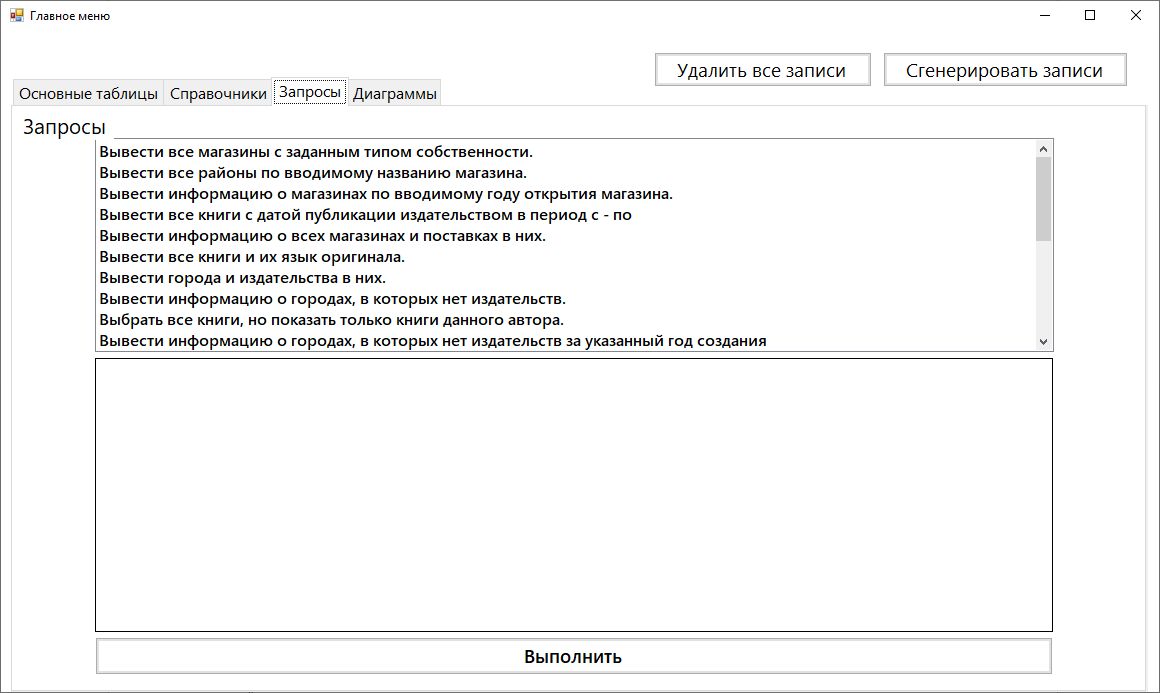


Рисунок 3.7 – Форма «Запросы»

Информационная система позволяет пользователю выполнять 20 различных запросов. При выборе интересующего пользователя запроса в нижней части формы будет показан результат запросы после нажатия на кнопку «Выполнить».

На рисунке 3.8 можно увидеть пример запроса в базу данных, а на рисунке 3.9 – результат запроса.

SELECT b.id id, b.name Название, b.photo Фото, p.name Издательство, l.name Язык

FROM book b

JOIN publisher p ON b.id\_publisher = p.id

JOIN lang l ON b.id\_lang = l.id

ORDER BY b.name, l.name

Рисунок 3.8 – Пример запроса в базу данных   
(Вывести все книги и их язык оригинала.)

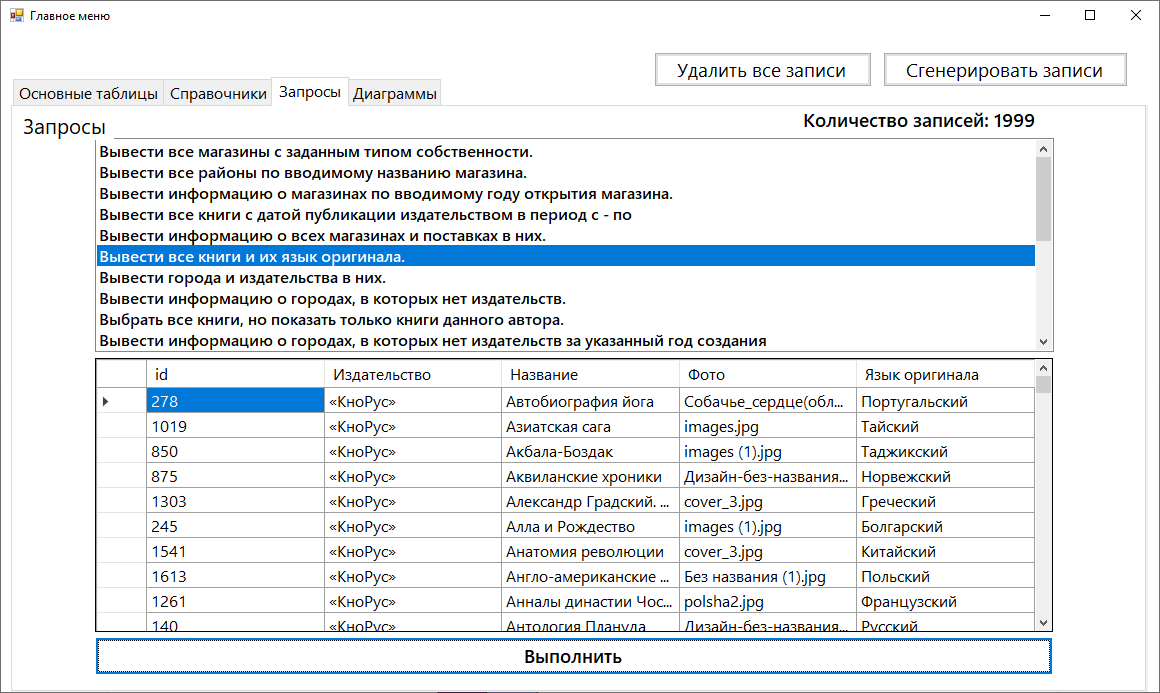


Рисунок 3.9 – Результат запроса

Важными выходными документами разрабатываемой системы являются диаграммы, построенные по статистическим показателям. В данной курсовой работе было построено три диаграммы.

По одномерному статистическому показателю была построена круговая диаграмма (рисунок 3.10). На диаграмме показан процент количества магазинов каждого типа собственности.

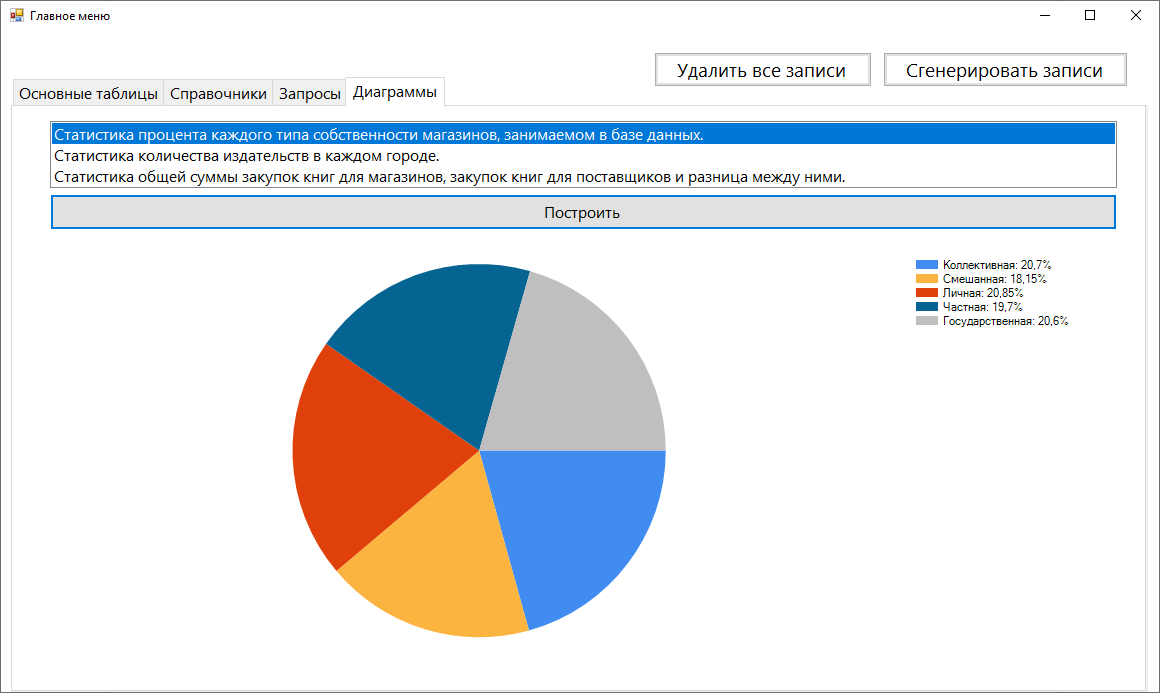


Рисунок 3.10 – Круговая диаграмма

По одномерному статистическому показателю была построена гистограмма (рисунок 3.11). На диаграмме показано количество издательств в каждом городе. Для показания количества издательств в конкретном городе нужно выбрать его из списка.

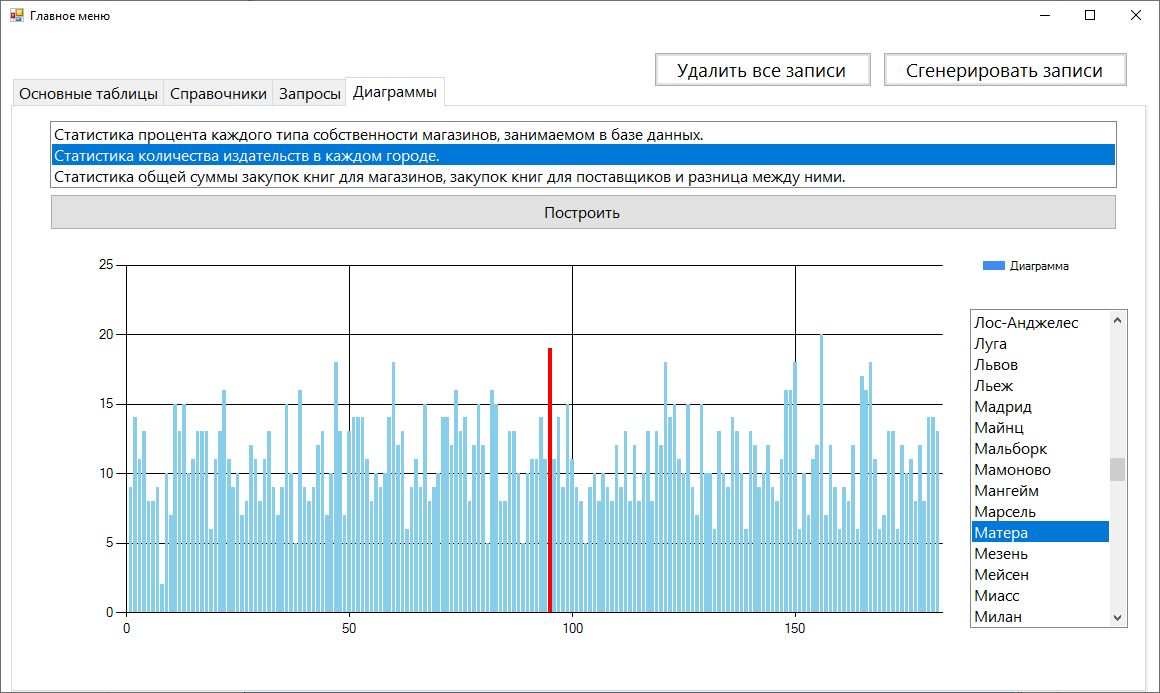


Рисунок 3.11 – Гистограмма.

Третьей диаграммой была построена гистограмма с тремя параметрами. Она показывала стоимость закупок магазинов, поставщиков и разницу в цене между ними за каждый год поставок (рисунок 3.12).

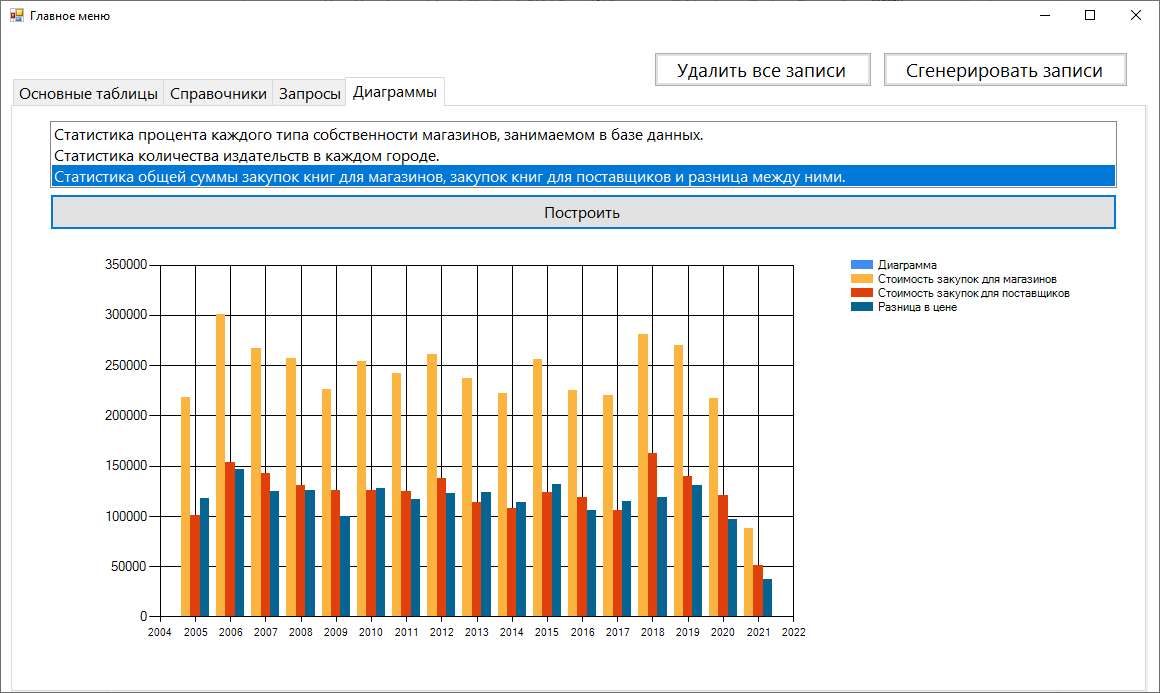


Рисунок 3.12 – Зависимость стоимости закупок магазинов и поставщиков от года

В работе важным инструментом являются отчеты, построенные по данным. В данной курсовой работе было создано три отчета. Отчеты в C# позволяют просматривать содержимое таблиц базы данных используя более расширенные возможности форматирования и группировки данных, чем таблицы, которые могут быть расположены на форме.

