Министерство образования и науки Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения

высшего профессионального образования

**«Владимирский государственный университет**

**имени Александра Григорьевича и Николая Григорьевича Столетовых»**

(МИ (филиал) ВлГУ)

Факультет ИТР

Кафедра ПИн

*КУРСОВАЯ РАБОТА*

по дисциплине Системы управления базами данных

Тема Информационная система издательства

Руководитель

Колпаков А.А.

(оценка)

(подпись) (дата)

Члены комиссии Студент ПИн-121

(группа)

Карпов Д.В.

(подпись) (дата)

(подпись) (дата) (подпись) (дата)

Муром 2023 г.

В работе представлена информационная система издательского дома. Цель исследования заключается в изучении принципов разработки информационных систем и создании программы на их основе.

Для реализации проекта были выбраны язык программирования С# и среда разработки Microsoft Visual Studio. Для хранения данных использовалась СУБД MS SQL Server.

Разработанное приложение обеспечивает возможность просмотра, добавления, удаления, изменения и поиска записей в базе данных. Также проведены тестирование и отладка программы.

The work presents an information system of a publishing house. The aim of the research is to study the principles of information system development and create a program based on them.

The C# programming language and Microsoft Visual Studio development environment were chosen for the project implementation. MS SQL Server was used for data storage.

The developed application provides the ability to view, add, delete, modify, and search records in the database. Testing and debugging of the program were also conducted.

Содержание

[Введение 6](#_Toc122464652)

[1 Анализ технического задания 7](#_Toc122464654)

[1.1 Описание предметной области 7](#_Toc122464655)

[1.2 Формирование требований к программному обеспечению 8](#_Toc122464656)

[1.3 Обоснование выбора средств реализации 8](#_Toc122464657)

[2 Разработка моделей данных 10](#_Toc122464658)

[2.1 Концептуальная модель данных 10](#_Toc122464659)

[2.2 Нормализация таблиц 15](#_Toc122464660)

[2.3 Логическая модель данных 16](#_Toc122464661)

[2.4 Физическая модель данных 18](#_Toc122464662)

[3 Разработка и реализация ИС 20](#_Toc122464663)

[3.1 Создание базы данных и хранимых процедур 20](#_Toc122464664)

[3.2 Классы для работы с данными из базы данных 20](#_Toc122464665)

[3.3 Руководство пользователя 31](#_Toc122464670)

[4 Тестирование ИС 35](#_Toc122464671)

[Заключение 39](#_Toc122464672)

[Список литературы 40](#_Toc122464673)

[Приложение 1. Модели данных 41](#_Toc122464674)

[Приложение 2. Cсылка на Github 42](#_Toc122464675)

[Приложение 3. Снимки окон программы 43](#_Toc122464676)

# Введение

## В настоящее время в большинстве сферах профессиональной деятельности человека не осталось ни одного предприятия или фирмы, не оснащенных современными средствами автоматизации. Поэтому перед их руководителями стоит задача внедрения современных информационных технологий. В частности, это разработка и внедрение автоматизированных систем сбора, обработки и анализа информации, которые позволят облегчить труд работников. Одним из таких видов деятельности является оказание услуг по изданию разнообразной печатной продукции.

С помощью информационной системы издательство сможет урегулировать и усовершенствовать многие рабочие процессы, от формирования заказа до выпуска и сдачи печатной продукции. Помимо этого, информационная система позволит хранить в удобном виде необходимые данные, позволит осуществить быстрый поиск необходимых данных по одному или нескольким критериям, позволит автоматически рассчитать стоимость выполнения заказа, предоставит данные, которые станут основой для принятия дальнейших управленческих решений.

Таким образом, целью курсовой работы является создание информационной системы, автоматизирующей работу издательства.

Для выполнения поставленной цели необходимо выполнить задачи:

1) описать структуру работы издательства, изучить взаимодействия её объектов;

2) построить концептуальную модель данных на основе информации, полученной при изучении работы издательства;

3) провести нормализацию таблиц, разработать логическую модель базы данных, физическую модель данных;

4) разработать основные запросы к базе данных, хранимые процедуры;

5) создать программное обеспечение и провести его тестирование.

# 1 Анализ технического задания

## 1.1 Описание предметной области

Издательская деятельность – производство различной печатной продукции. Издатель заключает соглашение с заказчиком (клиентом) на выполнение заказа. Заказчиком может быть как частное лицо, так и организация. Печатью заказа занимается типография.

Заказ может представлять собой книгу, брошюру, рекламный проспект, буклет, бюллетень для голосования, газету или другой вид издательской продукции. Стоимость печатной продукции зависит от её вида, наценки, материала, цвета и размера. Для создания печатной продукции используется один или несколько материалов: бумага для глубокой печати, газетная бумага, глянцевая бумага, дизайнерская бумага, картон, крафт-бумага, матовая бумага, мелованная бумага, обложечная бумага, офсетная бумага, самокопирующаяся бумага, типографская бумага, форзацная бумага.

Над выполнением заказа трудятся один или несколько сотрудников издательства. Они занимаются различными видами деятельности. Ниже представлены некоторые должности сотрудников и их обязанности: редактор по изображениям, фоторедактор – сотрудник, отвечающий за подбор и редактирование фотографий; верстальщик – специалист, занимающийся подгонкой шрифтов для печати, исправлением текстовых ошибок, форматированием иллюстраций; корректор – работник издательства, исправляющий грамматические (орфографические, пунктуационные, стилистические) ошибки и опечатки; переписыватель – специалист, который перерабатывает исходный текст и создает новую статью с тем же смыслом, но другими словами в соответствии с правилами поисковой системы; художник-иллюстратор – человек, создающий иллюстрации для оформления книг, журналов и других печатных изделий.

## 1.2 Формирование требований к программному обеспечению

Для определения всех требований к информационной системе издательства необходимо определить пользователей и их варианты использования системы. Пользователем информационной системы будет являться специалист по приёму и обработке заказов. Следовательно, разрабатываемая информационная система должна иметь функционал для приёма данных о заказе и их обработке: удаление, изменение, поиск, формирование отчётов.

На основании произведённого анализа предметной области и выявленного пользователя информационной системы были сформированы следующие требования к разрабатываемому приложению:

- Создать базу данных, хранящие необходимую информацию для работы с заказами

- Реализовать добавление, изменение, удаление данных из базы данных;

- Реализовать поиск необходимых данных по одному или нескольким критериям;

- Реализовать вывод моды данных, которые могут стать фундаментом для принятия управленческих решений издательства;

- Реализовать автоматический расчёт стоимость выполнения заказа;

- Реализовать сохранение данных о заказе в файл.

## 1.3 Обоснование выбора средств реализации

Для разработки информационной системы планируется использовать язык программирования C#. C# позволяет создавать различные типы безопасных и надежных приложений, работающих в .NET. C# подходит для создания и использования программных компонентов. С момента своего создания C# был расширен функциями для поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО.

Информационная система будет разрабатываться в Microsoft Visual Studio - интегрированной среде разработки. С ее помощью можно создавать как консольные, так и графические приложения, включая поддержку технологии Windows Forms. Visual Studio включает в себя редактор исходного кода, встроенный отладчик, редактор форм для упрощения создания графического интерфейса приложения, дизайнер классов и дизайнер схемы базы данных.

Для создания информационной системы издательства была выбрана СУБД Microsoft SQL Server, которая обладает множеством преимуществ: простота работы и администрирования, высокоскоростной доступ к данным, безопасное хранение данных в базе данных, работа с хранимыми процедурами, возможность хранить различные типы данных, такие как числовые, строковые, даты, изображения. Еще одним плюсом является то, что работать с данной СУБД можно непосредственно в Microsoft Visual Studio.

На основании вышесказанного, рекомендуемые системные требования для программного продукта следующие:

- Процессор x64 или x86 с тактовой частотой 1ГГц и выше;

- Объем оперативной памяти 512 МБ (минимум), 1 ГБ (рекомендуется);

- Операционные системы: Windows 7 SP1/8.1/10(версии 1607 и выше), Fedora 33+, Debian 9+ и другие операционные системы, поддерживающие .NET Core 3.1.

2 Разработка моделей данных

## 2.1 Концептуальная модель данных

При проектировании базы данных необходимо построить модель данных, основанную на предметной области. Для отображения множества понятий и связей между ними, определяющих смысловую структуру пред-метной области или её конкретного объекта, используется концептуальная модель данных.

В данной работе при построении модели данной предметной области используется ER-диаграмма. ER-диаграмма – это разновидность блок-схемы, где показано, как разные «сущности» (люди, объекты, концепции и так далее) связаны между собой внутри системы. ER-диаграммы чаще всего применяются для проектирования и отладки реляционных баз данных в сфере образования, исследования и разработки программного обеспечения и информационных систем для бизнеса. ER-диаграммы (или ER-модели) полагаются на стандартный набор символов, включая прямоугольники, ромбы, овалы и соединительные линии, для отображения сущностей, их атрибутов и связей. Эти диаграммы устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи — глаголов.

Под сущностью в концептуальной модели данных понимают объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении. Каждая сущность имеет свой набор атрибутов. Атрибуты представляют собой свойства, характеризующие сущность.

Между сущностями существует несколько видов связи, а именно:

- Связь один-к-одному. В этой связи объекту одной сущности можно сопоставить только один объект другой сущности;

- Связь один-ко-многим. В этом типе связей несколько строк из дочерний таблицы зависят от одной строки в родительской таблице;

- Связь многие-ко-многим. При этом типе связей одна строка из таблицы А может быть связана с множеством строк из таблицы В. В свою очередь одна строка из таблицы В может быть связана с множеством строк из таблицы А.

