

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ .....	4
1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНЫЙ МОДЕЛИ .....	5
1.1 Анализ предметной области и выявление необходимого набора сущностей	5
1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов.....	6
1.3 Определение связей между объектами.....	7
1.4 Описание полученной модели на языке инфологического проектирования..	7
2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ.....	9
2.1 Построение набора необходимых отношений базы данных .....	9
2.2 Задание первичных и внешних ключей определенных отношений .....	9
2.3 Третья нормальная форма .....	11
2.4 Определение ограничений целостности для внешних ключей и для отношений в целом .....	12
2.5 Графическое представление связей между внешними ключами .....	12
3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ .....	13
4 ЗАПИСЬ ВЫРАЖЕНИЙ, УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL.....	17
5 ВЫБОР И ОБОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ .....	23
6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ .....	24
6.1 Разработка и построение интерфейса главной и рабочих форм .....	24
6.2 Построение главного меню и кнопок панели инструментов .....	24
6.3 Выполнение программного кода на Microsoft Visual C# .....	25
ЗАКЛЮЧЕНИЕ .....	31
СПИСОК ЛИТЕРАТУРЫ .....	32
ПРИЛОЖЕНИЕ А (обязательное) .....	33
ПРИЛОЖЕНИЕ Б (обязательное) .....	34
ПРИЛОЖЕНИЕ В (обязательное) .....	35

					<i>РДА.508190.ПЗ</i>		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Резниченко О.А.			Информационная система магазина музыкальных инструментов		
Провер.		Скуковская А.А.					
Реценз.							
Н. Контр.							
Утверд.							
					Лит.	Лист	Листов
						3	27
					Учреждение образования «Полоцкий государственный университет имени Евфросинии Полоцкой» гр.20-ИТ-1		

## ВВЕДЕНИЕ

Тема данной курсовой работы – проектирование реляционной базы данных магазина музыкальных инструментов. В контексте музыкальной индустрии, где творческое выражение взаимодействует с технологическим прогрессом, неотъемлемым элементом эффективного управления информацией о продукции и клиентах является систематизированная база данных. Проект предполагает создание структуры, способной надежно хранить и обрабатывать данные о музыкальных инструментах, клиентах и поставщиках, в целях оптимизации бизнес-процессов и обеспечения оперативного принятия решений.

Для обеспечения функциональности базы данных и удобства взаимодействия с информационной системой предполагается разработка десктопного приложения. Это приложение будет предоставлять администраторам возможность управления базой данных. Простота, удобство и функциональность приложения являются приоритетными задачами при его разработке.

Актуальность данного исследования обусловлена активным внедрением цифровых технологий в различные области жизни. В условиях цифровой трансформации учет и обработка данных в электронной форме становятся неотъемлемой частью современного бизнеса. Разработка базы данных и приложения для магазина музыкальных инструментов представляет собой важный шаг в упрощении и оптимизации управления таким видом торговли. В условиях множества аналогичных программных продуктов, данная информационная система будет уникальна своей ориентацией на хранение и обработку данных о музыкальных инструментах, предоставляя более специализированный и точный инструмент для управления бизнесом.

					РДА.508190.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

# 1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНЫЙ МОДЕЛИ

## 1.1 Анализ предметной области и выявление необходимого набора сущностей

В ходе анализа знаний и разработке базы данных были выявлены следующие основные сущности:

Сущность Client описывает пользователей, которые делали покупки в данном магазине. Характеризуется именем, номером телефона, электронной почтой.

Сущность Orders описывает заказы пользователей. Характеризуется идентификационным номером заказанного товара, датой доставки и ценой.

Сущность PaymentMethod описывает вид оплаты. Характеризуется идентификационным номером заказа и названием метода оплаты.

Сущность Model описывает модель продукта. Характеризуется непосредственно названием модели.

Сущность ProductType описывает тип продукта. Характеризуется непосредственно названием типа товара.

Сущность Vendor описывает производителя продукта. Характеризуется непосредственно названием производителя.

Сущность ManufacturerCountry описывает страну производителя товара. Характеризуется непосредственно названием страны производителя.

Сущность Storage описывает склад, на котором хранится товар. Характеризуется адресом и общей вместимостью.