Примеры отчетов в html формате показаны на рисунках 3.13 – 3.15, а в xlsx формате – на рисунках 3.16-3.18.

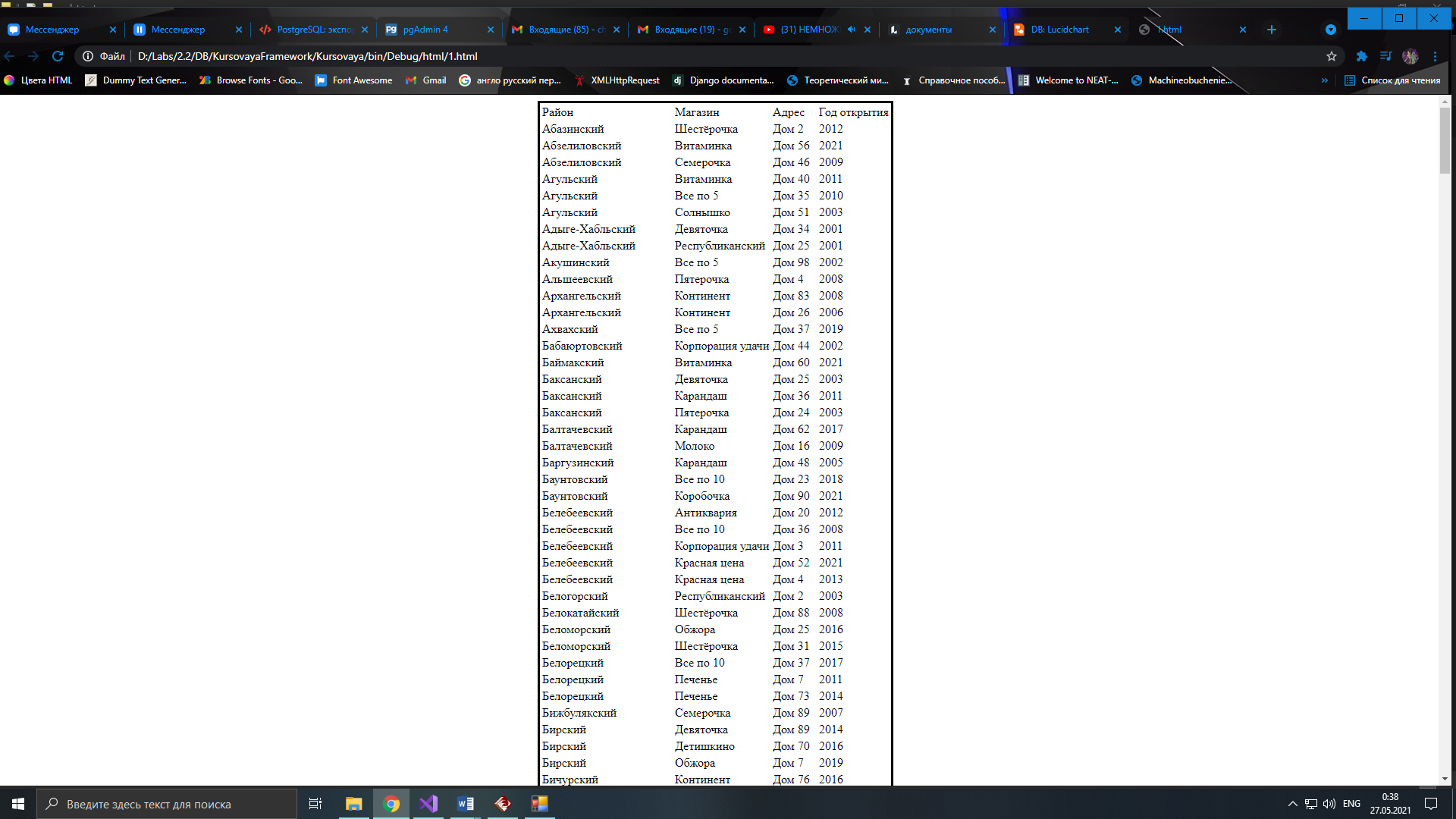


Рисунок 3.13 – Фрагмент отчета «Магазины с заданным типом собственности»

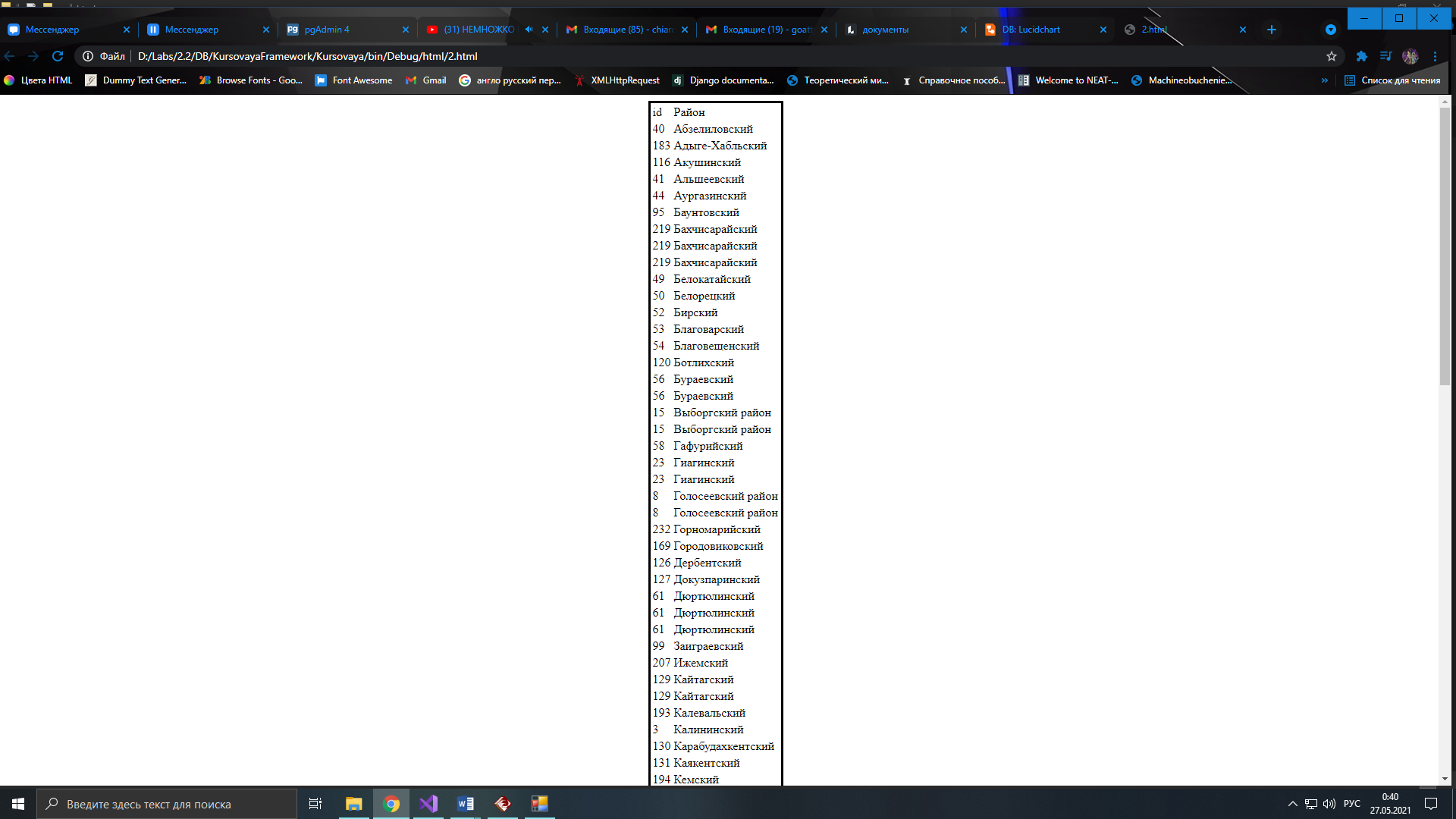


Рисунок 3.14 – Фрагмент отчета «Районы по заданному названию магазина»

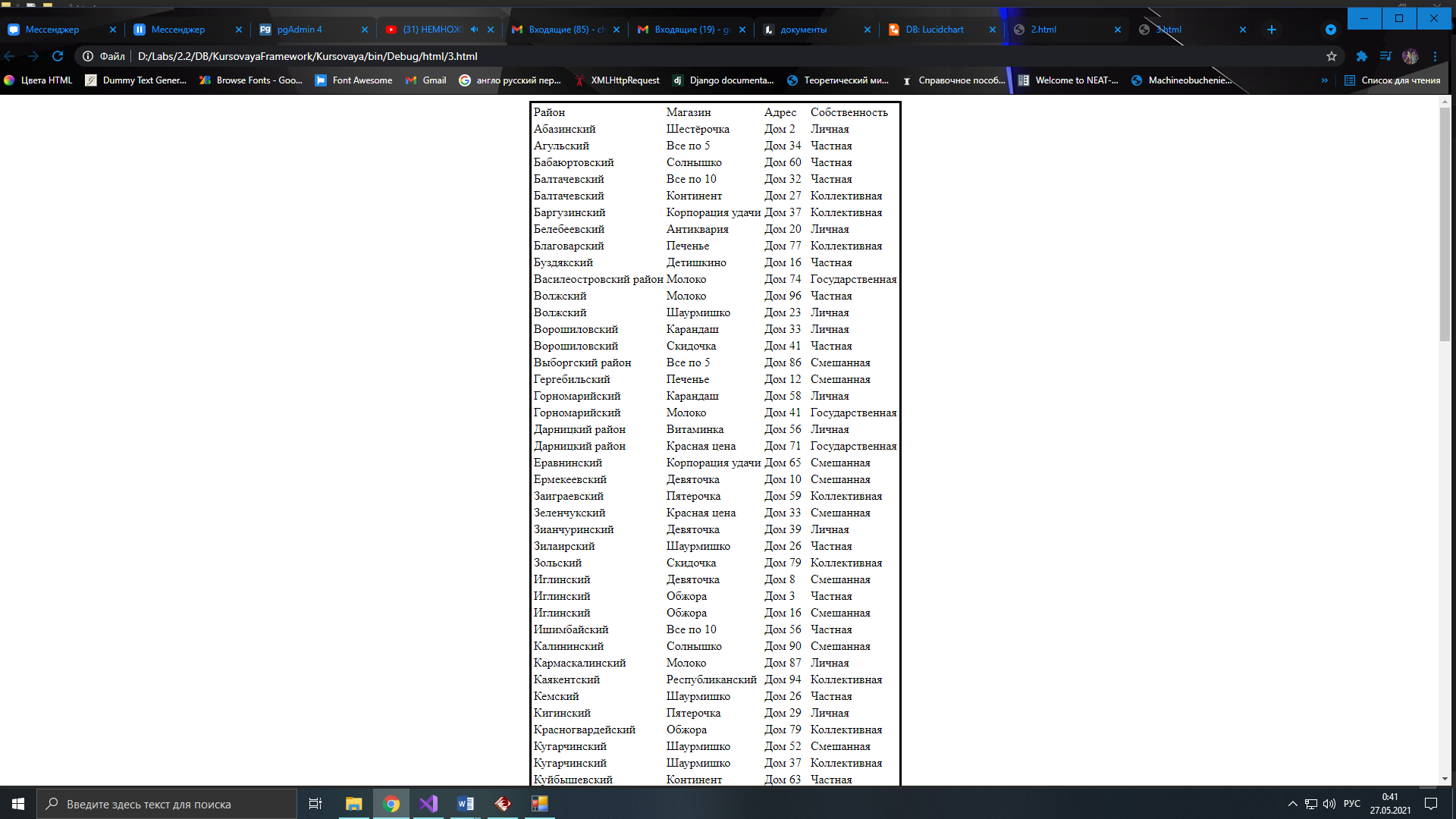


Рисунок 3.15 – Фрагмент отчета «Магазины по заданному году открытия»

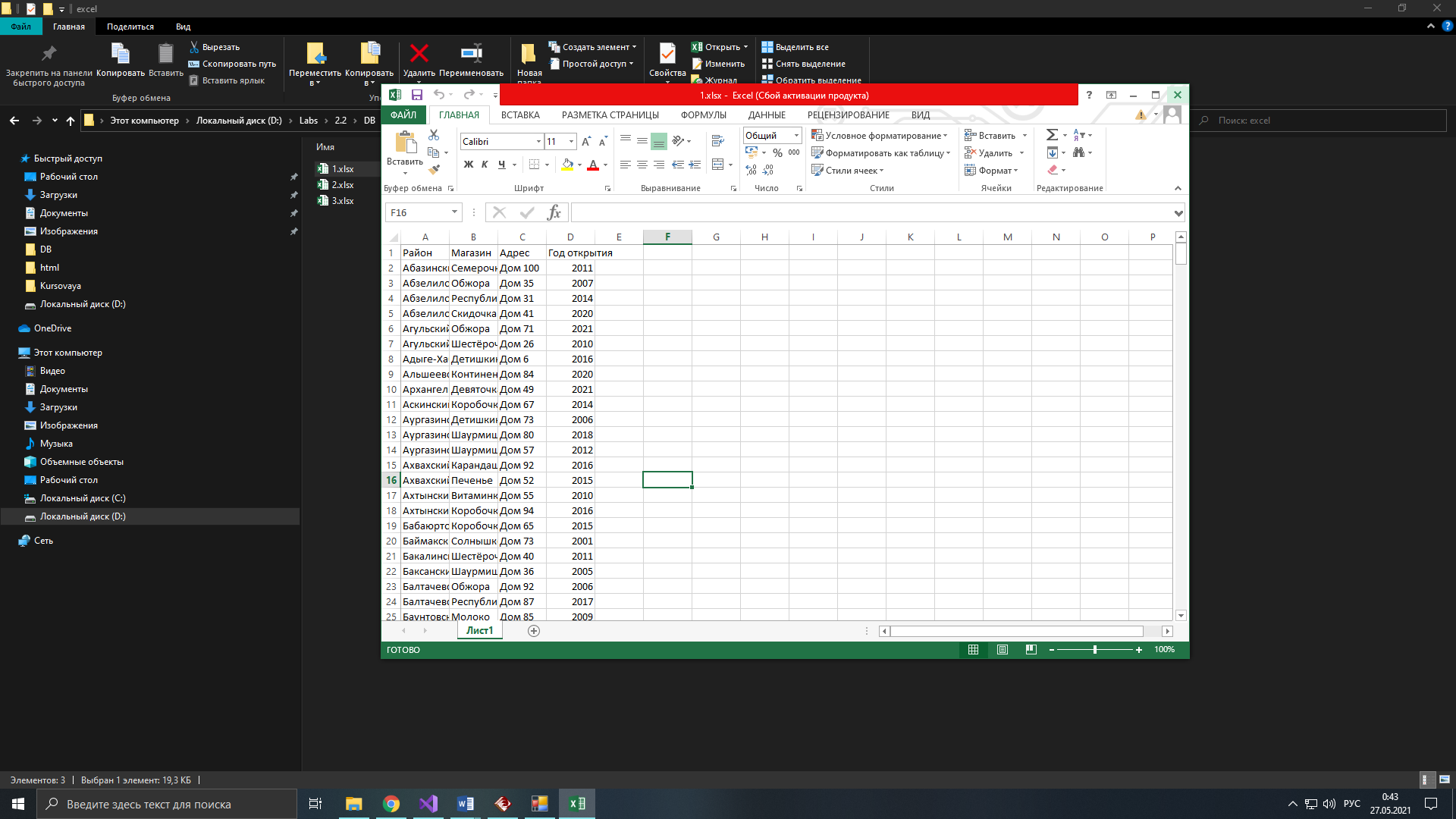


Рисунок 3.16 – Фрагмент отчета «Магазины с заданным типом собственности»