Издательство выполняет заказы. Следовательно, необходимо выделить следующие сущности: заказ и заказчик. Первая сущность имеет следующие атрибуты:номер, стоимость выполнения, дата приёма, статус выполнения, дата выполнения, вторая сущность – название, номер телефона и электронная почта. У заказчика может быть несколько заказов, поэтому между данными сущностями связь один-ко-многим (рис 1).

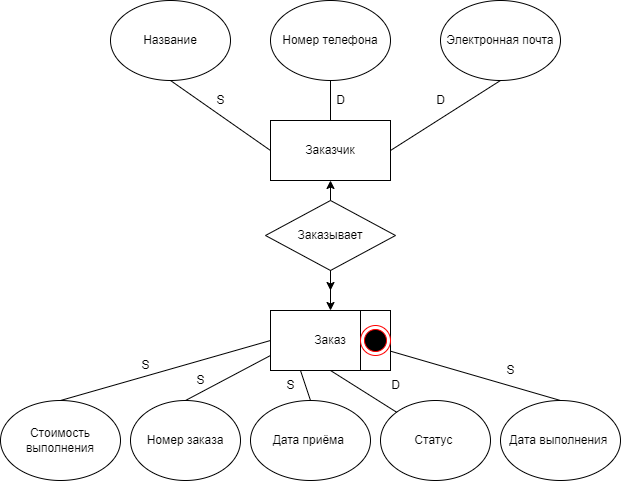


Рисунок 1 – связь между сущностями заказчик и заказ

Над заказом работает один или несколько сотрудников. Следовательно, необходимо выделить сущность сотрудник. У данной сущности имеются следующие атрибуты: ФИО, дата рождения, должность, номер телефона,

электронная почта, фото. Каждый сотрудник может выполнять несколько заказов и над заказом могут работать несколько сотрудников. Следовательно, между этими сущностями связь многие-ко-многим (рис 2).

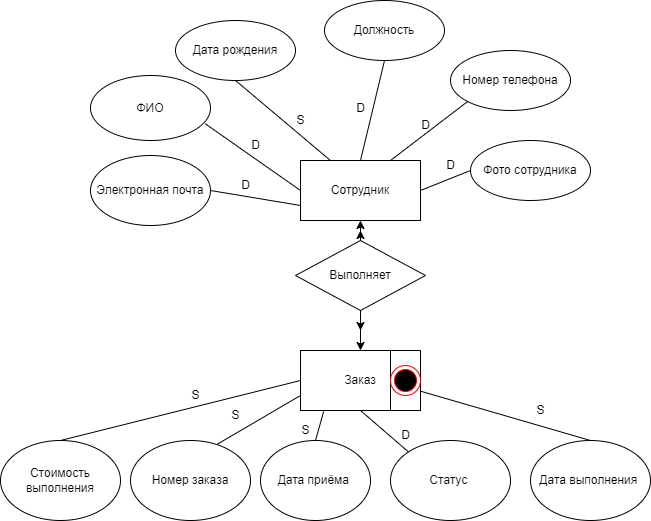


Рисунок 2 – связь между сущностями заказ - сотрудник

Над печатью заказа занимается типография. Следовательно, необходимо выделить сущность типография со следующими атрибутами: название, адрес, электронная почта, номер телефона. Одна типография может выполнять несколько заказов, следовательно между данными сущностями связь один-ко-многим (рис 3).

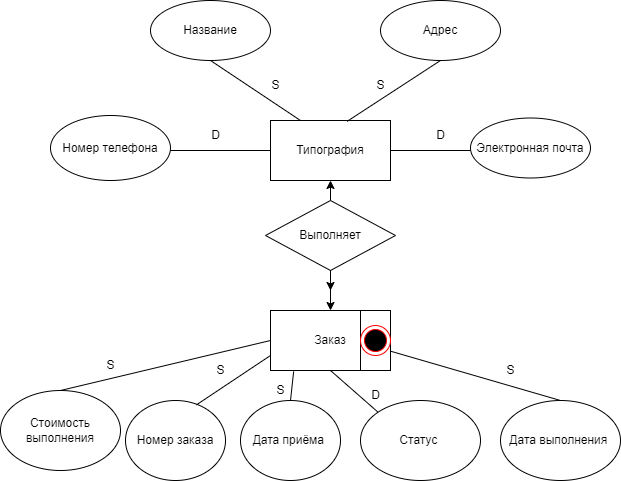


Рисунок 3 – связь между сущностями заказ - типография

При формировании заказа заказчик указывает печатную продукцию, которую он хочет получить. Следовательно, необходимо выделить сущность печатная продукция со следующими атрибутами: название, номер тиража, тираж, стоимость, макет. Несколько печатных продукций могут быть указаны в заказе, поэтому между этими сущностями связь один-ко-многим (рис 4).

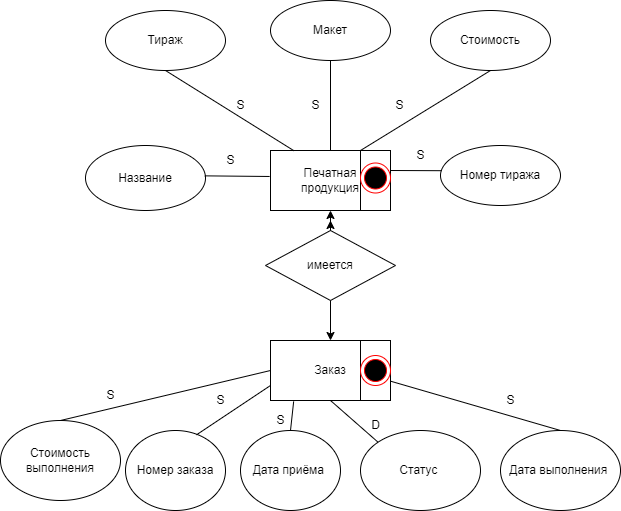


Рисунок 4 – связь между сущностями заказ - печатная продукция

Каждая печатная продукция имеет тип: газета, журнал, книга и т.д. Соответственно, необходимо выделить сущность тип печатной продукции со следующими атрибутами: название, наценка. Тип печатной продукции относиться ко многим печатным продукциям, значит между этими сущностями связь один-ко-многим (рис 5).

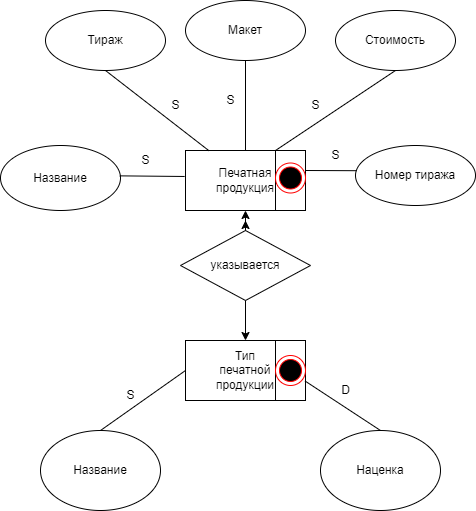


Рисунок 5 – связь между сущностями тип печатной продукции – печатная продукция

Для печати печатных продукций используются материалы. Следовательно, необходимо выделить сущность материал со следующими атрибутами: тип, цвет, размер, стоимость. Связь между этими сущностями многие-ко-многим (рис 6).

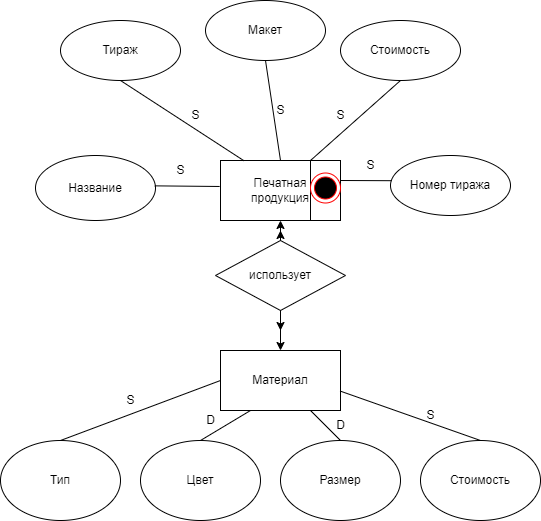


Рисунок 6 – связь между сущностями материал – печатная продукция

2.2 Нормализация таблиц

Избыточность данных приводит к непродуктивному расходованию свободного места на диске и затрудняет обслуживание баз данных. Например, если данные, хранящиеся в нескольких местах, потребуется изменить, в них придется внести одни и те же изменения во всех этих местах. Для избежания данной проблемы необходимо выполнить нормализацию таблиц базы данных.

Нормализация — это процесс организации данных в базе данных, включающий создание таблиц и установление отношений между ними в соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные зависимости.

Существует несколько правил нормализации баз данных. Каждое правило называется «нормальной формой». Если выполняется первое правило, говорят, что база данных представлена в «первой нормальной форме». Если выполняются три первых правила, считается, что база данных представлена в «третьей нормальной форме». Есть и другие уровни нормализации, однако для большинства приложений достаточно нормализовать базы данных до третьей нормальной формы.

В ходе нормализации таблиц были получены следующие таблицы: «Заказ» (номер заказа, дата приёма, статус, дата выполнения, стоимость), «Типография» (название, номер телефона, электронная почта, тип субъекта, название субъекта, город, тип улицы, название улицы, номер дома), «Заказ – Сотрудник», которая связывает таблицу «Заказ» с таблицей «Сотрудник» (фамилия, имя, отчество, дата рождения, должность, номер телефона, электронная почта, фото), «Заказчик» (наименование, номер телефона, электронная почта), «Печатная продукция» (название, номер тиража, тираж, стоимость, макет), «Печатная продукция - Материал», которая хранит количество используемых материалов и связывает таблицу «Печатная продукция» с таблицей «Материал» (тип, цвет, название, стоимость), «Тип печатной продукции» (название, наценка).

2.3 Логическая модель данных

**На основе нормализованных таблиц можно начать создание логической модели данных. Логическая модель данных** — это модель данных конкретной проблемной области, выраженная независимо от конкретного продукта управления базами данных или технологии хранения. Логические модели данных чаще всего используются в бизнес-процессах, которые стремятся охватить важные для организации вещи и то, как они связаны друг с другом. После проверки и утверждения логическая модель данных может стать основой физической модели данных и сформировать дизайн базы данных.

Логические модели данных должны основываться на структурах, определенных в предыдущей концептуальной модели данных, поскольку это описывает семантику информационного контекста, которую также должна отражать логическая модель. Тем не менее, поскольку логическая модель данных предполагает реализацию в конкретной вычислительной системе, содержимое логической модели данных корректируется для достижения определенной эффективности.

Логическая модель данных представлена на рис 7.

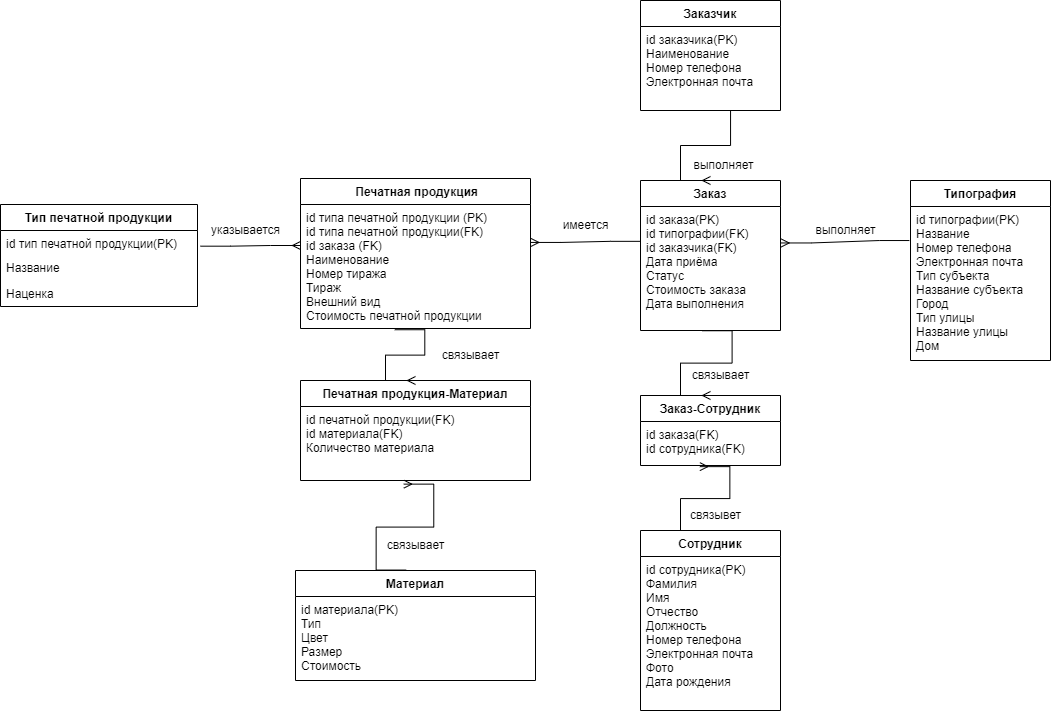


Рисунок 7 – логическая модель данных

Аббревиатура PK возле идентификатора означает, что данное поле таблицы является первичным ключом. Первичный ключ – это поле, которое используется для обеспечения уникальности данных в таблице. Это означает, что значение (информация) в поле первичного ключа в каждой строке (записи) таблицы должно быть уникальным. Уникальность необходима во избежание неоднозначности, когда неизвестно к какой записи таблицы можно обратиться, если в таблице есть повторяющиеся записи (две записи имеют одинаковые значения во всех полях таблицы).

Аббревиатура FK возле идентификатора означает, что данное поле таблицы является внешним ключом. Внешний ключ - это столбец или сочетание столбцов, которое применяется для принудительного установления связи между данными в двух таблицах с целью контроля данных, которые могут храниться в таблице внешнего ключа. Таблица, содержащая внешний ключ, называется дочерней таблицей, а таблица, содержащая ключ-кандидат, называется ссылочной или родительской таблицей.

## 2.4 Физическая модель данных

На основе логической модели данных строится физическая модель данных. Физическая модель данных описывает данные средствами конкретной СУБД. Ограничения, имеющиеся в логической модели данных, реализуются различными средствами СУБД, например, при помощи индексов, декларативных ограничений целостности, триггеров, хранимых процедур. При этом опять-таки решения, принятые на уровне логического моделирования определяют некоторые границы, в пределах которых можно развивать физическую модель данных.

Для информационной системы издательства была выбрана СУБД Microsoft SQL Server. Данная СУБД представляет возможность хранить различные типы данных. В нашем случае будут использоваться следующие типы: тип данных int для хранения id и целочисленных данных, тип данных float для хранения нецелочисленных данных, тип данных datetime2 для хранения дат, тип данных nvarchar для хранения строковых данных, тип данных image для хранений изображений.

Физическая модель данных представлена в приложении 1.

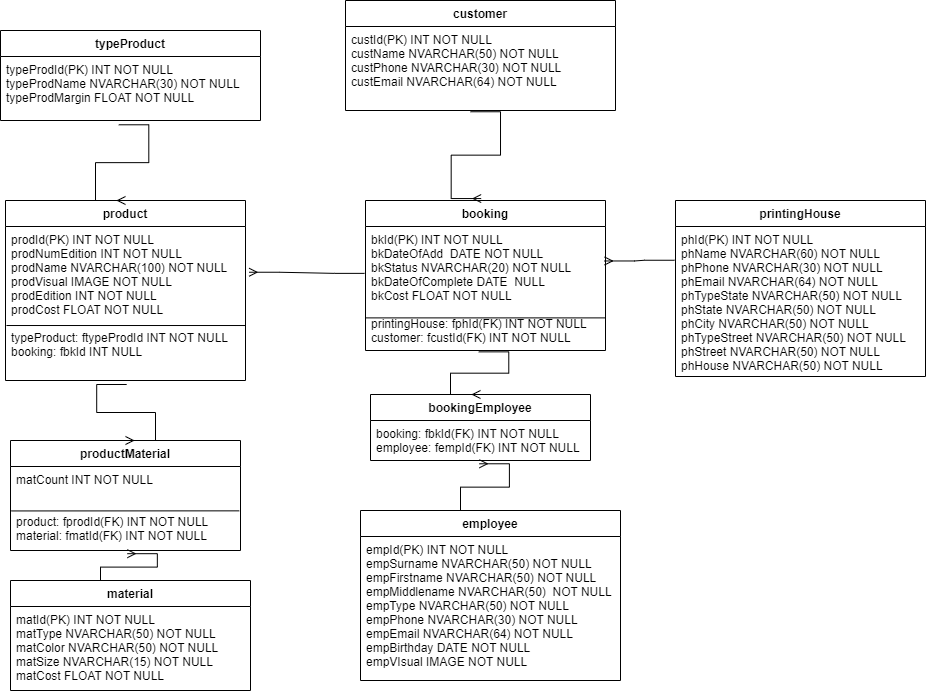


Рисунок 8 – физическая модель данных

Команда NOT NULL заставляет столбец не принимать нулевые значения, это гарантирует, что пользователь не сможет вставить или обновить запись без добавления значения в это поле.

3 Разработка и реализация ИС

3.1 Создание базы данных и хранимых процедур

База данных создана с помощью СУБД MS SQL Server. Таблицы в базе данных соответствуют таблицам из физической модели данных. Разница лишь в том, что у следующих id стоит конструкция ON DELETE CASCADE, которая указывает, что дочерние данные удаляются при удалении родительских данных: fprodId (таблица productMaterial), fbkId (таблица product), fbkId (таблица bookingEmployee).

Хранимые процедуры - объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере. Хранимые процедуры очень похожи на обыкновенные процедуры языков высокого уровня, у них могут быть входные и выходные параметры и локальные переменные, в них могут производиться числовые вычисления и операции над символьными данными, результаты которых могут присваиваться переменным и параметрам. В ходе разработки ИС были созданы следующие хранимые процедуры: addPrintingHouse (добавление данных о типографии), changeCustomer (изменение данных о заказчике), deleteBooking (удаление данных о заказе), GetIdTypeProduct (получение id типа печатной продукции), selectMaterial (получение данных о материалах).

3.2 Классы для работы с данными из базы данных

3.2.1 Класс Booking

Класс Booking – класс для работы с данными о заказе.

Описание класса:

int idPrintingHouse – поле для хранения id печатной продукции.

int idCustomer – поле для хранения id заказчика.

int numBooking – поле для хранения номера заказа

double cost – поле для хранения стоимости заказа

string status – поле для хранения статуса заказа

string namePrintingHouse – поле для хранения названия название типографии.

DateTime startBooking – поле для хранения даты приёма заказа

DateTime endBooking – поле для хранения даты выполнения заказа

int[] idProducts – поле для хранения id печатных продукций

int[] idEmployees – поле для хранения id сотрудников

Для каждого поля есть свойства, которые возвращают данные соответствующих полей.