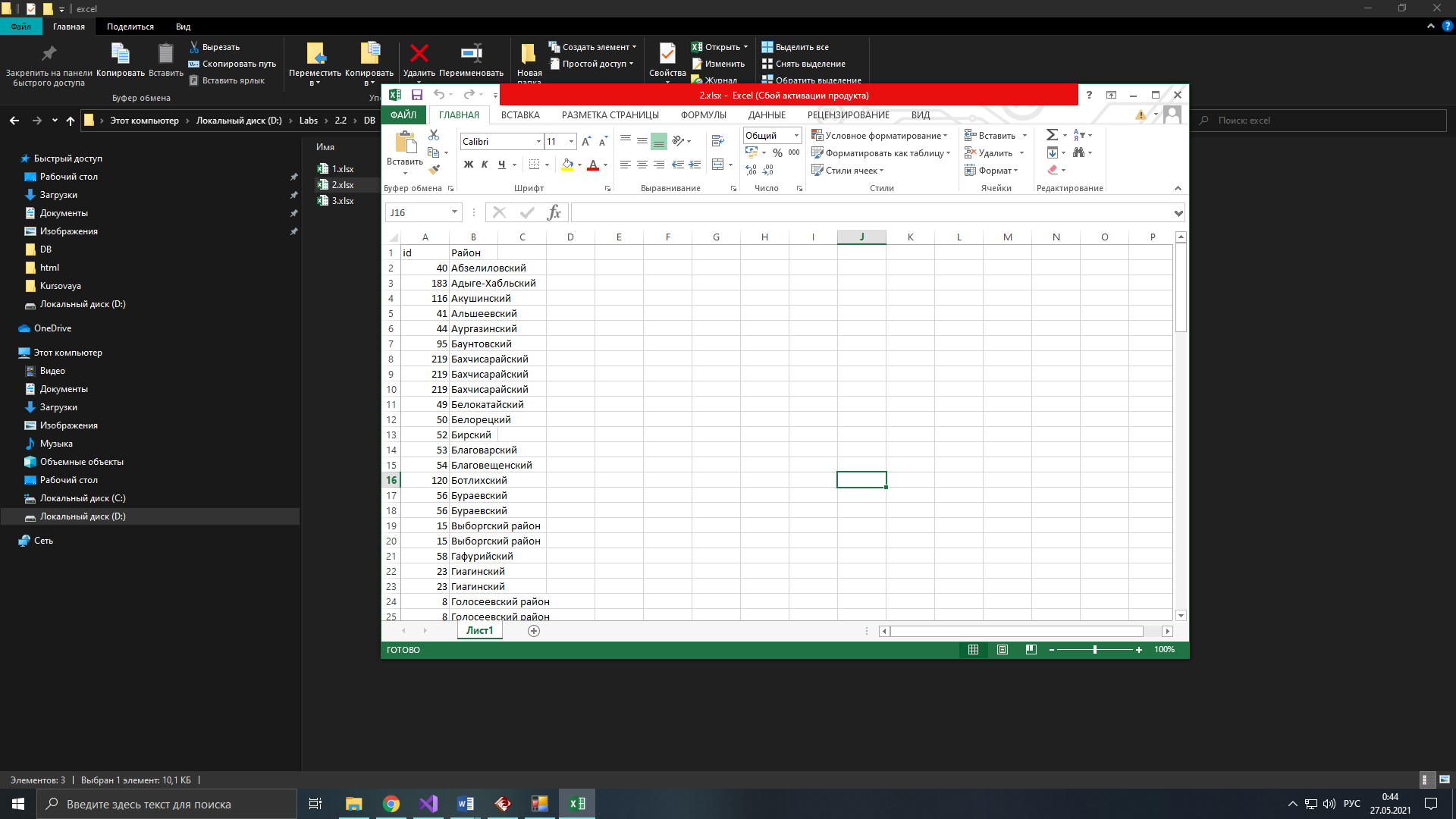


Рисунок 3.17 – Фрагмент отчета «Районы по заданному названию магазина»

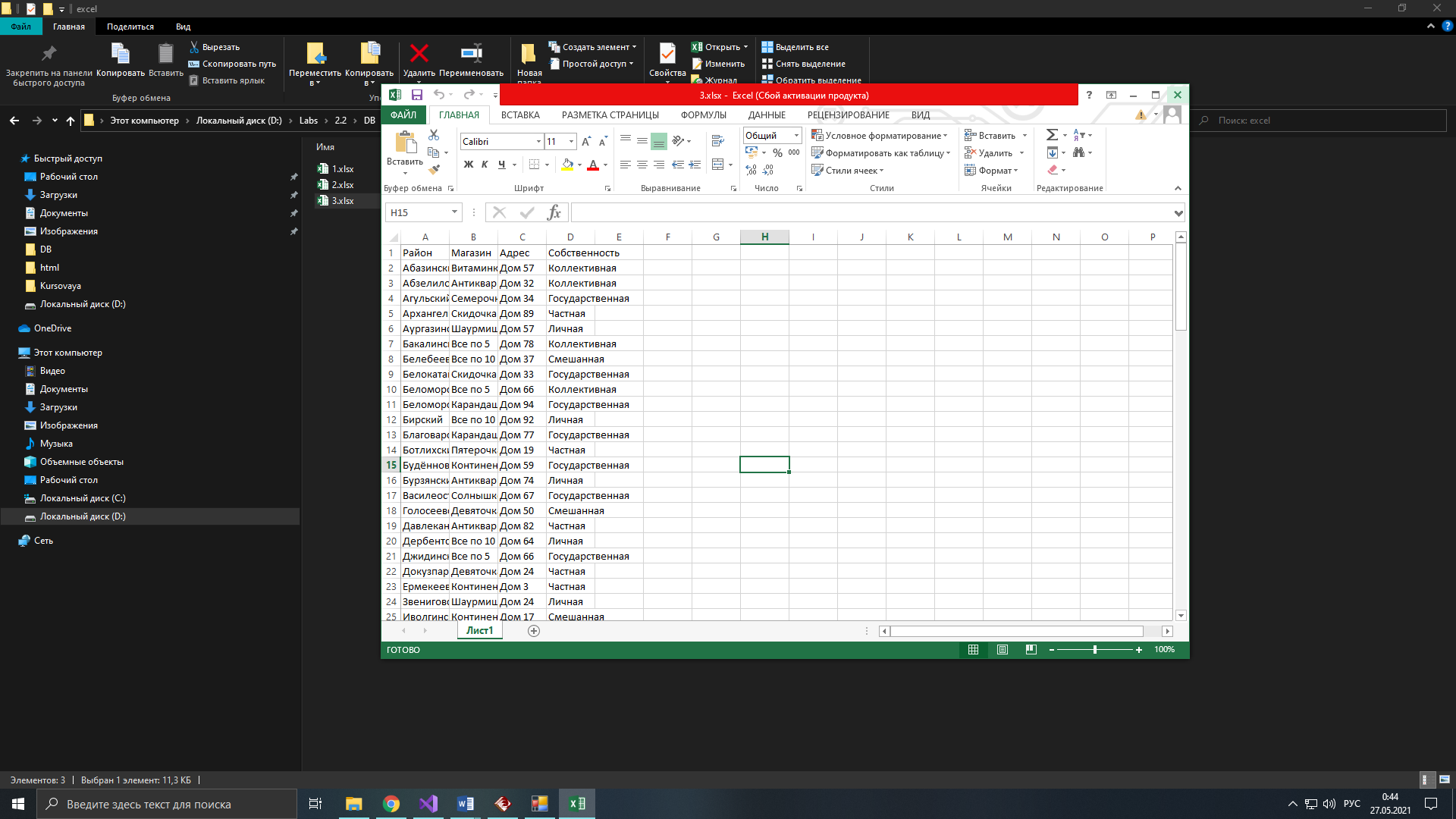


Рисунок 3.18 – Фрагмент отчета «Магазины по заданному году открытия»

Выводы

В ходе выполнения курсовой работы были закреплены практические навыки работы с системами управления базами данных, программирования информационных систем, методов автоматизации работы с базами данных.

В результате выполнения курсовой работы было проведено даталогическое, физическое проектирование базы данных, создана проект базы данных, созданы таблицы базы данных, сгенерированы случайные данные для заполнения таблиц. Был спроектирован, разработан и протестирован графический пользовательский интерфейс, были составлены запросы в базу данных в соответствии с требованиями к курсовой работе, созданы запросы и диаграммы.

В качестве руководства для программиста может быть использован данный отчет по курсовой работе. Было составлено руководство для оператора информационной системы.

Список источников

1. Базы данных: Учебник для высших учебных заведений / Под ред. проф. А.Д. Хомоненко. - СПб.: КОРОНА принт, 2000.
2. Гарсиа–Молиа Г., Ульман Дж. Д., Уидом Дж. Системы баз данных. Полный курс.– М: Изд.дом “Вильямс”, 2003.
3. Эрик Редмонд, Джим. Р. Уилсон Семь баз данных за семь недель. Введение в современные базы данных. Под редакцией Жаклин Картер / Пер. с англ. Слинкин А. А. – М.: ДМК Пресс, 2013. – 384с.

Приложение А  
Техническое задание

Приложение Б  
Руководство пользователя

Для работы необходимо устройство под управлением операционной системы Microsoft Windows XP или новее.

Для открытия программы необходимо запустить исполняемый файл Kursovaya.exe. При открытии программы у пользователя информационной системы будет запрошено имя пользователя, которое необходимо указать и ввести пароль пользователя. Если введенный пароль не совпадает с паролем, который хранится в базе данных пользователю информационной системы будет выведено сообщение об ошибке «Пароль введен не верно». В обратном случае, окно будет закрыто, на экране будет отображено главное меню и открыто окно с альтернативной навигацией.

В главном меню доступны главные пункты «Основные таблицы», «Справочники», «Запросы», «Диаграммы». Во вкладке «Основные таблицы» можно добавлять, удалять, редактировать, просматривать записи. Во вкладке «Справочники» можно добавлять и удалять записи. При удалении записи из вкладки «Справочники» будут удалены записи из «Основной таблицы», которые содержали данную справочную запись. Во вкладке «Запросы» можно выбрать из списка необходимый запрос и ввести запрашиваемые данные при необходимости. Диаграммы позволяют визуализировать некоторые запросы.

## ПРИЛОЖЕНИЕ В ТЕКСТ ПРОГРАММЫ

FORM1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

namespace Kursovaya

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

textBox1.Text = "postgres";

textBox2.Text = "123321";

}

SQL sql;

private void button1\_Click(object sender, EventArgs e)

{

sql = new SQL(textBox1.Text, textBox2.Text);

using (NpgsqlConnection connect = SQL.GetConnection())

{

try

{

connect.Open();

if (connect.State != ConnectionState.Open)

{

Message.AutorizationError();

Application.Exit();

}

connect.Close();

Main main = new Main();

main.Show();

this.Hide();

}

catch(Exception ex)

{

MessageBox.Show("Неверный логин или пароль.", "Ошибка 000", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

private void Form1\_Load(object sender, EventArgs e)

{

}

}

}

Main.cs

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Threading;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using Npgsql;

namespace Kursovaya

{

public partial class Main : Form

{

public Main()

{

InitializeComponent();

tabPage1.Text = "Основные таблицы";

tabPage2.Text = "Справочники";

tabPage3.Text = "Запросы";

tabPage4.Text = "Диаграммы";

SetTagPage2();

SetTagPage3();

AddTables();

dataGridView1.ReadOnly = true;

dataGridView2.ReadOnly = true;

button8.Enabled = false;

button12.Enabled = false;

chart1.Series.Clear();

chart1.Series.Add("Диаграммы");

}

private void SetTagPage2()

{

listBox2.Size = listBox1.Size;

listBox2.Location = listBox1.Location;

dataGridView2.Size = dataGridView1.Size;

dataGridView2.Location = dataGridView1.Location;

button3.Text = button1.Text;

button3.Size = button1.Size;

button3.Location = button1.Location;

button3.Font = button1.Font;

button4.Text = button2.Text;

button4.Size = button2.Size;

button4.Location = button2.Location;

button4.Font = button2.Font;

label3.Text = label2.Text;

label3.Location = label2.Location;

label3.Size = label2.Size;

label3.Font = label2.Font;

label7.Text = label6.Text;

label7.Location = label6.Location;

label7.Size = label6.Size;

label7.Font = label6.Font;

label9.Location = label8.Location;

label9.Text = " ";

}

private void SetTagPage3()

{

label5.Size = label2.Size;

label5.Font = label2.Font;

button5.Font = button1.Font;

}

private void AddTables()