В данном классе имеются три конструктора для записи данных в поля класса.

public static double GetCostBooking(DataGridView dataGridView) – метод получения стоимости выполнения заказа. Метод принимает таблицу с данными о печатных продукциях, возвращает стоимость выполнения заказа.

private static int GetIdBooking() – метод получения id заказа. Возвращает id заказа.

public int AddBooking() – метод добавления данных о заказе в базу данных. Возвращает 1, если количество добавленных записей равно ожидаемому количеству, в противном случае 0.

private static bool AddBookingEmployee(int[] idEmployees, int idBooking) - метод добавления данных в таблицу «Заказ-Сотрудник». Принимает массив id сотрудников, id заказа. Возвращает true, если данные добавлены в таблицу, в противном случае false.

private static bool DeleteBookingEmployees(int idBooking) – метод удаления данных из таблицы «Заказ-Сотрудник». Принимает id заказа, возвращает true, если данные успешно удалены, в противном случае false.

public static void LoadBookings(DataGridView dataGridView) – метод загрузки данных о заказе в таблицу. Принимает таблицу, в которую будут записываться данные.

public static bool BookingIsBeingExecuted(int idBooking) – метод проверки статуса заказа. Принимает id заказа, возвращает true, если заказ выполняется.

public int ChangeBooking(int idBooking) – метод изменения данных о заказе. Принимает id заказа, возвращает 1, если изменение прошло успешно.

public static int GetIdPrintingHouse(int idBooking) – метод получения id типографии. Принимает id заказа, возвращает id типографии.

public static int[] GetArrayIdEmployees(int idBooking) – метод получения id сотрудников. Принимает id заказа, возвращает массив id сотрудников.

public static int[] GetArrayIdProducts(int idBooking) – метод получения массива id печатных продукций. Принимает id заказа, возвращает массив id печатных продукций.

public static int BookingIsCompleted(int idBooking) – метод выполнения заказа. Принимает id заказа, возвращает количество выполненных заказов.

public static int[] GetArrayIdBookings(DataGridView dataGridView, List<int> selectedRows) – метод получения id выбранных записей о заказах. Принимает таблицу с данными о заказах, список индексов выбранных строк, возвращает массив id выбранных записей о заказах.

public static int DeleteBooking(int[] arrayIdBookings) – метод удаления данных о заказах. Принимает массив id заказов, возвращает количество удалённых строк.

3.2.2 СonnectionToDb

ConnectionToDb – статический класс для подключения/отключения соединения с базой данных.

private SqlConnection connection – поле, хранящее строку подключения к базе данных.

У данного поля есть одноименное свойство, созвращающее данные поля.

public static void Open() – метод подключения соединения с базой данных.

public static void Close() – метод отключения соединения с базой данных.

3.2.3 Customer

Customer – класс для работы с данными о заказчике.

string name – наименование заказчика.

string email – электронная почта заказчика.

string phone – номер телефона заказчика.

Данный класс имеет один конструктор и переопределенный метод ToString(), возвращающий данные о заказчике.

public static Customer GetCustomer(int idCustomer) – метод получения даннных о заказчике. Принимает id заказчика, возвращает данные о заказе.

public int AddCustomer() – метод добавления данных о заказчике в базу данных. Возвращает количество добавленных заказчиков

public static void LoadCustomers(DataGridView dataGridView) – метод добавления данных о заказчиках в таблицу. Принимает таблицу, в которую будут добавляться данные.

public static List<int> GetNumbersOfOrdersThisCustomer(string email) – метод получения списка номер заказов заказчика. Принимает электронную почту заказчика, возвращает список номер заказов заказчика.

public static int GetIdCustomerByPhone(string phone) – метод получения id записи о заказчике. Принимает номер телефона заказчика, возвращает id заказчика.

public static int GetIdCustomerByEmail(string email) – метод получения id записи о заказчике. Принимает электронную почту заказчика, возвращает id заказчика.

public static int GetCountRecords() – метод получения количества записей в таблице. Возвращает количество записей.

public static DataTable GetTableByOccurence(string order, int outStringCount) – метод получения данных о заказчиках по определенному порядку фильтрации. Принимает порядок фильтрации, количество выводимых строк, возвращает DataTable с отсортированными данными.

public static bool ExistEmailInDb (char typeWork, string pastEmail, string newEmail) – метод, проверяющий существование электронной почты заказчика в базе данных, принимает тип работы с данными, прошлую и новую электронную почту.

public static bool ExistPhoneInDb (char typeWork, string pastEmail, string newEmail) – метод, проверяющий существование номер телефона заказчика в базе данных, принимает тип работы с данными, прошлый и новый номера телефона.

public static bool CustomerHasBooking(int idCustomer) – метод, определяющий есть ли у заказчика заказ. Принимает id заказчика, возвращает true, если есть заказ.

public int ChangeCustomer(int id) – метод изменения данных о заказчике. Принимает id заказчика, возвращает количество измененных данных.

public int DeleteCustomers(int[] arrayId) – метод удаления данных о заказчиках. Принимает массив id заказчиков, возвращает количество удаленных записей.

3.2.4 Employee

Employee – класс для работы с данными о сотруднике.

string name – имя сотрудника

string surname – фамилия сотрудника

string middlename – отчество сотрудника

string type – должность сотрудника

string email – электронная почта сотрудника

string phone – номер телефона сотрудника

DateTime birthday – дата рождения сотрудника

Image photoAsImage – фотография сотрудника

У данных полей есть свойства, которые возвращают данные соответствующих полей.

У данного класс есть 3 конструктора, в которых происходит заполнение переменных класса, а также переопределенных метод ToString(), выводящий данные о сотруднике.

public static Employee GetEmployee(int idEmployee) – метод получения данных о сотруднике. Принимает id сотрудника, возвращает данные о заказчике.

public int AddEmployee() – метод добавления данных о сотруднике в базу данных. Возвращает количество добавленных сотрудников.

public static void LoadEmployees(DataGridView dataGridView) – метод добавления данных о сотрудниках в таблицу. Принимает таблицу, в которую будут добавляться данные.

private static byte[] GetPhotoEmployeeByPhone(string phone) – метод получения фотографии сотрудника. Принимает номер телефона сотрудника, возвращает фотографию сотрудника в виде массива байт.

public static int GetIdEmployeeByPhone(string phone) – метод получения id записи о сотруднике. Принимает номер телефона, возвращает id записи о сотруднике.

public int ChangeEmployee(int id) – метод изменения данных о сотруднике. Принимает id сотрудника, возвращает количество измененных данных.

public static bool ExistEmailInDb (char typeWork, string pastEmail, string newEmail) – метод, проверяющий существование электронной почты сотрудника в базе данных, принимает тип работы с данными, прошлую и новую электронную почту.

public static bool ExistPhoneInDb (char typeWork, string pastEmail, string newEmail) – метод, проверяющий существование номер телефона сотрудника в базе данных, принимает тип работы с данными, прошлый и новый номера телефона.

public int DeleteEmployees(int[] arrayId) – метод удаления данных о сотрудниках. Принимает массив id сотрудников, возвращает количество удаленных записей.

public static int GetIdEmployeeByEmail(string email) – метод получения id сотрудника. Принимает электронную почту, возвращает id сотрудника.

3.2.5 Material

Material – класс для работы с данными о материалах.

string type – тип материала.

string color – цвет материала.

string size – размер материала.

double cost – стоимость материала.

int id – id материала.

int count – количество материалов.

Данный класс имеет два свойства для полей id и count, которые возвращают данные этих полей.

Данный класс имеет 3 конструктора для заполнения полей класса, а также переопределенный метод ToString(), который выводит информацию о материале.

public static DataTable LoadMaterial() – метод получения данных о материалах из базы данных. Возвращает DataTable с данными о материалах.

public bool ExistMaterial(DataGridView dataGridView) – метод проверки существования записи о материале. Принимает таблицу, у которой ищутся данные о материале.

public int GetIdMaterial() – метод получения id материала. Возвращает id материала.

public int AddMaterial() – метод добавления данных о материале в базу данных. Возвращает количество добавленных записей.

public static int DeleteMaterial(Material[] materials) – метод удаления данных о материалах. Принимает массив материалов, возвращает количество удаленных записей.

public int ChangeMaterial(int id) – метод изменения данных материала. Принимает id материала, возвращает количество изменённых записей.

public static DataTable GetTableOccurrence(string columnName, string order, int countString) – метод, возвращающий таблицу с определенным количеством строк по определенному порядку вывода. Метод принимает название столбца, порядок сортировки, количество выводимых строк, возвращает DataTable с отсортированными данными.

3.2.6 PrintingHouse

PrintingHouse – класс для работы с данными о типографии.

string name – название типографии.

string numberPhone – номер телефона типографии.

string email – электронная почта типографии.

string typeState – тип субъекта.

string nameState – название субъекта.

string city – город.

string typeStreet – тип улицы.

string nameStreet – название улицы.

string numberHouse – дом.

Для каждого поля есть свойство, возвражающее данные о поле.

Класс имеет один конструктор для заполнения полей классов и переопределённый метод ToString(), возвращающий данные о типографии.

public int AddPrintingHouse() – метод добавления данных о типографии в базу данных. Возвращает количество добавленных записей.

public static void LoadPrintingHouse(DataGridView dataGridView) – метод добавления данных о типографиях в таблицу. Принимает таблицу, в которую будут добавляться данные.

public static List<int> GetNumbersOfOrdersThisPrintingHouse(string email) – метод получения списка номер заказов типографии. Принимает электронную почту типографии, возвращает список номер заказов типографии.

public static int GetIdPrintingHouseByEmail(string email) – метод получения id записи о типографии по электронной почте. Принимает электронную почту, возвращает id записи.

public int ChangePrintingHouse(int id) – метод изменения данных о типографии. Принимает id типографии, возвращает количество измененных данных.

public int DeletePrintingHouses(int[] arrayId) – метод удаления данных о типографиях. Принимает массив id типографий, возвращает количество удаленных записей.