{

listBox1.Items.Add("Магазины");

listBox1.Items.Add("Поставки");

listBox1.Items.Add("Книги");

listBox1.Items.Add("Издательства");

listBox1.Items.Add("Книги-авторы");

listBox2.Items.Add("Район");

listBox2.Items.Add("Город");

listBox2.Items.Add("Язык");

listBox2.Items.Add("Тип собственности");

listBox2.Items.Add("Жанр");

listBox2.Items.Add("Тип переплета");

listBox2.Items.Add("Автор");

listBox3.Items.Add("Вывести все магазины с заданным типом собственности.");

listBox3.Items.Add("Вывести все районы по вводимому названию магазина.");

listBox3.Items.Add("Вывести информацию о магазинах по вводимому году открытия магазина.");

listBox3.Items.Add("Вывести все книги с датой публикации издательством в период с - по");

listBox3.Items.Add("Вывести информацию о всех магазинах и поставках в них.");

listBox3.Items.Add("Вывести все книги и их язык оригинала.");

listBox3.Items.Add("Вывести города и издательства в них.");

listBox3.Items.Add("Вывести информацию о городах, в которых нет издательств.");

listBox3.Items.Add("Выбрать все книги, но показать только книги данного автора.");

listBox3.Items.Add("Вывести информацию о городах, в которых нет издательств за указанный год создания");

listBox3.Items.Add("Выдать количество издательств в каждом городе");

listBox3.Items.Add("Вывести название и тираж книг с датой публикации издательством в период с-по.");

listBox3.Items.Add("Выбрать названия магазинов с общей стоимостью закупок, превышающей заданное значение и определить количество поставленных партий.");

listBox3.Items.Add("Выбрать названия магазинов с общей стоимостью закупок, превышающей заданное значение и определить количество поставленных партий в период поставок с (дата) – по (дата)");

listBox3.Items.Add("Определить количество магазинов каждой магазинной сети и количество поставок для каждой сети.");

listBox3.Items.Add("Выдать информацию об издательствах у которых год создания больше среднего года создания всех издательств");

listBox3.Items.Add("Определить процент изданий поступивших в магазины после 2019 года по каждому магазину района");

listBox3.Items.Add("Определить процент изданий поступивших в магазины после 2019 года по каждому району в целом");

listBox3.Items.Add("Определить среднее количество изданий каждого издательства");

listBox3.Items.Add("Определить среднее количество изданий по всем издательствам в целом");

listBox3.Items.Add("Определить стоимость и количество книг каждого магазина за указанный период (по месяцам)");

dataGridView2.ColumnHeadersHeight = 30;

label1.Location = label9.Location;

label1.Font = label9.Font;

label9.Size = label9.Size;

}

private void tagPage1\_Selecting(object sender, TabControlCancelEventArgs e)

{

if (e.TabPage.Text == "Основные таблицы")

{

UpdateDatagrid(dataGridView1, listBox1, label1);

}

else if (e.TabPage.Text == "Справочники")

{

UpdateDatagrid(dataGridView2, listBox2, label9);

}

else if (e.TabPage.Text == "Запросы")

{

//Открыть форму Request

}

}

private void tagPage1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void button2\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedItem == null)

return;

if (listBox1.SelectedItem.ToString() == "Магазины")

new EditShop().ShowDialog(this);

else if (listBox1.SelectedItem.ToString() == "Поставки")

new EditDeliveries().ShowDialog(this);

else if (listBox1.SelectedItem.ToString() == "Книги")

new EditBook().ShowDialog(this);

else if (listBox1.SelectedItem.ToString() == "Издательства")

new EditPublisher().ShowDialog(this);

UpdateDatagrid(dataGridView1, listBox1, label1);

}

private void button4\_Click(object sender, EventArgs e)

{

if (listBox2.SelectedItem != null)

{

if (listBox2.SelectedItem.ToString() == "Автор")

{

EditAuthor AddAuthor = new EditAuthor();

AddAuthor.ShowDialog(this);

}

else

{

EditSup AddSup = new EditSup(listBox2.SelectedItem.ToString());

AddSup.ShowDialog(this);

}

UpdateDatagrid(dataGridView2, listBox2, label9);

}

}

private void button5\_Click(object sender, EventArgs e)

{

if (listBox3.SelectedItem == null)

return;

string Query = null;

InputText it = new InputText();

dataGridView3.DataSource = new DataTable();

if (listBox3.SelectedIndex == 0)

{

it.ShowDialog(this);

if (it.GetResult() == null)

return;

Query = $"SELECT a.name Район, s.name Магазин, s.address Адрес, s.date\_open \"Год открытия\" FROM shop s " +

$"JOIN own o ON s.id\_own = o.id " +

$"JOIN area a ON s.id\_area = a.id " +

$"WHERE o.name = '{it.GetResult()}' " +

$"ORDER BY a.name, s.name";

label10.Text = "Собственность: " + it.GetResult();

new Thread(() =>

{

new HtmlReport("1.html", new string[] { "Район", "Магазин", "Адрес", "Год открытия" }).Parse(Query);

new ExcelReport("1", new string[] { "Район", "Магазин", "Адрес", "Год открытия" }).Parse(Query);

}).Start();

}

else if (listBox3.SelectedIndex == 1)

{

it.ShowDialog(this);

if (it.GetResult() == null)

return;

Query = $"SELECT a.id id, a.name Район FROM area a " +

$"JOIN shop s ON a.id = s.id\_area WHERE s.name = '{it.GetResult()}' " +

$"ORDER BY a.name";

label10.Text = "Магазин: " + it.GetResult();

new Thread(() =>

{

new HtmlReport("2.html", new string[] { "id", "Район" }).Parse(Query);

new ExcelReport("2", new string[] { "id", "Район" }).Parse(Query);

}).Start();

}

else if (listBox3.SelectedIndex == 2)

{

it.ShowDialog(this);

int Year;

if (it.GetResult() == null)

return;

try

{

Year = Convert.ToInt32(it.GetResult());

}

catch { Message.ErrorShow("Значение указано неверно."); return; }

Query = "SELECT a.name Район, s.name Магазин, s.address Адрес, o.name Собственность " +

"FROM shop s " +

"JOIN area a ON s.id\_area = a.id " +

"JOIN own o ON s.id\_own = o.id " +

$"WHERE s.date\_open = {Year} " +

"ORDER BY a.name, s.name";

label10.Text = "Год открытия: " + it.GetResult();

new Thread(() =>

{

new HtmlReport("3.html", new string[] { "Район", "Магазин", "Адрес", "Собственность" }).Parse(Query);

new ExcelReport("3", new string[] { "Район", "Магазин", "Адрес", "Собственность" }).Parse(Query);

}).Start();

}

else if (listBox3.SelectedIndex == 3)

{

InputPeriod ip = new InputPeriod();

ip.ShowDialog(this);

if (ip.GetResult() == null)

return;

int FirstYear, SecondYear;

try

{

FirstYear = Convert.ToInt32(ip.GetResult()[0]);

SecondYear = Convert.ToInt32(ip.GetResult()[1]);

if (FirstYear > SecondYear)

throw new Exception();

}

catch { Message.ErrorShow("Значение указано неверно."); return; }

Query = "SELECT b.date\_public \"Дата публикации\" , p.name Издательство ,b.name Книга, b.description Описание, l.name Язык, bind.name Переплет, b.date\_create \"Дата создания\"," +

" b.photo Фото FROM book b " +

"JOIN lang l ON b.id\_lang = l.id " +

"JOIN publisher p ON p.id = b.id\_publisher " +

"JOIN binding bind ON bind.id = b.id\_binding " +

$"WHERE b.date\_public >= {FirstYear} AND b.date\_public <= {SecondYear} " +

$"ORDER BY b.date\_public, p.name, b.name";

label10.Text = "Период: " + FirstYear.ToString() + "-" + SecondYear.ToString();

//new HtmlReport("4.html", new string[] { "Дата публикации", "Издательство", "Книга", "Описание", "Язык", "Переплет", "Дата создания", "Фото" }).Parse(Query);

//new ExcelReport("4", new string[] { "Дата публикации", "Издательство", "Книга", "Описание", "Язык", "Переплет", "Дата создания", "Фото" }).Parse(Query);

}

else if (listBox3.SelectedIndex == 4)

{

Query = "SELECT d.id \"id Поставки\", a.name Район, s.name Магазин, s.address Адрес, o.name Собственность, b.name Книга, d.count\_book Количество, d.date\_come \"Дата поступления\"," +

" d.cost \"Цена для магазина\", d.def\_cost \"Цена для поставщика\", l.name Язык, d.size Объем, d.pre\_order Предзаказ " +

"FROM deliveries d " +

"JOIN book b ON d.id\_book = b.id " +

"JOIN shop s ON d.id\_shop = s.id " +

"JOIN area a ON s.id\_area = a.id " +

"JOIN lang l ON l.id = b.id\_lang " +

"JOIN own o ON s.id\_own = o.id " +

"ORDER BY a.name, s.name, d.date\_come";

label10.Text = null;

//new HtmlReport("5.html", new string[] { "id", "Магазин", "Район", "Адрес", "Собственность", "Книга", "Тираж", "Дата поступления", "Цена для магазина", "Цена для поставщика", "Язык", "Объем", "Предзаказ" }).Parse(Query);

}

else if (listBox3.SelectedIndex == 5)

{

label10.Text = null;

Query = "SELECT b.id id, p.name Издательство, b.name Название, b.photo Фото, l.name \"Язык оригинала\" FROM book b JOIN publisher p ON b.id\_publisher = p.id JOIN lang l ON b.id\_lang = l.id ORDER BY p.name, b.name, l.name";

//new HtmlReport("6.html", new string[] { "id", "Издательство", "Название", "Фото", "Язык оригинала" }).Parse(Query);

}

else if (listBox3.SelectedIndex == 6)

{

label10.Text = null;

Query = "SELECT c.name Город, p.name Издательство FROM publisher p JOIN city c ON c.id = p.id\_city ORDER BY c.name, p.name";

//new HtmlReport("7.html", new string[] { "Город", "Издательство" }).Parse(Query);

}

else if (listBox3.SelectedIndex == 7)

{

label10.Text = null;

Query = "SELECT c.id id, c.name Город FROM city c LEFT JOIN publisher p ON p.id\_city = c.id ORDER BY c.name";

//new HtmlReport("8.html", new string[] { "id", "Город" }).Parse(Query);

}

else if (listBox3.SelectedIndex == 8)

{

InputAuthor ia = new InputAuthor();

ia.ShowDialog(this);

if (ia.GetResult() == null)

return;

string[] Author = ia.GetResult().Split(' ');

Query = "SELECT b.name Книга, b.photo Фото "+

"FROM book\_author ba "+

"RIGHT JOIN book b ON b.id = ba.id\_book "+

"JOIN author a ON a.id = ba.id\_author " +

$"WHERE a.name = '{Author[1]}' AND a.second\_name = '{Author[0]}' AND a.last\_name = '{Author[2]}'ORDER BY b.name";

label10.Text = "Автор: " + Author[0] + " " + Author[1] + " " + Author[2];

//new HtmlReport("9.html", new string[] { "Книга", "Фото" }).Parse(Query);

}

else if (listBox3.SelectedIndex == 9)

{

InputPeriod ip = new InputPeriod();

ip.ShowDialog(this);

if (ip.GetResult() == null)

return;

int FirstYear, SecondYear;

try

{

FirstYear = Convert.ToInt32(ip.GetResult()[0]);

SecondYear = Convert.ToInt32(ip.GetResult()[1]);

if (FirstYear > SecondYear)

throw new Exception();

}

catch { Message.ErrorShow("Значение указано неверно."); return; }

Query = $"SELECT DISTINCT c.id id, c.name Город FROM city c " +

$"LEFT JOIN publisher p ON p.id\_city = c.id " +

$"WHERE c.id NOT IN (SELECT c.id FROM publisher p " +

$"JOIN city c ON p.id\_city = c.id WHERE p.date\_create >= {FirstYear} AND p.date\_create <= {SecondYear}) ORDER BY c.name; ";

}

else if (listBox3.SelectedIndex == 10)

{

Query = $"SELECT c.name Город, COUNT(p.id) \"Количество издательств\" FROM city c JOIN publisher p ON p.id\_city = c.id GROUP BY (c.name) ORDER BY c.name";

}

else if (listBox3.SelectedIndex == 11)

{

InputPeriod ip = new InputPeriod();

ip.ShowDialog(this);

if (ip.GetResult() == null)

return;

int FirstYear, SecondYear;

try

{

FirstYear = Convert.ToInt32(ip.GetResult()[0]);

SecondYear = Convert.ToInt32(ip.GetResult()[1]);

if (FirstYear > SecondYear)

throw new Exception();

}

catch { Message.ErrorShow("Значение указано неверно."); return; }

Query = $"SELECT b.name Книга, COUNT(b.id) Тираж FROM book b WHERE b.date\_public >= {FirstYear} AND b.date\_public <= {SecondYear} GROUP BY (b.name) ORDER BY b.name";

label10.Text = "Период: " + FirstYear.ToString() + "-" + SecondYear.ToString();

}

else if (listBox3.SelectedIndex == 12)

{

it.ShowDialog(this);

if (it.GetResult() == null)

return;

int Cost;

try

{

Cost = Convert.ToInt32(it.GetResult());

}

catch { return; }

Query = $"SELECT s.id, s.name Магазин, SUM(d.cost\*d.count\_book) \"Стоимость закупок\", COUNT(d.id) \"Количество поставок\" FROM deliveries d " +

"JOIN shop s ON s.id = d.id\_shop " +

"GROUP BY(s.id) " +

$"HAVING SUM(d.cost\* d.count\_book) > {Cost} " +

"ORDER BY(s.name)";

label10.Text = "Общая стоимость превышает: " + Cost.ToString();

}

else if (listBox3.SelectedIndex == 13)

{

it.ShowDialog(this);

if (it.GetResult() == null)

return;

int Cost;

try

{

Cost = Convert.ToInt32(it.GetResult());

}

catch { return; }

InputDates id = new InputDates();

id.ShowDialog(this);

if (id.GetResult() == null)

return;

string[] date = id.GetResult();

Query = $"SELECT s.id, s.name Магазин, SUM(d.cost\*d.count\_book) \"Стоимость закупок\", COUNT(d.id) \"Количество поставок\" FROM deliveries d " +

"JOIN shop s ON s.id = d.id\_shop " +

$"WHERE d.date\_come >= '{date[0]}' AND d.date\_come <= '{date[1]}' " +

"GROUP BY(s.id) " +

$"HAVING SUM(d.cost\* d.count\_book) > {Cost} " +

"ORDER BY(s.name)";

label10.Text = "Общая стоимость превышает: " + Cost.ToString();