3.2.7 Product

Product – класс для работы с данными о печатной продукции.

string name – название печатной продукции.

string nameTypeProduct – название типа печатной продукции.

int numberEdition – номер тиража.

int prodEdition – тираж.

int idTypeProduct – id типа печатной продукции.

double cost – стоимость печатной продукции.

Image designAsImage – макет печатной продукции.

List<Material> Materials – список материалов.

Данный для каждого поля свойство, возвращающее данные соответствующего поля, 5 конструкторов и переопределенный метод ToString().

public static int GetIdProduct(string name, int numEdition) – метод получения id печатной продукции. Принимает название и номер тиража, возвращает id печатной продукции.

public static double GetCostProduct(DataGridView dataGridView, int margin, int countProduct) – метод получения стоимости печатной продукции. Принимает таблицу с данными о печатных продукциях, наценку, количество печатной продукции, возвращает стоимость печатной продукции.

public int AddProduct() – метод добавления данных о печатной продукции в базу данных. Возвращает 1, если добавление прошло успешно.

private static bool AddProductTypeProduct(List<Material> materials, int idProduct) – метод добавления данных в таблицу «Печатная продукция-материал». Принимает список материалов, id печатной продукции, возвращает true, если добавление прошло успешно.

public static void LoadProductsInTable(DataGridView dataGridView) – метод добавления данных о печатных продукциях в таблицу. Принимает таблицу, в которую будут добавляться данные.

private static byte[] GetPhotoProduct(string name, int numEdition) – метод получения макета печатной продукции из базы данных. Принимает название печатной продукции, номер тиража, возвращает макет печатной продукции.

public int ChangeProduct(int idProduct) – метод изменения данных о печатной продукции. Принимает id печатной продукции, возвращает количество измененных данных.

public static int DeleteProducts(int[] arrayId) – метод удаления данных о печатных продукциях. Принимает массив id печатных продукций, возвращает количество удаленных записей.

3.2.8 TypeProduct

TypeProduct – класс для работы с типами печатной продукции.

string name – название

double margin - наценка

Данный класс имеет свойства для этих полей, один конструктор и переопределённый метод toString(), возвращающий данные о типе печатной продукции.

public static int GetIdTypeProduct(string name, double margin) – метод получения id типа печатной продукции. Принимает название и наценку, возвращает id.

public int AddTypeProduct() – метод добавления данных о типе печатной продукции. Возвращает количество добавленных записей.

public static void LoadTypesProduct(DataGridView dataGridView) – метод добавления данных о типах печатной продукции в таблицу. Принимает таблицу, в которую будут добавляться данные.

public int ChangeTypeProduct(int idTypeProduct) – метод изменения данных о типе печатной продукции. Принимает id типа печатной продукции, возвращает количество измененных данных.

public static int DeleteTypesProduct(int[] arrayId) – метод удаления данных о типах печатной продукции. Принимает массив id типов печатной продукции, возвращает количество удаленных записей.

public static List<Product> GetListProductsThisTypeProduct(string name, double margin) – метод получения печатных продукций выбранного типа печатной продукции. Принимает название типа печатной продукции, наценку, возвращает список печатных продукций.

public static void SelectRowTypeProduct(DataGridView dataGridView, int idTypeProduct) – метод поиска строки о типе печатной продукции в таблице. Принимает таблицу с данными о типе печатной продукции, id типа печатной продукции.

public static int GetCountRecords() – метод получения количества записей. Возвращает количество записей о типах печатных продукций.

public static bool TypeProductIsIndicated(int idTypeProduct) – метод, определяющий указан ли тип печатной продукции в печатной продукции. Принимает id типа печатной продукции, возвращает true, если тип печатной продукции указан в печатной продукции.

public static void SelectRowTypeProduct(DataGridView dataGridView, int idTypeProduct) – метод поиска строки о типе печатной продукции в таблице. Принимает таблицу с данными о типах печатной продукции и id типа печатной продукции

Весь код проекта с комментариями можно просмотреть, открыв репозиторий проекта. Ссылка на репозиторий содержится в Приложение 2.

3.3 Руководство пользователя

После запуска приложения откроется главное меню ИС издательства. В этом меню находится таблица с данными о заказах, блок для работы с данными, а также вкладки для перехода к соответствующим данным.

При нажатии на вкладку поиск отобразятся компоненты для поиска данных о заказах.

Нажав вкладку обработка, а следом на кнопку для оформления заказа, откроется меню для ввода данных о заказе.

Выбрав все необходимые данные, пользователь после нажатия на кнопку сохранения введенных данных попадает на главную форму, где он может добавить заказ или его отменить.

Чтобы изменить данные о заказе, пользователю необходимо будет выбрать одну запись. Если заказ не выполнен, то пользователю откроется форма для ввода данных о заказе, которая будет подсвечивать текущие данные о заказе.

После ввода измененных данных пользователь возвращается на главное меню, где он может изменить запись или отменить изменение.

Если заказ выполнен, то пользователь может его удалить, поставив галочку в таблице и нажав на соответствующую кнопку.

Кнопки «Выбрать всё» и «Отменить выбор строк» отвечают за выбор всех строк в таблицы и за отмену этого действия.

При выборе заказа и нажатии на кнопку «Выполнить заказ», пользователь переводит заказ из состояния «Выполняется» в состояние «Выполнен». Датой выполнения заказа становится день, когда пользователь «выполнил» заказ.

Чтобы получить отчёт о заказе, пользователь должен выбрать выполненный заказ из таблицы и нажать на соответствующую кнопку. После этого откроется меню, содержащее все данные о заказе. При нажатии на кнопку «Загрузить в RTF» пользователь получит файла формата .rtf, который будет храниться в выбранной пользователем директории.

Во вкладке поиска пользователь может найти следующие данные: данные, имеющие строковый тип данных (для этого необходимо выбрать столбец таблицы для поиска и ввести данные для поиска в текстовое поле), данные, имеющие числовой тип данных (для этого необходимо выбрать столбец таблицы, ввести диапазон значений в текстовые поля и нажать на кнопку «Поиск числовых данных»), данные о датах (для этого необходимо выбрать столбец для поиска, выбрать диапазон дат и нажать на кнопку «Поиск по дате»). Если пользователь захочет отменить выбор строк, то необходимо нажать на кнопку «Сброс поиска».

Нажав на вкладку «Сотрудники», пользователь попадает в соответствующее меню.

Логика обработки такая же, как и у меню с заказами, только отсутствует возможность получения отчёта и формирования заказа. Чтобы добавить заказ, необходимо нажать на кнопку «Ввести данные», где откроется меню для ввода данных. Это же меню откроется при изменении данных. Разница лишь в том, что при изменении данных текущие данные будут отображены в соответствующих компонентах.

В меню «Сотрудники» кнопки для выбора строк и отмены выбора строк были перенесены во вкладку «Выбор».

Во вкладке «Поиск» пользователь может найти строковые данные, данные о дате рождения (логика поиска данных о сотруднике идентична логике поиска данных о заказе), найти заказы, которые выполняет сотрудник. Для этого нужно выбрать одного сотрудника из таблицы и нажать на кнопку «Выбрать заказ». После этого в пустой блок будут внесены данные о заказах сотрудника. При нажатии на кнопку «Мода сотрудников» откроется меню, где пользователь сможет увидеть данные о сотрудниках, количестве выполняемых заказов этим сотрудником.

Пользователь может сам выбрать количество выводимых данных и порядок вывода, указав это в соответствующих компонентах.

Вернувшись в главное меню, пользователь может перейти в меню «Материалы», нажав на соответствующую вкладку.

Ввод данных и их изменение происходит на этой же форме. Логика изменения данных: пользователь выбирает материал из таблицы, нажимает на кнопку «Выбрать для изменения», если материал не используется, то в соответствующие компоненты будут занесены текущие данные о материале. Пользователь может отменить изменение данных, нажав на соответствующую кнопку или изменить данные.

Пользователь может найти данные, выбрав столбец для поиска и вводя данные в соответствующее текстовое поле или указав диапазон стоимости материала и нажав на кнопку «Поиск по стоимости».

Нажав на кнопку «Мода данных» откроется окно с модой данных о каждом столбце таблицы. Логика отображения данных такая же, что и у моды сотрудников.

Вернувшись в главное меню, пользователь может просмотреть данные о типографиях. Для этого необходимо нажать на одноименную вкладку, после чего откроется меню с данными о типографиях.

Функционал в данном меню совпадает с функционалом меню сотрудников, за исключением того, что данные о типографиях можно найти только при выборе соответствующего столбца из выпадающего списка и вводе данных для поиска.

Для открытия меню «Заказчики» необходимо в главном меню нажать на одноименную вкладку.

Функционал данного меню совпадает с функционалом меню с типографиями.

Данная форма имеет схожий функционал, что и у формы с сотрудниками. Разница лишь в том, что кнопки для выбора строк и отмены строк находятся во вкладке «Обработка». Вкладка «Поиск» имеет компоненты для поиска строковых данных и числовых данных.

Вместо моды данных меню «Печатная продукция» имеет кнопку вывода всех данных о печатной продукции. Логика получения данных и сохранение данных в файл идентична логике получения всех данных о заказе.

В меню «Печатная продукция» можно перейти в меню «Типы печатной продукции», нажав на соответствующую кнопку

Форма для ввода данных о типе печатной продукции представлена в

Форма «Типы печатной продукции» имеет схожий функционал, что и у формы «Печатная продукция». Разница лишь в том, что вместо вывода всех данных в текущей форме имеется возможность просмотреть моду данных.