}

else if (listBox3.SelectedIndex == 14)

{

Query = "SELECT s.name, COUNT(s.id) \"Количество магазинов\", MAX(dc.count) \"Количество поставок\" FROM shop s "+

"JOIN(SELECT s.name, COUNT(d.id) FROM deliveries d JOIN shop s ON s.id = " +

"d.id\_shop GROUP BY(s.name)) dc ON s.name = dc.name GROUP BY (s.name)";

}

else if (listBox3.SelectedIndex == 15)

{

Query = "SELECT p.name Название, c.name Город, p.phone Телефон, p.date\_create \"Дата создания\" FROM publisher p JOIN city c ON p.id\_city = c.id WHERE p.date\_create > (SELECT AVG(p.date\_create) FROM publisher p)";

}

else if (listBox3.SelectedIndex == 16)

{

it.ShowDialog(this);

if (it.GetResult() == null)

return;

string Value;

try

{

Value = InputData.CheckString(it.GetResult());

}

catch { return; }

Query = "SELECT d.id\_shop, s.name Магазин, ROUND(COUNT(DISTINCT d.id\_book)/ "+

"(SELECT COUNT(DISTINCT d.id\_book) FROM deliveries d "+

"JOIN shop s ON s.id = d.id\_shop "+

"JOIN area a ON a.id = s.id\_area "+

$"WHERE EXTRACT(YEAR FROM d.date\_come) >= 2019 AND a.name = {Value})::numeric\*100, 2) Процент " +

"FROM deliveries d "+

"JOIN shop s ON s.id = d.id\_shop "+

"JOIN area a ON a.id = s.id\_area "+

$"WHERE EXTRACT(YEAR FROM d.date\_come) >= 2019 AND a.name = {Value} "+

"GROUP BY(d.id\_shop, s.name, a.id)";

label10.Text = "Район: " + Value;

}

else if(listBox3.SelectedIndex == 17)

{

Query = "SELECT a.name Район, ROUND(COUNT(DISTINCT d.id\_book)/" +

"(SELECT COUNT (DISTINCT d.id\_book) FROM deliveries d WHERE EXTRACT(YEAR FROM d.date\_come) >= 2019)::numeric\*100, 2) Процент " +

"FROM deliveries d " +

"JOIN shop s ON s.id=d.id\_shop " +

"JOIN area a ON a.id=s.id\_area " +

"WHERE EXTRACT(YEAR FROM d.date\_come) >= 2019 " +

"GROUP BY(a.name)";

}

else if(listBox3.SelectedIndex == 18)

{

Query = "SELECT p.name Издательство, ROUND(COUNT(DISTINCT b.id)/COUNT(p.id)::numeric, 2) Количество " +

"FROM deliveries d " +

"JOIN book b ON b.id = d.id\_book " +

"JOIN publisher p ON p.id = b.id\_publisher " +

"GROUP BY(p.name) " +

"ORDER BY(p.name)";

}

else if(listBox3.SelectedIndex == 19)

{

Query = "SELECT COUNT(DISTINCT b.id)/COUNT(DISTINCT p.id) Количество "+

"FROM deliveries d "+

"JOIN book b ON b.id = d.id\_book "+

"JOIN publisher p ON p.id = b.id\_publisher";

}

else if(listBox3.SelectedIndex == 20)

{

Query = "SELECT s.id, s.name Магазин, EXTRACT(YEAR from d.date\_come) Год, EXTRACT(MONTH from d.date\_come) Месяц, SUM(d.count\_book) Количество, SUM(d.count\_book\*d.cost) Стоимость FROM deliveries d "+

"JOIN shop s ON s.id = d.id\_shop "+

"WHERE EXTRACT(YEAR from d.date\_come) >= 2018 AND EXTRACT(YEAR from d.date\_come) <= 2018 "+

"GROUP BY(s.id, EXTRACT(YEAR from d.date\_come), EXTRACT(MONTH from d.date\_come)) "+

"ORDER BY(s.id, EXTRACT(YEAR from d.date\_come), EXTRACT(MONTH from d.date\_come))";

}

if (Query == null)

return;

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

NpgsqlDataAdapter nda = new NpgsqlDataAdapter(Query, SQL.GetConnection());

DataTable dt = new DataTable();

nda.Fill(dt);

dataGridView3.DataSource = dt;

connect.Close();

}

label4.Text = "Количество записей: " + dataGridView3.Rows.Count.ToString();

}

private void Main\_FormClosing(object sender, FormClosingEventArgs e)

{

Application.Exit();

}

private void button1\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedItem == null) return;

else

{

List<int> Id = new List<int>();

foreach(DataGridViewRow row in dataGridView1.SelectedRows)

{

Id.Add(Convert.ToInt32(dataGridView1[0, row.Index].Value.ToString()));

}

Table.ReturnTable(listBox1.SelectedItem.ToString()).Delete(Id.ToArray());

UpdateDatagrid(dataGridView1, listBox1, label1);

}

}

void UpdateDatagrid(DataGridView dgv, ListBox lb, Label count\_nodes)

{

if (lb.SelectedItem == null) return;

else

{

DataTable dt = new DataTable();

Table.ReturnTable(lb.SelectedItem.ToString()).Select().Fill(dt);

dgv.DataSource = dt;

count\_nodes.Text = "Количество записей: " + dgv.Rows.Count;

}

}

private void listBox1\_MouseClick(object sender, MouseEventArgs e)

{

if (listBox1.SelectedItem.ToString() == "Книги-авторы")

{

button2.Enabled = false;

button9.Enabled = false;

}

else { button2.Enabled = true; button9.Enabled = true; }

if (listBox1.SelectedItem.ToString() == "Книги")

button8.Enabled = true;

else button8.Enabled = false;

if (listBox1.SelectedItem.ToString() == "Магазины" || listBox1.SelectedItem.ToString() == "Издательства")

button12.Enabled = true;

else button12.Enabled = false;

UpdateDatagrid(dataGridView1, listBox1, label1);

}

private void AreaCreateColumns()

{

DataTable dt = new DataTable();

dt.Columns.Add("id");

dt.Columns.Add("Адресс");

dataGridView1.DataSource = dt;

}

private void tabPage1\_Click(object sender, EventArgs e)

{

}

private void listBox2\_Click(object sender, EventArgs e)

{

UpdateDatagrid(dataGridView2, listBox2, label9);

}

private void tabPage2\_MouseClick(object sender, MouseEventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

if (listBox2.SelectedItem == null) return;

if (Message.DeleteWarning() == DialogResult.OK)

{

List<int> Id = new List<int>();

foreach (DataGridViewRow row in dataGridView2.SelectedRows)

{

Id.Add(Convert.ToInt32(dataGridView2[0, row.Index].Value.ToString()));

}

Table.ReturnTable(listBox2.SelectedItem.ToString()).Delete(Id.ToArray());

UpdateDatagrid(dataGridView2, listBox2, label9);

}

}

private void button6\_Click(object sender, EventArgs e)

{

new Thread(() =>

{

Table.Generate();

Action action = () => { UpdateDatagrid(dataGridView2, listBox2, label9); UpdateDatagrid(dataGridView1, listBox1, label1); };

Invoke(action);

}).Start();

}

private void button7\_Click(object sender, EventArgs e)

{

Table.TruncateAll();

UpdateDatagrid(dataGridView2, listBox2, label9);

UpdateDatagrid(dataGridView1, listBox1, label1);

}

private void button8\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 0)

return;

string image = "images/" + dataGridView1[2, dataGridView1.SelectedRows[0].Index].Value.ToString();

if (!File.Exists(image))

{

Message.ErrorShow("Картинка была удалена из каталога");

return;

}

new PictureViewer(image).Show();

}

private void button9\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedItem == null || dataGridView1.SelectedRows.Count == 0)

return;

if (listBox1.SelectedItem.ToString() == "Магазины")

new EditShop(Convert.ToInt32(dataGridView1[0, dataGridView1.SelectedRows[0].Index].Value.ToString()), dataGridView1[1, dataGridView1.SelectedRows[0].Index].Value.ToString(), Convert.ToInt32(dataGridView1[2, dataGridView1.SelectedRows[0].Index].Value.ToString()), dataGridView1[4, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[3, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[5, dataGridView1.SelectedRows[0].Index].Value.ToString()).ShowDialog(this);

else if (listBox1.SelectedItem.ToString() == "Поставки")

new EditDeliveries(dataGridView1[0, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[1, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[2, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[3, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[4, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[5, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[6, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[7, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[8, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[9, dataGridView1.SelectedRows[0].Index].Value.ToString()).ShowDialog(this);

else if (listBox1.SelectedItem.ToString() == "Книги")

new EditBook(dataGridView1[0, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[1, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[2, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[3, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[4, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[5, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[6, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[7, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[8, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[9, dataGridView1.SelectedRows[0].Index].Value.ToString()).ShowDialog(this);

else if (listBox1.SelectedItem.ToString() == "Издательства")

new EditPublisher(Convert.ToInt32(dataGridView1[0, dataGridView1.SelectedRows[0].Index].Value.ToString()), dataGridView1[1, dataGridView1.SelectedRows[0].Index].Value.ToString(), dataGridView1[2, dataGridView1.SelectedRows[0].Index].Value.ToString(), Convert.ToInt32(dataGridView1[4, dataGridView1.SelectedRows[0].Index].Value.ToString()), dataGridView1[3, dataGridView1.SelectedRows[0].Index].Value.ToString()).ShowDialog(this);

UpdateDatagrid(dataGridView1, listBox1, label1);

}

private void button10\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedItem == null)

return;

Searcher search = new Searcher(dataGridView1);

search.ShowDialog(this);

int[] indexes = search.GetIndexes();

string text = search.GetText();

if (indexes == null)

return;

UpdateDatagrid(dataGridView1, listBox1, label1);

DataTable dt = new DataTable();

foreach (DataGridViewColumn column in dataGridView1.Columns)

dt.Columns.Add(column.HeaderText);

foreach (DataGridViewRow node in dataGridView1.Rows)

{

bool find = true;

foreach (int index in indexes)

{

if (!(node.Cells[index].Value.ToString() == text))

find = false;

}

if (find)

dt.Rows.Add(CreateNode(node));

}

dataGridView1.DataSource = dt;

label1.Text = "Количество записей: " + dataGridView1.Rows.Count.ToString();

string[] CreateNode(DataGridViewRow node)

{

string[] n = new string[dataGridView1.ColumnCount];

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

n[i] = node.Cells[i].Value.ToString();

return n;

}

}

private void button11\_Click(object sender, EventArgs e)

{

if (listBox1.SelectedItem == null)

return;

Deleter delete = new Deleter(dataGridView1);

delete.ShowDialog(this);

int index = delete.GetIndex();

string text = delete.GetText();

if(index != -1 && text != "")

MainTable.ReturnMainTable(listBox1.SelectedItem.ToString()).ColumnDelete(dataGridView1.Columns[index].HeaderText, text);

}

private void button12\_Click(object sender, EventArgs e)

{

string Column = null;

foreach (DataGridViewColumn DataCol in dataGridView1.Columns)

{

Column = MainTable.ReturnMainTable(listBox1.SelectedItem.ToString()).ConvertColumnName(DataCol.HeaderText);

if (Column != null)

break;

}

if (Column == null)

return;

new Viewer(Column, MainTable.ReturnMainTable(listBox1.SelectedItem.ToString()).ClassName).Show();

}

private void button13\_Click(object sender, EventArgs e)

{

chart1.Series.Clear();

chart1.Series.Add("Диаграмма");

if (listBox4.SelectedItem == null)

return;

if (listBox4.SelectedIndex == 0)

PieDiagramm();

else if (listBox4.SelectedIndex == 1)

ColumnDiagramm();

else if (listBox4.SelectedIndex == 2)

ThreeColumnDiagramm();

}

private void ThreeColumnDiagramm()

{

listBox5.Visible = false;

var count = chart1.Series.Add("Стоимость закупок для магазинов");

var shopcost = chart1.Series.Add("Стоимость закупок для поставщиков");

var defcost = chart1.Series.Add("Разница в цене");

using (NpgsqlConnection connect = SQL.GetConnection())

{

string Query = "SELECT EXTRACT(YEAR FROM d.date\_come),SUM(d.cost), SUM(d.def\_cost), SUM(d.cost) - SUM(d.def\_cost) FROM deliveries d GROUP BY (EXTRACT(YEAR FROM d.date\_come)) ORDER BY (EXTRACT(YEAR FROM d.date\_come));";

connect.Open();

try

{

NpgsqlCommand command = new NpgsqlCommand(Query, connect);

command.ExecuteNonQuery();

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

count.Points.AddXY(reader.GetDouble(0), reader.GetDouble(1));

shopcost.Points.AddXY(reader.GetDouble(0), reader.GetDouble(2));

defcost.Points.AddXY(reader.GetDouble(0), reader.GetDouble(3));

}

}

catch (Npgsql.PostgresException ex)

{

}

connect.Close();

}

}

private void PieDiagramm()

{

listBox5.Visible = false;

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Pie;

chart1.Series[0].Points.Clear();

using (NpgsqlConnection connect = SQL.GetConnection())

{

string Query = "SELECT o.name, (COUNT(o.id)/(SELECT COUNT(\*) FROM shop s)::float) FROM shop s JOIN own o ON o.id = s.id\_own GROUP BY(o.name)";

connect.Open();

try

{

NpgsqlCommand command = new NpgsqlCommand(Query, connect);

command.ExecuteNonQuery();

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

chart1.Series[0].Points.AddY(reader.GetDouble(1));

chart1.Series[0].Points[chart1.Series[0].Points.Count - 1].LegendText = reader.GetString(0) + ": " + Math.Round(reader.GetDouble(1)\*100, 2).ToString() + "%";

}

}

catch (Npgsql.PostgresException ex)

{

}

connect.Close();

}

}

private void ColumnDiagramm()

{

chart1.Series[0].ChartType = System.Windows.Forms.DataVisualization.Charting.SeriesChartType.Column;

chart1.Series[0].Points.Clear();

using (NpgsqlConnection connect = SQL.GetConnection())

{

string Query = "SELECT c.name, COUNT(p.id) FROM city c JOIN publisher p ON p.id\_city = c.id GROUP BY(c.name) ORDER BY c.name";

connect.Open();

try

{

NpgsqlCommand command = new NpgsqlCommand(Query, connect);

command.ExecuteNonQuery();

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

chart1.Series[0].Points.AddY(reader.GetInt32(1));

chart1.Series[0].Points[chart1.Series[0].Points.Count - 1].LegendText = reader.GetString(0);

}

foreach (DataPoint dp in chart1.Series[0].Points)

{

dp.Color = System.Drawing.Color.SkyBlue;

}

}

catch (Npgsql.PostgresException ex)

{

}

connect.Close();

}

listBox5.Visible = true;

using (NpgsqlConnection connect = SQL.GetConnection())

{

listBox5.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT c.name FROM city c JOIN publisher p ON p.id\_city = c.id GROUP BY(c.name) ORDER BY c.name", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

listBox5.Items.Add(reader.GetString(0));

}

connect.Close();

}

}

private void listBox5\_MouseClick(object sender, MouseEventArgs e)

{

foreach(DataPoint dp in chart1.Series[0].Points)

{

dp.Color = System.Drawing.Color.SkyBlue;

}

chart1.Series[0].Points[listBox5.SelectedIndex].Color = System.Drawing.Color.Red;

}

}

}

Deleter.cs

using System;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class Deleter : Form

{

public Deleter(DataGridView dgv)

{

InitializeComponent();

for (int i = 0; i < dgv.ColumnCount; i++)

listBox1.Items.Add(dgv.Columns[i].HeaderText);

}

string Text;

int Index; //Колонки для поиска

private void button1\_Click(object sender, EventArgs e)

{

Index = listBox1.SelectedIndex;

Text = textBox1.Text;

Hide();

}

public int GetIndex() { return Index; }

public string GetText() { return Text; }

}

}

EditAuthor.cs

using System;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class EditAuthor : Form

{

public EditAuthor()

{

InitializeComponent();

}

string Name, SecondName, LastName;

private void button1\_Click(object sender, EventArgs e)

{

Name = InputData.CheckString(textBox1.Text);

SecondName = InputData.CheckString(textBox2.Text);

LastName = InputData.CheckString(textBox3.Text);

if (new Author(Name, SecondName, LastName).Insert())

{

Message.Success();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

}

}

}

EditBook.cs

using System;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using Npgsql;

namespace Kursovaya

{

public partial class EditBook : Form

{

public EditBook()

{

InitializeComponent();

}

public EditBook(string id, string name, string photo, string description, string lang, string publisher, string style, string binding, string date, string publishDate)

{

InitializeComponent();

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

NpgsqlCommand command = new NpgsqlCommand($"SELECT p.id FROM book b JOIN publisher p ON p.id = b.id\_publisher WHERE b.id = {id}", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

publisher = reader.GetValue(0) + " " + publisher;

}

connect.Close();

}

this.id = Convert.ToInt32(id);

textBox1.Text = name;

textBox2.Text = description;

textBox3.Text = publishDate.ToString();

textBox4.Text = date.ToString();

comboBox1.Text = publisher;

comboBox2.Text = binding;

comboBox4.Text = lang;

comboBox5.Text = style;

button1.Text = photo;

this.photo = button1.Text;

State = false;

comboBox3.Visible = false;

label4.Visible = false;

button8.Visible = false;

listBox1.Visible = false;

}

private bool State = true;

private int id;

private string name;

private string photo;

private string description;

private string lang;

private int date;

private string style;

private int publisher;

private string binding;

private int publish\_date;

//Обложка

private void button1\_Click(object sender, EventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "Image files (\*.jpg, \*.jpeg, \*.png) | \*.jpg; \*.jpeg; \*.png";

var state = ofd.ShowDialog();

if (state == DialogResult.Cancel)

return;

if (state == DialogResult.OK)

{

button1.Text = ofd.FileName;

}

}

//Удалить автора из списка

private void button8\_Click(object sender, EventArgs e)

{

for(int i = 0; i < listBox1.SelectedItems.Count; i++)

{

listBox1.Items.Remove(listBox1.SelectedItems[i]);

}

}

//Добавление записи

private void button7\_Click(object sender, EventArgs e)

{

photo = button1.Text;

name = InputData.CheckString(textBox1.Text);

if (InputData.CheckInt(comboBox1.Text.Split(' ')[0], "Издательство"))

publisher = Convert.ToInt32(comboBox1.Text.Split(' ')[0]);

binding = InputData.CheckString(comboBox2.Text);

lang = InputData.CheckString(comboBox4.Text);

style = InputData.CheckString(comboBox5.Text);

InputData.CheckString(photo);

description = InputData.CheckString(textBox2.Text);

if (!InputData.CheckInt(textBox3.Text, "\"Год создания(автором)\""))

return;

if (!InputData.CheckInt(textBox4.Text, "\"Дата публикации(издательством)\""))

return;

date = Convert.ToInt32(textBox4.Text);

publish\_date = Convert.ToInt32(textBox3.Text);

if (date > publish\_date)

{

Message.ErrorShow("Дата публикации не может быть больше даты созданияю.");

return;

}

if (State)

{

if (listBox1.Items.Count == 0)

{

Message.ErrorShow("У книги должны быть авторы");

return;

}

bool success\_authors = false;

bool success\_book = new Book(name, photo, description, lang, date, publisher, style, binding, publish\_date).Insert();

if (!success\_book)

return;

new BookAuthor(name, listBox1.Items.Cast<String>().ToArray()).InsertAllAuthors();

Message.Success();

listBox1.Items.Clear();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";//14635

comboBox1.Text = null;

comboBox2.Text = null;

comboBox3.Text = null;

comboBox4.Text = null;

comboBox5.Text = null;

button1.Text = "Выбрать";

photo = "0";

lang = null;

style = null;

binding = null;

}

else

{

new Book(id, name, photo, description, lang, date, publisher, style, binding, publish\_date).Update();

//new BookAuthor(id, listBox1.Items.Cast<String>().ToArray()).Update();

}

}

private void comboBox1\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox1.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT id, name FROM publisher;", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add($"{reader.GetValue(0)} {reader.GetValue(1)}");

}

connect.Close();

}

}

private void comboBox2\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox2.Items.Clear();

comboBox2.Items.Add("Добавить");

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT name FROM binding;", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox2.Items.Add(reader.GetString(0));

}

connect.Close();

}

}

private void comboBox3\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox3.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT second\_name, name, last\_name FROM author;", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox3.Items.Add(reader.GetString(0) + " " + reader.GetString(1) + " " + reader.GetString(2));

}

connect.Close();

}

}

private void comboBox4\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox4.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT name FROM lang;", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox4.Items.Add(reader.GetString(0));

}

connect.Close();

}

}

private void comboBox5\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox5.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT name FROM style;", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox5.Items.Add(reader.GetString(0));

}

connect.Close();

}

}

private void comboBox3\_SelectedIndexChanged(object sender, EventArgs e)

{

foreach (string item in listBox1.Items)

{

if (item == comboBox3.Text)

{

MessageBox.Show("Данный автор уже добавлен в список!", "Ошибка013", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

if(comboBox3.Text != null || comboBox3.Text != "")

listBox1.Items.Add(comboBox3.Text);

}

}

}

Deliveries.cs

using Npgsql;

using System;

using System.Drawing;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class EditDeliveries : Form

{

public EditDeliveries()

{

InitializeComponent();

}

public EditDeliveries(string id, string shop, string book, string count, string date, string cost, string firstCost, string lang, string volume, string preOrder)

{

InitializeComponent();

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

NpgsqlCommand command = new NpgsqlCommand($"SELECT s.id FROM deliveries d JOIN shop s ON s.id = d.id\_shop WHERE d.id = {id}", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

shop = reader.GetValue(0) + " " + shop;

}

connect.Close();

}

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

NpgsqlCommand command = new NpgsqlCommand($"SELECT b.id FROM deliveries d JOIN book b ON b.id = d.id\_book WHERE d.id = {id}", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

book = reader.GetValue(0) + " " + book;

}

connect.Close();

}

Id = Convert.ToInt32(id);

textBox1.Text = cost.ToString();

textBox2.Text = firstCost.ToString();

dateTimePicker1.Value = Convert.ToDateTime(date);

numericUpDown1.Value = Convert.ToInt32(count);

textBox3.Text = volume.ToString();

comboBox1.Text = shop;

comboBox2.Text = book;

comboBox3.Text = lang;

if (Convert.ToBoolean(preOrder))

radioButton1.Checked = true;

else radioButton2.Checked = true;

State = false;

}

string Lang, Date;

int Count, Id, Shop, Book;

double Cost, FirstCost, Volume;

bool State = true;

private void comboBox3\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox3.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT name FROM lang", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox3.Items.Add($"{reader.GetString(0)}");

}

connect.Close();

}

}

private void comboBox2\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox2.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT id, name FROM book", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox2.Items.Add($"{reader.GetValue(0)} {reader.GetValue(1)}");

}

connect.Close();

}

}

bool PreOder = false;

private void comboBox1\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox1.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT id, name FROM shop", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add($"{reader.GetValue(0)} {reader.GetValue(1)}");

}

connect.Close();

}

}

private void button4\_Click(object sender, EventArgs e)

{

if(!InputData.CheckDouble(textBox1.Text, "Цена книги(для магазина)"))

return;

if (!InputData.CheckDouble(textBox2.Text, "Цена книги(поставщика)"))

return;

if (!InputData.CheckDouble(textBox3.Text, "Объем"))

return;

if (radioButton1.Checked == true)

PreOder = true;

else PreOder = false;

Shop = Convert.ToInt32(comboBox1.Text.Split(' ')[0]);

Book = Convert.ToInt32(comboBox2.Text.Split(' ')[0]);

Lang = InputData.CheckString(comboBox3.Text);

Cost = Convert.ToDouble(textBox1.Text);

FirstCost = Convert.ToDouble(textBox2.Text);

Date = InputData.CheckString(dateTimePicker1.Value.ToShortDateString());

Count = Convert.ToInt32(numericUpDown1.Value);

Volume = Convert.ToDouble(textBox3.Text);

if(Cost < FirstCost)

{

Message.ErrorShow("Цена для магазина не может быть меньше цены для поставщика.");

return;

}

if (State)

{

bool success = new Deliveries(Shop, Book, Lang, Count, Date, Cost, Volume, FirstCost, PreOder).Insert();

if (success)

{

Message.Success();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

comboBox1.Text = null;

comboBox2.Text = null;

comboBox3.Text = null;

}

}

else

{

new Deliveries(Id,Shop, Book, Lang, Count, Date, Cost, Volume, FirstCost, PreOder).Update();

}

}

}

}

EditPublisher.cs

using System;

using System.Windows.Forms;

using Npgsql;

namespace Kursovaya

{

public partial class EditPublisher : Form

{

int State;

public EditPublisher()

{

InitializeComponent();

State = 1;

}

public EditPublisher(int id, string name, string city, int date, string phone)

{

InitializeComponent();

State = 2;

textBox1.Text = name;

textBox2.Text = date.ToString();

textBox3.Text = phone;

this.city = city;

this.id = id;

}

private int id;

private string name;

private string telephone;

private string city = "NULL";

private int date;

private void button2\_Click(object sender, EventArgs e)

{

name = InputData.CheckString(textBox1.Text);

telephone = InputData.CheckString(textBox3.Text);

city = InputData.CheckString(comboBox1.Text);

if (!InputData.CheckInt(textBox2.Text, "\"Дата создания\""))

return;

date = Convert.ToInt32(textBox2.Text);

if (this.State == 1)

{

bool success = new Publisher(name, city, telephone, date).Insert();

if (success)

{

Message.Success();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

}

else

new Publisher(id, name, city, telephone, date).Update();

}

private void Clear()

{

textBox1.Text = null;

textBox2.Text = null;

textBox3.Text = null;

}

public string GetResult() { return this.name; }

private void comboBox1\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox1.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT name FROM city;", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(reader.GetString(0));

}

connect.Close();

}

}

}

}

EditShop.cs

using System;

using System.Drawing;

using System.Windows.Forms;

using Npgsql;

namespace Kursovaya

{

public partial class EditShop : Form

{

public EditShop()

{

InitializeComponent();

}

public EditShop(int id, string name, int date, string area, string address, string own)

{

InitializeComponent();

Id = id;

Title = name;

Date = date;

Area = area;

Address = address;

Own = own;

State = false;

textBox1.Text = Title;

textBox2.Text = Address;

textBox3.Text = Date.ToString();

comboBox1.Text = Area;

comboBox2.Text = Own;

}

string Area, Own, Title, Address;

int Id, Date;

bool State = true;

private void Clear()

{

this.textBox1.Clear();

this.textBox2.Clear();

this.textBox3.Clear();

this.comboBox2.Items.Clear();

this.comboBox1.Items.Clear();

}

private void button3\_Click(object sender, EventArgs e)

{

if (!InputData.CheckInt(textBox3.Text, "\"Дата открытия\""))

return;

Area = InputData.CheckString(comboBox1.Text);

Own = InputData.CheckString(comboBox2.Text);

Date = Convert.ToInt32(textBox3.Text);

Title = InputData.CheckString(textBox1.Text);

Address = InputData.CheckString(textBox2.Text);

bool success = false;

if (State)

{

success = new Shop(Title, Date, Area, Address, Own).Insert();

if (success)

{

Message.Success(); this.Clear();

}

}

else

new Shop(Id, Title, Date, Area, Address, Own).Update();

}

private void comboBox2\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox2.Items.Clear();

//comboBox2.Items.Add("Добавить");

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT name FROM own;", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox2.Items.Add(reader.GetString(0));

}

connect.Close();

}

}

private void comboBox1\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox1.Items.Clear();

//comboBox1.Items.Add("Добавить");

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT name FROM area;", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(reader.GetString(0));

}

connect.Close();

}

}

}

}

EditSup.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Npgsql;

namespace Kursovaya

{

public partial class EditSup : Form

{

public EditSup(string type)

{

InitializeComponent();

this.type = type;

}

string type, edit = "Создание";

private void EditSup\_Load(object sender, EventArgs e)

{

label1.Text = edit + " записи в таблице " + type;

label2.Text = type + ':';

if (edit == "Создание")

button1.Text = "Создать";

}

private void button1\_Click(object sender, EventArgs e)

{

string text = InputData.CheckString(textBox1.Text);