4 Тестирование ИС

Последним этапом создания приложения является его тестирование и отладка. В ходе тестирования программы выявляются все её недостатки и особенности работы. Ниже будет представлена таблица, описывающая все действия, которые будет выполнять специалист по приёму и обработке заказов в информационной системе издательства.

Таблица 1 – тестирование работы ИС с данными о заказах

|  |  |
| --- | --- |
| Ожидаемый результат | Полученный результат |
| При запуске exe файла откроется главное меню ИС, в которой будут отображены данные о заказах. | При запуске exe файла открывается главное меню ИС, в которой отображены данные о заказах. |
| При нажатии на кнопку «Сформировать заказ» должна открыться форма для ввода данных о заказе.  В ней должны отобразиться все сотрудники, заказчики, типографии и печатные продукции, которые не принадлежат никакому заказу. | При нажатии на кнопку «сформировать заказ» открывается форма для ввода данных о заказе.  В ней отображаются все сотрудники, заказчики, типографии и печатные продукции, которые не принадлежат никакому заказу. |
| При корректном вводе всех данных и нажатии на кнопку «Сохранить введённые данные» должен произойти переход в основную форму, где над кнопкой "Добавить заказ" высветиться сообщение «Вы можете добавить запись».  В противном случае, должно высветиться окно с сообщением о том, что не все данные были введены. Переход между формами отсутствует | При корректном вводе всех данных и нажатии на кнопку «Сохранить введённые данные» происходит переход в основную форму, где над кнопкой "Добавить заказ" высвечивается сообщение «Вы можете добавить запись».  В противном случае, высвечивается окно с сообщением о том, что не все данные были введены. Переход между формами отсутствует. |
| После нажатия на кнопку «Добавить заказ» должно высветиться сообщение о успешном добавлении заказа, сообщение о добавлении должно исчезнуть, данные в таблице должны быть обновлены. | После нажатия кнопку «Добавить заказ» высвечивается сообщение о успешном добавлении заказа, сообщение о добавлении исчезает, данные в таблице обновляются. |
| При нажатии на кнопку добавления заказа должно высветиться сообщение о том, что пользователь должен сперва ввести данные, если пользователь не ввел данные. | При нажатии на кнопку добавления заказа высвечивается сообщение о том, что пользователь должен сперва ввести данные, если пользователь не ввел данные. |
| При нажатии на кнопку «Выбрать для изменения» должно высвечиваться сообщение о том, что необходимо выбрать одну запись, если пользователь не выбрал запись,  в противном случае должно высветиться сообщение о том, что невозможно изменить данные о заказе, если заказ выполнен, в противном случае должна открыться форма для ввода данных, где должны быть подсвечены текущие данные о заказе. | При нажатии на кнопку «Выбрать для изменения» высвечиваться сообщение о том, что необходимо выбрать одну запись, если пользователь не выбрал запись,  в противном случае высвечивается сообщение о том, что невозможно изменить данные о заказе, если заказ выполнен, в противном случае открывается форма для ввода данных, где подсвечены текущие данные о заказе. |
| При сохранении измененных данных  должен происходить переход на главное меню, где должно высветиться сообщение о том, что пользователь может изменить запись | При сохранении измененных данных  происходит переход на главное меню, где высвечивается сообщение о том, что пользователь может изменить запись |
| При нажатии на кнопку изменения заказа должно высветиться соответствующее сообщение, сообщение об изменении должно исчезнуть, данные должны быть обновлены в таблице. | При нажатии на кнопку изменения заказа высвечивается соответствующее сообщение, сообщение об изменении исчезает, данные обновляются в таблице. |
| Если заказ выполнен, то при нажатии на кнопку удаления заказа должно высветиться сообщение о успешном удалении, данные в таблице должны быть обновлены, в противном случае должно высветиться предупреждающее сообщение о том, что заказ выполняется. | Если заказ выполнен, то при нажатии на кнопку удаления заказа высвечивается сообщение о успешном удалении, данные в таблице обновляются, в противном случае высвечивается предупреждающее сообщение о том, что заказ выполняется. |
| При нажатии на кнопку «Выбрать всё» все записи в таблице должны быть выбраны. | При нажатии на кнопку «Выбрать всё» все записи в таблице становятся выбранными. |
| При нажатии на кнопку отмены добавления/изменения, сообщения должны исчезнуть, действие должно быть отменено. | При нажатии на кнопку отмены добавления/изменения, сообщения исчезают, действие отменяется. |
| При нажатии на кнопку «Выполнить заказ» должно высветиться сообщение о том, что после выполнения заказа изменение его  данных станет невозможным. Если пользователь соглашается на выполнение, то данные о статусе и дате выполнения обновляются. | При нажатии на кнопку «Выполнить заказ» высвечивается сообщение о том, что после выполнения заказа изменение его данных станет  невозможным. Если пользователь соглашается на выполнение, то данные о статусе и дате выполнения обновляются. |
| При выборе одного выполненного заказа и при нажатии на кнопку «Сформировать отчёт» должна открыться форма, содержащая все данные о заказе. В противном случае, должно вывестись сообщение о невозможности формирования отчёта. | При выборе одного выполненного заказа и при нажатии на кнопку «Сформировать отчёт» открывается форма, содержащая все данные о заказе. В противном случае, высвечивается сообщение о невозможности формирования отчёта. |
| При нажатии на кнопку «Загрузить в RTF» и выборе пути сохранения файла, должно вывестись сообщение о успешном сохранении данных в файл. | При нажатии на кнопку «Загрузить в RTF» и выборе пути сохранения файла, выводится сообщение о успешном сохранении данных в файл. |
| При поиске данных должны выводится данные, отвечающие запросу. | При поиске данных выводятся данные, отвечающие запросу. |
| При нажатии на кнопку сброса поиска должен происходить сброс поиска. | При нажатии на кнопку сброса поиска происходит сброс поиска. |

Заключение

В курсовой работе была разработана информационная система издательства. Данная ИС отвечает требованиям, которые были выявлены в анализе технического задания. Программа была протестирована и отлажена.

В ходе проектирования информационной системы были разработаны концептуальная, логическая и физическая модели данных.

Также были реализованы хранимые процедуры, которые позволяют удалять, изменять, добавлять и получать необходимые данные.

В итоге был реализован следующий функционал:

- Добавление, удаление, изменение, вывод данных для работы с заказами;

- Проверки на корректность введённых данных для обеспечения логической целостности системы;

- Вывод окон с сообщениями об успешном или неуспешном действии пользователя;

- Поиск данных по одному или нескольким критериям;

- Автоматический подсчёт стоимости выполнения заказа;

- Формирование полного отчёта о заказе или о печатной продукции;

- Сохранение данных о заказе или о печатной продукции в файл.

Данная информационная система готова к использованию издательством.

# Список литературы

1. Бейли Л. Изучаем SQL. — СПб.: Питер, 2012. — 592 с.

2. К. Дж. Кейт. SQL и реляционная теория. Как грамотно писать код на SQL. – Пер. с англ. – СПб.: Символ-Плюс, 2010. – 480 с.

3. Петкович Д. Microsoft SQL Server 2012. Руководство для начинающих: Пер. с англ. — СПб.: БХВ-Петербург, 2013. — 816 с.

# 

# Приложение 1. Модели данных

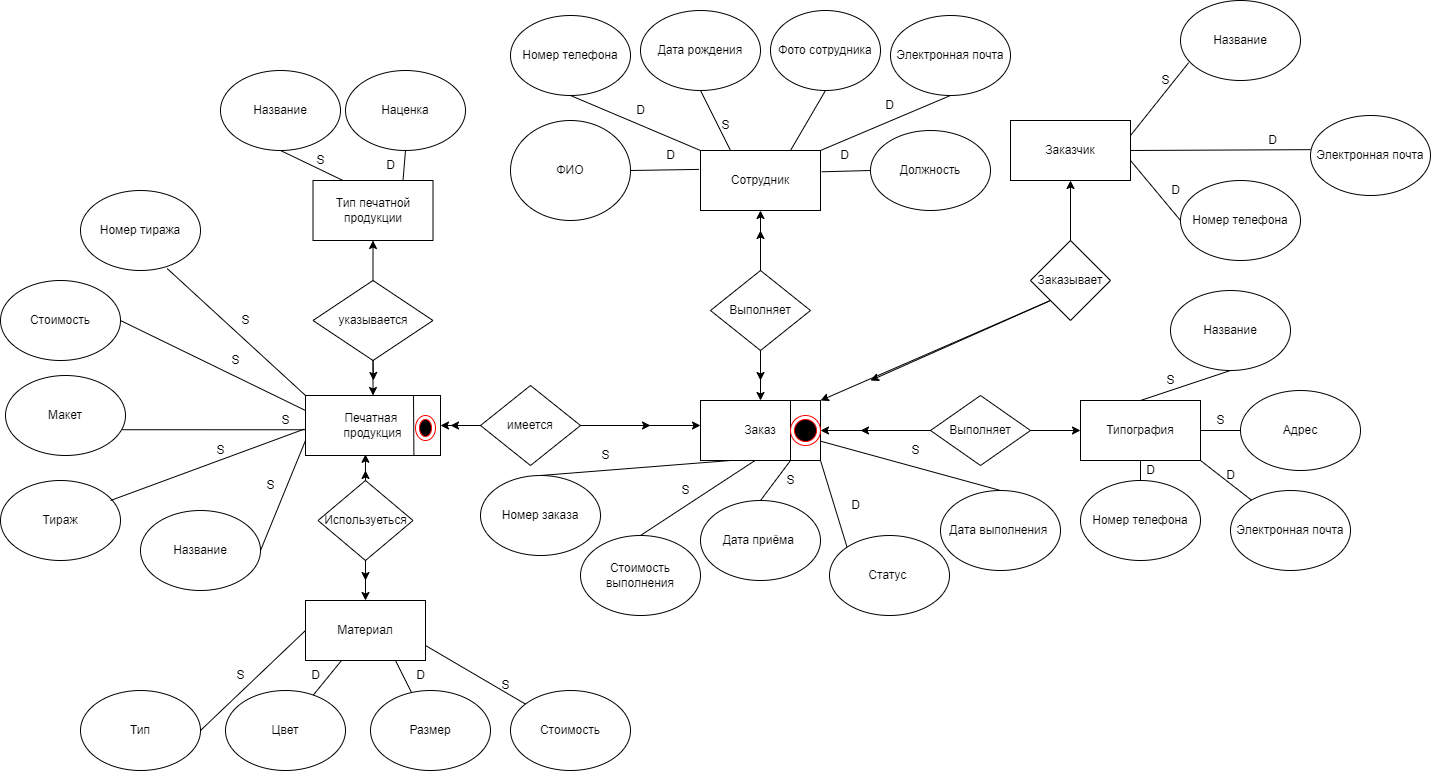


Рисунок 1 – концептуальная модель данных

# Приложение 2.

Ссылка на репозиторий с исходным кодом:

https://github.com/shish1mora/Database-Publishing-House

# Приложение 3. Снимки окон программы

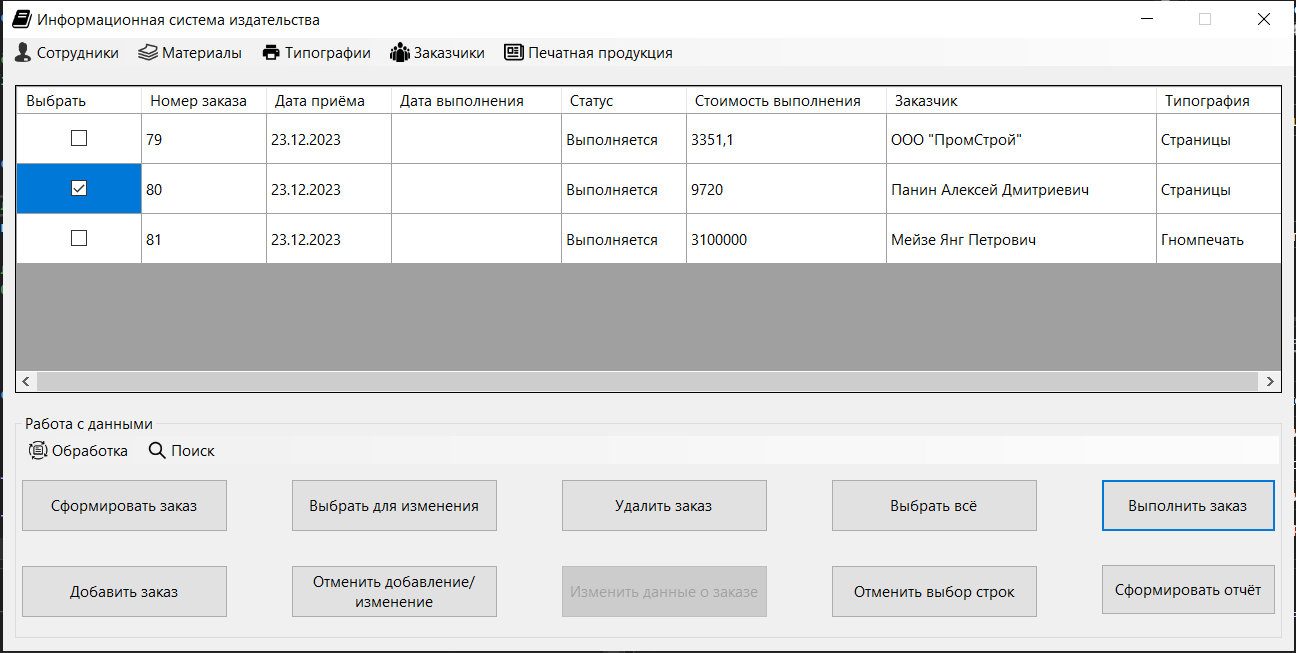


Рисунок 1 – основная форма приложения

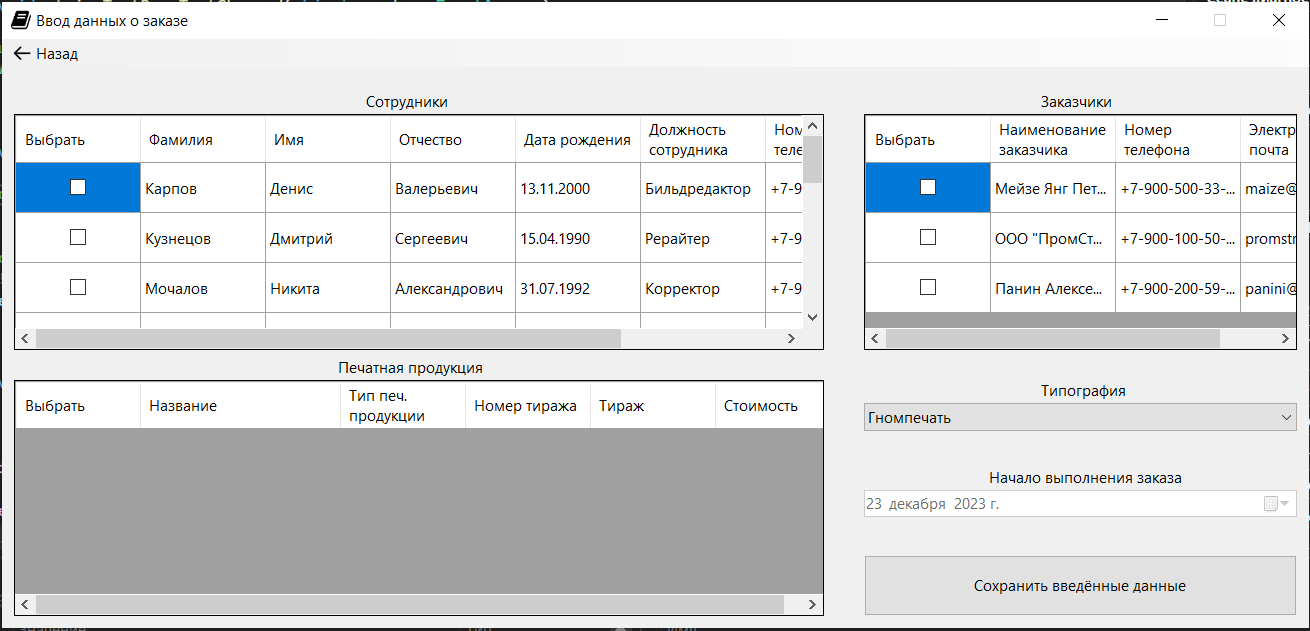


Рисунок 2 – составление заказа

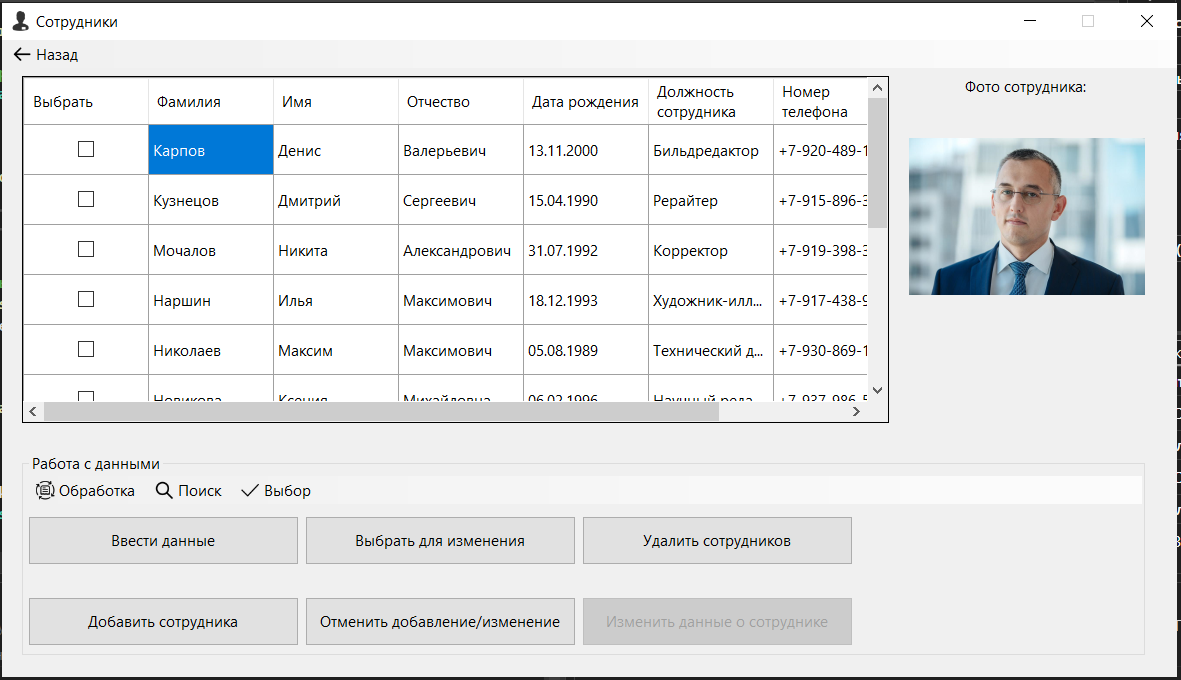


Рисунок 3 – форма с данными о сотрудниках