Table sup = Table.ReturnTable(type);

if (type == "Район")

sup = new Area(text);

else if (type == "Тип собственности")

sup = new Own(text);

else if (type == "Язык")

sup = new Lang(text);

else if (type == "Город")

sup = new City(text);

else if (type == "Жанр")

sup = new Style(text);

else if (type == "Тип переплета")

sup = new Binding(text);

if (sup.Insert())

{

textBox1.Text = "";

Message.Success();

}

}

private void EditSup\_FormClosing(object sender, FormClosingEventArgs e)

{

if(e.CloseReason != new CloseReason())

{

textBox1.Text = null;

}

}

public string GetResult() { return textBox1.Text; }

}

}

InputAuthor.cs

using Npgsql;

using System;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class InputAuthor : Form

{

public InputAuthor()

{

InitializeComponent();

}

string Author;

private void button1\_Click(object sender, EventArgs e)

{

Author = comboBox1.Text;

Hide();

}

private void comboBox1\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox1.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand("SELECT second\_name, name, last\_name FROM author;", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(reader.GetString(0) + " " + reader.GetString(1) + " " + reader.GetString(2));

}

connect.Close();

}

}

public string GetResult() { return Author; }

}

}

InputDates.cs

using System;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class InputDates : Form

{

public InputDates()

{

InitializeComponent();

}

string Date1, Date2;

private void button1\_Click(object sender, EventArgs e)

{

Date1 = dateTimePicker1.Value.ToShortDateString();

Date2 = dateTimePicker2.Value.ToShortDateString();

Hide();

}

public string[] GetResult() { return new string[] {Date1, Date2 }; }

}

}

InputPeriod.cs

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class InputPeriod : Form

{

public InputPeriod()

{

InitializeComponent();

}

string[] TextT;

private void button1\_Click(object sender, EventArgs e)

{

TextT = new string[2];

TextT[0] = textBox1.Text;

TextT[1] = textBox2.Text;

Hide();

}

public string[] GetResult() { return TextT; }

}

}

InputText.cs

using System;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class InputText : Form

{

public InputText()

{

InitializeComponent();

}

string Text;

private void button1\_Click(object sender, EventArgs e)

{

Text = textBox1.Text;

Hide();

}

public string GetResult() { return Text; }

}

}

PictureViewer.cs

using System.Drawing;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class PictureViewer : Form

{

public PictureViewer(string filename)

{

InitializeComponent();

Bitmap bitmap = new Bitmap(filename);

if (bitmap.Width > 700 || bitmap.Height > 500)

{

Size size = new Size(bitmap.Width / 4, bitmap.Height / 4);

bitmap = new Bitmap(bitmap, size);

}

pictureBox1.Image = (Image)bitmap;

pictureBox1.Size = pictureBox1.Image.Size;

this.Size = pictureBox1.Size;

}

private void pictureBox1\_Click(object sender, System.EventArgs e)

{

}

}

}

Searcher.cs

using System;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class Searcher : Form

{

public Searcher(DataGridView dgv)

{

InitializeComponent();

Columns = new string[dgv.Columns.Count];

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

Columns[i] = dgv.Columns[i].HeaderText;

}

string[] Columns;

string Text;

int[] SearchIndexes; //Колонки для поиска

private void button2\_Click(object sender, EventArgs e)

{

SetIndexes();

Text = textBox1.Text;

Hide();

}

public int[] GetIndexes() { return SearchIndexes; }

public string GetText() { return Text; }

private void SetIndexes()

{

SearchIndexes = new int[listBox1.Items.Count];

int n = 0;

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

{

foreach (string item in listBox1.Items)

{

if (Columns[i] == item)

{

SearchIndexes[n] = i;

n++;

}

}

}

}

private void comboBox1\_MouseClick(object sender, MouseEventArgs e)

{

comboBox1.Items.Clear();

foreach (string header in Columns)

comboBox1.Items.Add(header);

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

foreach (string item in listBox1.Items)

{

if (item == comboBox1.Text)

{

MessageBox.Show("Данное поле уже добавлен в список.", "Ошибка013", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

}

listBox1.Items.Add(comboBox1.Text);

}

private void button1\_Click(object sender, EventArgs e)

{

for (int i = 0; i < listBox1.SelectedItems.Count; i++)

{

listBox1.Items.Remove(listBox1.SelectedItems[i]);

Viewer.cs

using Npgsql;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Kursovaya

{

public partial class Viewer : Form

{

public Viewer(string columnName, string table)

{

InitializeComponent();

Column = columnName;

Table = table;

}

string Column;

string Table;

private void button1\_Click(object sender, EventArgs e)

{

if (comboBox1.Text == null)

return;

string Query = null;

if (Column == "area")

Query = $"SELECT s.id ID, s.name Название, s.date\_open \"Дата открытия\", s.address Адресс, o.name Собственность FROM shop s JOIN own o ON s.id\_own = o.id WHERE s.id\_area = (SELECT id FROM area WHERE name = \'{comboBox1.Text}\') ";

else if (Column == "city")

Query = $"SELECT p.id ID, p.name Название, p.date\_create \"Дата создания\", p.phone Телефон FROM publisher p WHERE p.id\_city = (SELECT id FROM city WHERE name = \'{comboBox1.Text}\')";

if (Query == null)

return;

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

NpgsqlDataAdapter nda = new NpgsqlDataAdapter(Query, SQL.GetConnection());

DataTable dt = new DataTable();

nda.Fill(dt);

dataGridView1.DataSource = dt;

connect.Close();

}

label1.Text = "Количество записей: " + dataGridView1.Rows.Count.ToString();

}

private void comboBox1\_MouseClick(object sender, MouseEventArgs e)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

comboBox1.Items.Clear();

connect.Open();

NpgsqlCommand command = new NpgsqlCommand($"SELECT name FROM {Column};", connect);

NpgsqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(reader.GetString(0));

}

connect.Close();

}

}

}

}

SQL.cs

using System;

using System.Windows.Forms;

using Npgsql;

namespace Kursovaya

{

class SQL

{

static string login, password;//postgres, 123321

public SQL() { }

public SQL(string login, string password)

{

SQL.login = login;

SQL.password = password;

}

public static NpgsqlConnection GetConnection()

{

return new NpgsqlConnection(@"Server=localhost;Port=5432;User Id="+login+";Password="+password+";Database=Kursovaya;");

return new NpgsqlConnection(@"Server=localhost;Port=5432;User Id=postgres;Password=1956;Database=Kursovaya;");

}

}

}

Program.cs

using System;

using System.Windows.Forms;

namespace Kursovaya

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

Table.cs

using Npgsql;

using System.Collections.Generic;

using System.Windows.Forms;

using System.IO;

namespace Kursovaya

{

abstract class Table

{

protected string TruncateQuery => $"TRUNCATE TABLE {ClassName} RESTART IDENTITY CASCADE"; //+

protected string FileGeneratorPath => $"sql/{ClassName}.sql";

public abstract string ClassName { get; }//+

protected string PrimaryKey => "id";//+

protected abstract string InsertQuery { get;} //+

protected abstract string SelectQuery { get; }//+

protected string DeleteQuery => $"DELETE FROM {ClassName} WHERE {PrimaryKey} = ";//+

protected abstract List<string[]> Constraint { get; }//+

protected abstract List<string[]> ColumnError { get; } //+

protected void Truncate()

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

try

{

NpgsqlCommand command = new NpgsqlCommand(TruncateQuery, connect);

command.ExecuteNonQuery();

}

catch { }

connect.Close();

}

}//+

public static void TruncateAll()//+

{

new Area().Truncate();

new Own().Truncate();

new Binding().Truncate();

new City().Truncate();

new Style().Truncate();

new Author().Truncate();

new Lang().Truncate();

}

public NpgsqlDataAdapter Select()

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

NpgsqlDataAdapter nda = new NpgsqlDataAdapter(SelectQuery, SQL.GetConnection());

connect.Close();

return nda;

}

}//+

public bool Insert()

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

bool result;

connect.Open();

try

{

NpgsqlCommand command = new NpgsqlCommand(InsertQuery, connect);

command.ExecuteNonQuery();

result = true;

}

catch (Npgsql.PostgresException ex)

{

SQLError(ex); result = false;

}

connect.Close();

return result;

}

}//+

public void Delete(int[] id)

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

foreach (int ID in id)

{

try

{

NpgsqlCommand command = new NpgsqlCommand(DeleteQuery + ID, connect);

command.ExecuteNonQuery();

}

catch (Npgsql.PostgresException ex) { SQLError(ex); }

}

connect.Close();

Message.Success();

}

}//+

protected void ValidateConstraint(string error)

{

foreach (string[] temp in Constraint)

{

if (error.Contains(temp[0]))

{

ErrorShow($"Неверно указано поле {temp[1]}.");

break;

}

}

}//+

protected void ValidateColumn(string error)

{

foreach (string[] temp in ColumnError)

{

if (error.Contains(temp[0]))

{

ErrorShow($"Неверно указано поле {temp[1]}.");

break;

}

}

}//+

public void SQLError(Npgsql.PostgresException ex)

{

if (ex.ConstraintName != null)

ValidateConstraint(ex.ConstraintName);

else if (ex.ColumnName != null)

ValidateColumn(ex.ColumnName);

else ErrorShow("Не все поля заполнены верно.");

}//+

public static void ErrorShow(string msg) { MessageBox.Show(msg, "Ошибка 010", MessageBoxButtons.OK, MessageBoxIcon.Error); }//+

public static Table ReturnTable(string name)

{

//Основные таблицы

if (name == "Магазины")

return new Shop();

else if (name == "Издательства")

return new Publisher();

else if (name == "Поставки")

return new Deliveries();

else if (name == "Книги")

return new Book();

else if(name == "Книги-авторы")

return new BookAuthor();

//Справочники

else if (name == "Район")

return new Area();

else if (name == "Автор")

return new Author();

else if (name == "Город")

return new City();

else if (name == "Язык")

return new Lang();

else if (name == "Тип собственности")

return new Own();

else if (name == "Жанр")

return new Style();

else if (name == "Тип переплета")

return new Binding();

else return new Author();

}

public static void Generate()

{

if (Message.Warning("Для генерации данных все записи будут удалены. Продолжить?") == DialogResult.Cancel)

return;

TruncateAll();

new Area().GenerateTable();

new Lang().GenerateTable();

new Own().GenerateTable();

new Binding().GenerateTable();

new City().GenerateTable();

new Style().GenerateTable();

new Shop().GenerateTable();

new Publisher().GenerateTable();

new Book().GenerateTable();

new Author().GenerateTable();

new BookAuthor().GenerateTable();

new Deliveries().GenerateTable();

Message.Success();

}//+

protected bool GenerateTable()

{

if (!File.Exists(FileGeneratorPath))

{

Message.ErrorShow("Присутствуют не все файлы генерации.");

return false;

}

string GenerateQuery = File.ReadAllText(FileGeneratorPath);

if (GenerateQuery == null)

{

Message.ErrorShow("Один из файлов генерации пуст.");

return false;

}

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

try

{

NpgsqlCommand command = new NpgsqlCommand(GenerateQuery, connect);

command.ExecuteNonQuery();

}

catch (Npgsql.PostgresException ex) { }

connect.Close();

}

return true;

}//+

}

}

Author.cs

using System;

using System.Collections.Generic;

namespace Kursovaya

{

class Author : MainTable

{

public override string ClassName => "author";

string Name, SecondName, LastName;

public Author() { }

public Author(string name, string second\_name, string last\_name)

{

Name = name;

SecondName = second\_name;

LastName = last\_name;

}

protected override string InsertQuery => $"INSERT INTO {ClassName} (name, second\_name, last\_name) VALUES ({Name}, {SecondName}, {LastName})";

protected override string SelectQuery => $"SELECT id ID, second\_name Фамилия, name Имя, last\_name Отчество FROM {ClassName}";

protected override List<string[]> Constraint => new List<string[]> {

new string[]{"fio\_uniq","\"Уникальность\"" } };

protected override List<string[]> ColumnError => new List<string[]> {

new string[]{"last\_name","\"Отчество\"" },

new string[]{"second\_name","\"Имя\""},

new string[]{"name","\"Фамилия\""}

};

protected override string[][] TableNames => throw new NotImplementedException();

}

}

Book.cs

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Kursovaya

{

class Book : MainTable

{

public override string ClassName => "book";

string Name, Description, Lang, Style, Binding;

int PublishDate, Date, Id, PublisherId;

string Photo;

public Book() { }

public Book(string name, string photo, string description, string lang, int date, int publisher\_id, string style, string binding, int pub\_date)

{

Name = name;

Description = description;

Lang = lang;

Date = date;

PublisherId = publisher\_id;

Style = style;

Binding = binding;

PublishDate = pub\_date;

if (photo == null || photo == "NULL")

{

Photo = photo;

return;

}

string[] filename = photo.Split('\\');

string newName = null;

try

{

newName = filename[filename.Length - 1];

File.Copy(photo, $"images/{newName}");

}

catch

{

//newName = $"images/{filename[filename.Length - 1]}{new Random().NextDouble()}";

//File.Copy(photo, newName);

}

Photo = InputData.CheckString(newName);

}

public Book(int id, string name, string photo, string description, string lang, int date, int publisher\_id, string style, string binding, int pub\_date)

{

Id = id;

Name = name;

Description = description;

Lang = lang;

Date = date;

PublisherId = publisher\_id;

Style = style;

Binding = binding;

PublishDate = pub\_date;

if (photo == null || photo == "NULL")

{

Photo = photo;

return;

}

string[] filename = photo.Split('\\');

string newName = null;

try

{

newName = filename[filename.Length - 1];

File.Copy(photo, $"images/{newName}");

}

catch

{

//newName = $"images/{filename[filename.Length - 1]}{new Random().NextDouble()}";

//File.Copy(photo, newName);

}

Photo = InputData.CheckString(newName);

}

protected override string InsertQuery => $"INSERT INTO book (name, photo, description, id\_lang, date\_create, id\_publisher, id\_style, id\_binding, date\_public) VALUES " +

$"({Name}, {Photo}, {Description}, " +

$"(SELECT id FROM lang WHERE name = {this.Lang}), {Date}, " +

$"{PublisherId}, " +

$"(SELECT id FROM style WHERE name = {this.Style})," +

$"(SELECT id FROM binding WHERE name = {this.Binding}), {PublishDate})";

protected override string SelectQuery => $"SELECT b.id id, b.name Название, b.photo Фото, b.description Описание, l.name Язык, p.name Издательство, s.name Жанр, bind.name Переплет, " +

$"b.date\_create \"Дата создания\", b.date\_public \"Дата издания\" FROM book b " +

"JOIN lang l ON b.id\_lang = l.id "+

"JOIN publisher p ON b.id\_publisher = p.id "+

"JOIN style s ON b.id\_style = s.id "+

"JOIN binding bind ON b.id\_binding = bind.id";

protected override string UpdateQuery => $"UPDATE book SET name = {Name},description = {Description}, id\_lang = (SELECT id FROM lang WHERE name = {Lang}), id\_publisher = {PublisherId}, id\_style = (SELECT id FROM style WHERE name = {Style}), id\_binding = (SELECT id FROM binding WHERE name = {Binding}), date\_public = {PublishDate}, date\_create = {Date}, photo = {Photo} WHERE id = {Id} ";

protected override List<string[]> Constraint => new List<string[]> {

new string[]{"UQ\_book", "\"Название, Фото (Уникальность)\""},

new string[]{"date\_public", "\"Дата публикации\""},

new string[]{"date", "\"Дата создания\""}};

protected override List<string[]> ColumnError => new List<string[]> {

new string[]{"name","\"Название\""},

new string[]{"description","\"Описание\""},

new string[]{"style", "\"Жанр\""},

new string[]{"publisher", "\"Издательство\""},

new string[]{"lang", "\"Язык\""},

new string[]{"binding", "\"Тип переплета\""}

};

protected override string[][] TableNames => new string[][]

{

new string[]{ "id", "id"},

new string[]{ "Название", "name"},

new string[]{ "Фото", "photo"},

new string[]{ "Описание", "description"},

new string[]{ "Язык", "id\_lang"},

new string[]{ "Издательство", "id\_publisher"},

new string[]{ "Жанр", "id\_style"},

new string[]{ "Переплет", "id\_binding"},

new string[]{ "Дата создания", "date\_create"},

new string[]{ "Дата издания", "date\_public"},

};

}

}

BookAuthor.cs

using Npgsql;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Kursovaya

{

class BookAuthor : MainTable

{

public override string ClassName => "book\_author";

List<string[]> Authors = new List<string[]>();

string[] Author;

string BookName;

int BookId;

public BookAuthor() { }

public BookAuthor(string book\_name, string[] authors)

{

BookName = book\_name;

foreach (string author in authors)

{

Authors.Add(author.Split(' '));

}

}

public BookAuthor(int id, string[] authors)

{

BookId = id;

foreach (string author in authors)

{

Authors.Add(author.Split(' '));

}

}

protected override string InsertQuery => $"INSERT INTO book\_author (id\_book, id\_author) VALUES " +

$"( (SELECT id FROM book WHERE name = {BookName} LIMIT 1)," +

$" ( SELECT id FROM author WHERE name = '{Author[1]}' AND second\_name = '{Author[0]}' AND last\_name = '{Author[2]}'))";

protected override string SelectQuery => "SELECT ba.id id, b.name Книга, a.second\_name Фамилия, a.name Имя, a.last\_name Отчество FROM book\_author ba "+

"JOIN book b ON ba.id\_book = b.id "+

"JOIN author a ON ba.id\_author = a.id";

protected override List<string[]> Constraint => new List<string[]> {new string[]{ "UQ\_book\_author", "\"Книга-автор (Уникальность)\""},};

protected override List<string[]> ColumnError => null;

protected override string[][] TableNames => throw new NotImplementedException();

public void InsertAllAuthors()

{

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

{

Author = Authors[i];

Insert();

}

}

}

}

Deliveries.cs

using System;

using System.Collections.Generic;

namespace Kursovaya

{

class Deliveries : MainTable

{

string Lang;

bool PreOrder;

string Date;

Double Cost, Size, DefCost;

int Count, Id, Shop, Book;

public Deliveries() { }

public Deliveries(int shop, int book, string lang, int count, string date, double cost, double size, double defcost, bool preorder)

{

Shop = shop;

Book = book;

Lang = lang;

Date = date;

Cost = cost;

Size = size;

DefCost = defcost;

Count = count;

PreOrder = preorder;

}

public Deliveries(int id, int shop, int book, string lang, int count, string date, double cost, double size, double defcost, bool preorder)

{

Id = id;

Shop = shop;

Book = book;

Lang = lang;

Date = date;

Cost = cost;

Size = size;

DefCost = defcost;

Count = count;

PreOrder = preorder;

}

public override string ClassName => "deliveries";

protected override string InsertQuery => $"INSERT INTO deliveries (id\_shop, id\_book, count\_book, date\_come, cost, id\_lang, size, pre\_order, def\_cost) VALUES (" +

$"{Shop}, {Book}," +

$"{Count}, {Date}, {Cost}, (SELECT lang.id FROM lang WHERE lang.name = {Lang}), {Size}, {PreOrder}, {DefCost})";

protected override string SelectQuery => "SELECT d.id id, s.name Магазин, b.name Книга, d.count\_book Количество, d.date\_come \"Дата поступления\", d.cost \"Цена для магазина\", d.def\_cost \"Цена для поставщика\", l.name Язык, d.size Объем, d.pre\_order Предзаказ FROM deliveries d " +

"JOIN shop s ON d.id\_shop = s.id " +

"JOIN book b ON d.id\_book = b.id " +

"JOIN lang l ON d.id\_lang = l.id;";

protected override string UpdateQuery => $"UPDATE {ClassName} SET id\_shop = {Shop}, id\_book = {Book}, count\_book = {Count}, date\_come = {Date}, cost = {Cost}, id\_lang = (SELECT id FROM lang WHERE name = {Lang}), size = {Size}, pre\_order = {PreOrder}, def\_cost = {DefCost} WHERE id = {Id};";

protected override List<string[]> Constraint => new List<string[]> {

new string[]{"count","\"Количество экземпляров\""},

new string[]{"ck\_cost\_value","\"Цена (для магазина)\""},

new string[]{"ck\_size\_value", "\"Объем\""},

new string[]{"ck\_defcost\_value", "\"Цена (для поставщиков)\""},

new string[]{"ck\_date\_come","\"Дата поступления\""},

new string[]{"deliver\_uniq","\"Уникальность\""}

};

protected override List<string[]> ColumnError => new List<string[]> {

new string[]{"shop","\"Магазин\""},

new string[]{"book","\"Книга\""},

new string[]{"lang", "\"Язык\""},

};

protected override string[][] TableNames => new string[][]

{

new string[]{ "id", "id"},

new string[]{ "Магазин", "id\_shop"},

new string[]{ "Книга", "id\_book"},

new string[]{ "Количество", "count\_book"},

new string[]{ "Дата поступления", "date\_come"},

new string[]{ "Цена для магазина", "cost"},

new string[]{ "Цена для поставщика", "def\_cost"},

new string[]{ "Язык", "id\_lang"},

new string[]{ "Объем", "size"},

new string[]{ "Предзаказ", "pre\_order"}

};

}

}

MainTable.cs

using Npgsql;

namespace Kursovaya

{

abstract class MainTable: Table

{

protected virtual string UpdateQuery { get; } //+

public void Update()

{

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

try

{

NpgsqlCommand command = new NpgsqlCommand(UpdateQuery, connect);

command.ExecuteNonQuery();

Message.Success();

}

catch (Npgsql.PostgresException ex)

{

SQLError(ex);

}

connect.Close();

}

}//+

public string ConvertColumnName(string ColumnName)

{

if (ColumnName == "id")

return null;

foreach(string[] item in TableNames)

{

if (ColumnName == item[0] && item[1].Contains("id"))

return item[1].Split('\_')[1];

}

return null;

}

private string ConvertIdToQuery(string column, string value)

{

string TableName = column.Split('\_')[1];

return $"DELETE FROM {ClassName} USING {TableName} WHERE {ClassName}.id\_{TableName} IN (SELECT id FROM {TableName} WHERE name = '{value}')";

}

public bool ColumnDelete(string tableName, string value)

{

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

{

if (tableName != TableNames[i][0])

continue;

string Query;

if (TableNames[i][1] != "id" && TableNames[i][1].Contains("id"))

Query = ConvertIdToQuery(TableNames[i][1], value);

else Query = $"DELETE FROM {ClassName} WHERE {TableNames[i][1]} = '{value}'";

using (NpgsqlConnection connect = SQL.GetConnection())

{

connect.Open();

try

{

NpgsqlCommand command = new NpgsqlCommand(Query, connect);

command.ExecuteNonQuery();

Message.Success();

}

catch (Npgsql.PostgresException ex)

{

SQLError(ex);

connect.Close();

return false;

}

connect.Close();

}

return true;

}

return true;

}

protected virtual string[][] TableNames { get; } = new string[][]

{

new string[]{"id", "id"},

new string[]{"Название", "name"}

};

public static MainTable ReturnMainTable(string name)

{

if (name == "Магазины")

return new Shop();

else if (name == "Издательства")

return new Publisher();

else if (name == "Поставки")

return new Deliveries();

else if (name == "Книги")

return new Book();

else

return new BookAuthor();

}

}

}

Publisher.cs

using System.Collections.Generic;

namespace Kursovaya

{

class Publisher : MainTable

{

public override string ClassName => "publisher";

string Name, Telephone, City;

int Date, Id;

public Publisher() { }

public Publisher(string name, string city, string telephone, int date)

{

Name = name;

City = city;

Telephone = telephone;

Date = date;

}

public Publisher(int id, string name, string city, string telephone, int date)

{

Id = id;

Name = name;

City = city;

Telephone = telephone;

Date = date;

}

protected override string InsertQuery => $"INSERT INTO publisher (name, id\_city, phone, date\_create) VALUES " +

$"({Name}, (SELECT id FROM city WHERE name = {City}), {Telephone} , {Date})";

protected override string SelectQuery => "SELECT p.id id, p.name Название, c.name Город, p.phone Телефон, p.date\_create \"Дата создания\" FROM publisher p "+

" JOIN city c ON p.id\_city = c.id";

protected override string UpdateQuery => $"UPDATE {ClassName} SET name = {Name}, id\_city = (SELECT id FROM city WHERE name = {City}), phone = {Telephone}, date\_create = {Date} WHERE id = {Id}";

protected override List<string[]> Constraint => new List<string[]> {

new string[]{"ck\_create\_date","\"Дата создания\""},

new string[]{ "UQ\_publisher\_name", "\"Название (Уникальность)\"" }

};

protected override List<string[]> ColumnError => new List<string[]> {

new string[]{"name","\"Название\""},

new string[]{"phone","\"Телефон\""},

new string[]{"city", "\"Город\""}

};

protected override string[][] TableNames => new string[][]

{

new string[]{ "id", "id"},

new string[]{ "Город", "id\_city"},

new string[]{ "Телефон", "phone"},

new string[]{ "Дата создания", "date\_create"},

};

}

}

Shop.cs

using System.Collections.Generic;

namespace Kursovaya

{

class Shop : MainTable

{

public override string ClassName => "shop";

private int Id, Date;

private string Name, Area, Address, Own;

public Shop() { }

public Shop(string name, int date, string area, string address, string own)

{

Name = name;

Date = date;

Area = area;

Address = address;

Own = own;

}

public Shop(int id, string name, int date, string area, string address, string own)

{

Id = id;

Name = name;

Date = date;

Area = area;

Address = address;

Own = own;

}

protected override string SelectQuery => "SELECT s.id id, s.name Название, s.date\_open \"Год открытия\", s.address Адрес, a.name Район, o.name Собственность FROM shop s " +

"JOIN area a ON s.id\_area = a.id " +

"JOIN own o ON s.id\_own = o.id";

protected override string InsertQuery => $"INSERT INTO shop (name, date\_open, id\_area, address, id\_own) VALUES " +

$"({Name}, {Date}, " +

$"(SELECT area.id FROM area WHERE area.name = {Area}), " +

$"{Address}, (SELECT own.id FROM own WHERE own.name = {Own}))";

protected override string UpdateQuery => $"UPDATE {ClassName} SET name = {Name}, id\_area = (SELECT id FROM area WHERE name = {Area}), address = {Address}, id\_own = (SELECT id FROM own WHERE name = {Own}), date\_open = {Date} WHERE id = {Id}";

protected override List<string[]> Constraint => new List<string[]> {

new string[]{"date\_open","\"Дата открытия\""},

};

protected override List<string[]> ColumnError => new List<string[]> {

new string[]{"name","\"Название\""},

new string[]{"date\_open","\"Год открытия\""},

new string[]{"own", "\"Тип собственности\""},

new string[]{"address", "\"Адресс\""},

new string[]{"area","\"Район\""},

new string[]{ "UQ\_shop", "\"Уникальность\"" }

};

protected override string[][] TableNames => new string[][]

{

new string[]{ "id", "id"},

new string[]{ "Название", "name"},

new string[]{ "Год открытия", "date\_open"},

new string[]{ "Район", "id\_area"},

new string[]{ "Адрес", "address"},

new string[]{ "Собственность", "id\_own"}

};

}

}

Area.cs

namespace Kursovaya

{

class Area : SupTable

{

public override string ClassName => "area";

public Area() { }

public Area(string value){ Value = value; }

}

}

Binding.cs

namespace Kursovaya

{

class Binding : SupTable

{

public override string ClassName => "binding";

public Binding() { }

public Binding(string value) { Value = value; }

}

}

City.cs

namespace Kursovaya

{

class City : SupTable

{

public override string ClassName => "city";

public City() { }

public City(string value) { Value = value; }

}

}

Lang.cs

namespace Kursovaya

{

class Lang : SupTable

{

public override string ClassName => "lang";

public Lang() { }

public Lang(string value) { Value = value; }

}

}

Own.cs

namespace Kursovaya

{

class Own : SupTable

{

public override string ClassName => "own";

public Own() { }

public Own(string value) { Value = value; }

}

}

Style.cs

namespace Kursovaya

{

class Style : SupTable

{

public override string ClassName => "style";

public Style() { }

public Style(string value) { Value = value; }

}

}

Sup.cs

using System.Collections.Generic;

namespace Kursovaya

{

abstract class SupTable:Table

{

protected string Value { get; set; }

protected override string InsertQuery => $"INSERT INTO {ClassName} (name) VALUES ({Value})";

protected override string SelectQuery => $"SELECT id ID, name Название FROM {ClassName}";

protected override List<string[]> Constraint => new List<string[]> {

new string[]{"name","\"Уникальность\""},

};

protected override List<string[]> ColumnError => new List<string[]> {

new string[]{"name","\"Название\""},

};

}

}