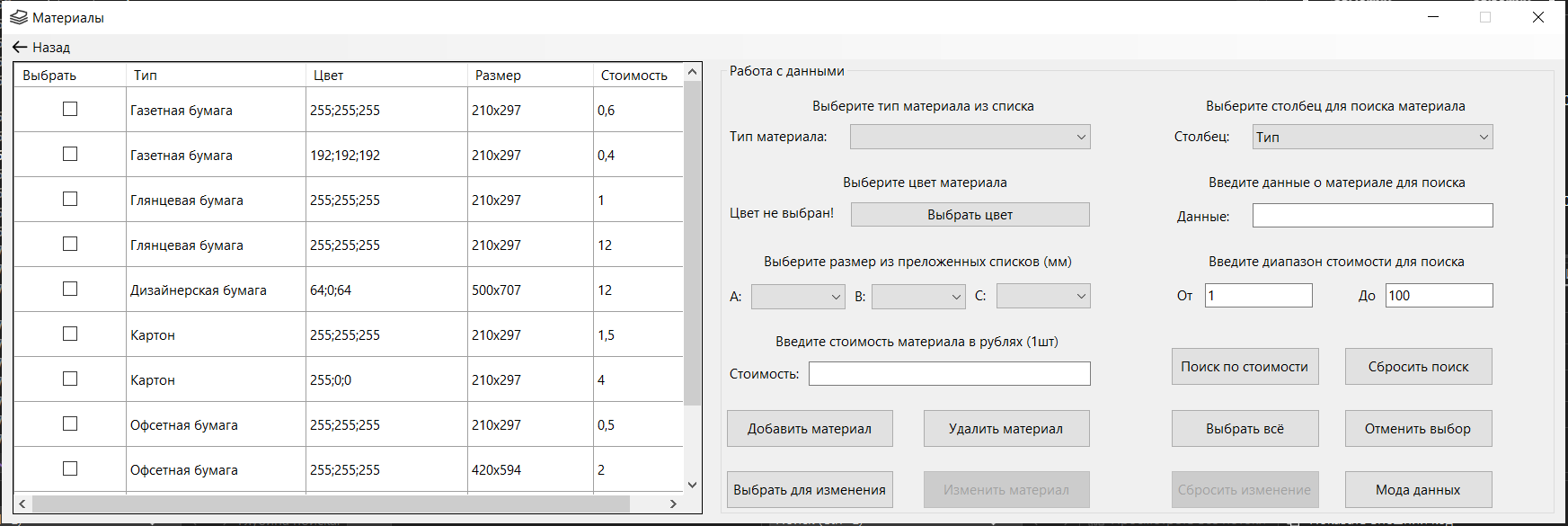


Рисунок 4 – форма с данными о материалах

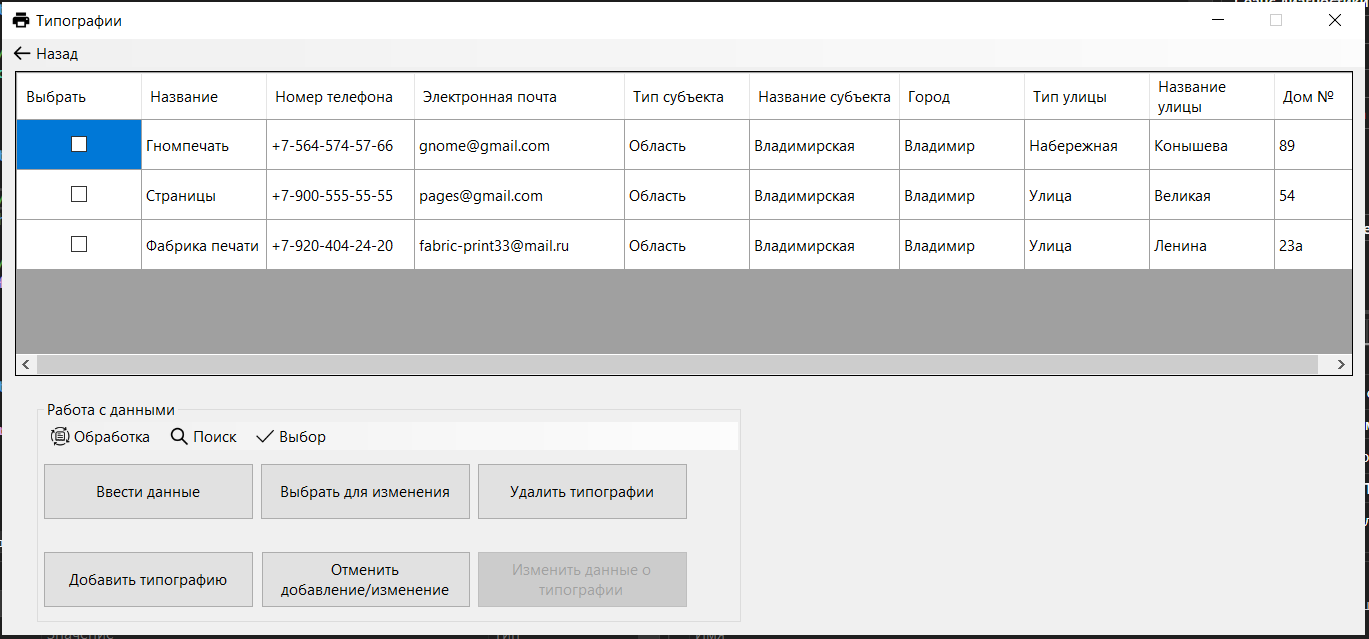


Рисунок 5 – форма с данными о типографиях

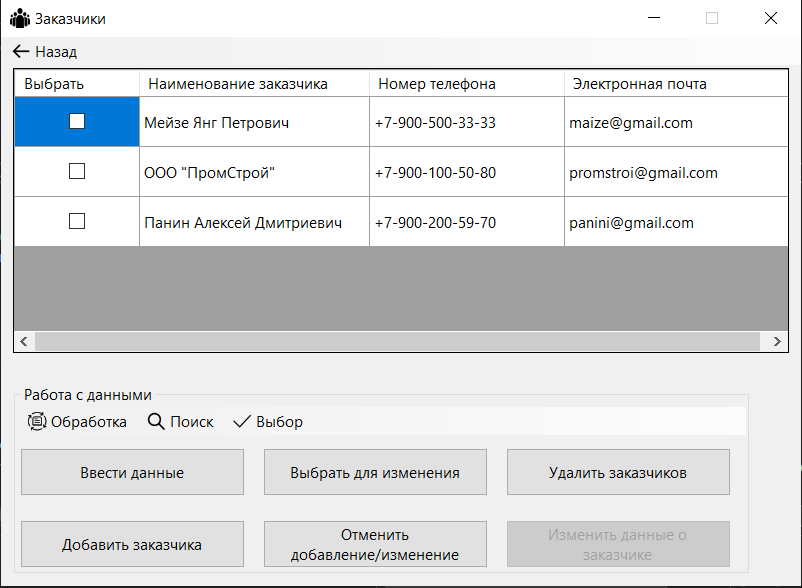


Рисунок 6 – форма с данными о заказчиках

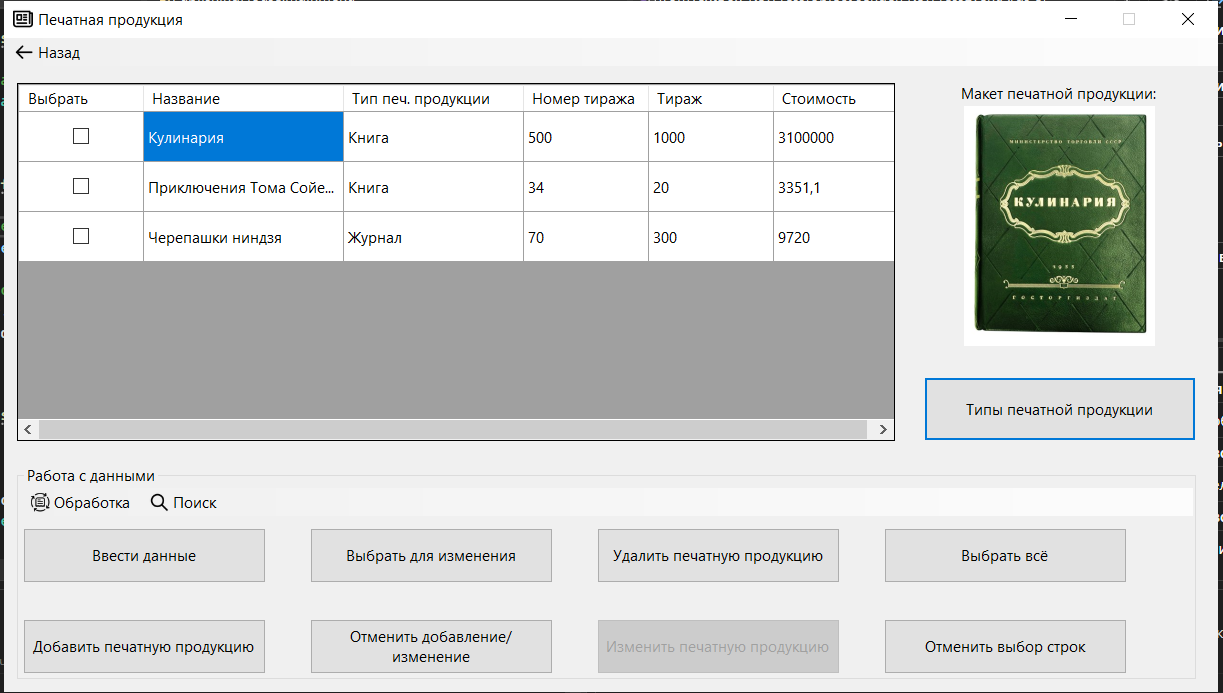


Рисунок 7 – форма с данными о печатной продукции