Сущность Provider описывает поставщиков товаров. Характеризуется именем, номером телефона, электронной почтой.

Сущность Delivery описывает заказы, сделанные у поставщиков. Характеризуется идентификационным номером заказанного продукта, датой доставки, ценой и количеством заказанного товара.

Сущность Product описывает сам товар. Характеризуется идентификационным номером типа продукта, идентификационным номером страны производителя, идентификационным номером модели товара, ценой и гарантией.

Сущность ClientOrder является связующей сущностью между таблицами Client и Orders. Характеризуется идентификационным номером клиента и идентификационным номером заказа.

Сущность ProviderDelivery является связующей сущностью между таблицами Provider и Delivery. Характеризуется идентификационным номером поставщика и идентификационным номером заказа.

Сущность ProductInStorage является связующей сущностью между таблицами Product и Storage. Описывает на каком складе находится товар.

Характеризуется идентификационным номером склада и идентификационным номером продукта.

Сущность ProductVendor является связующей сущностью между таблицами Product и Vendor. Характеризуется идентификационным номером продукта и идентификационным номером производителя.

Сущность Admin описывает администраторов, которые имеют доступ к управлению базой данных. Характеризуется логином и паролем.

## **1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов**

Для построения инфологической концептуальной модели необходимо для каждой сущности, выявленной в предыдущем пункте, определить требуемый набор атрибутов. Атрибутом является наименованная характеристика сущности. Его наименование должно быть уникальным для конкретного типа сущности, но может быть одинаковым для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности [4].

Ниже представлены сущности и определенные для них атрибуты, а также ключи. Имена сущностей и атрибутов указываются, как они будут определены в созданной базе данных:

- Provider: id, name, phoneNumber, email.
- Client: id, name, phoneNumber, email.
- Vendor: id, name.
- Storage: id, address, capacity.
- Model: id, label.
- ManufacturerCountry: id, country.
- ProductType: id, productType.
- Product: id, productTypeID, manufacturerCountryID, modelID, price, warranty.
- Delivery: id, productID, deliveryDate, price, productCount.
- Orders: id, productID, deliveryDate, price.
- PaymentMethod: id, ordersID, method.
- ClientOrder: clientID, ordersID.
- ProviderDelivery: providerID, deliveryID.
- ProductInStorage: productID, storageID.
- ProductVendor: productID, vendorID.
- Admin: id, name, password.

### 1.3 Определение связей между объектами

Между двумя сущностями может быть установлена связь. Отношения между сущностями характеризуются глаголом, который можно применить для взаимодействия между ними.

Связь – это некое отношение между двумя типами сущностей. Связи создаются с помощью внешних ключей – «foreign key» [2].

Внешний ключ – это атрибут или набор атрибутов, которые ссылаются на «primary key» или «unique» другой таблицы. Другими словами, это указатель на строку другой таблицы.

Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Каждая связь имеет два конца и одно или два наименования. Наименование обычно выражается в неопределенной глагольной форме: «иметь», «принадлежать» и тому подобное. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся из-за их очевидности.

Связи делятся на:

- Многие ко многим. Для реализации связи многие ко многим нам нужен некий посредник между двумя рассматриваемыми таблицами. Он должен хранить два внешних ключа, первый из которых ссылается на первую таблицу, а второй – на вторую.

- Один ко многим: с обязательной связью; с необязательной связью. Сущность является общей для всех остальных, но может связываться с каждой сущностью не более одного раза.

- Один к одному: с обязательной связью; с необязательной связью. В такой связи отличительной особенностью является уникальность значения в столбце, который относится ко второй сущности.

Для реализации информационной системы станции необходимо установить все связи между объектами. Для этого нужно рассмотреть всю информационную систему в совокупности и определить отношения объектов, составляющих систему.

Проследить отношения, в которых состоят таблицы базы данных можно по схеме, изображенной на рисунке А.1 приложения А.

### 1.4 Описание полученной модели на языке инфологического проектирования

В деловой или личной сфере часто приходится работать с данными из различных источников, каждый из которых связан с определенным видом деятельности. В настоящее время благодаря огромным возможностям

					РДА.508190.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		7

компьютеров, которые связаны с хранением и обработкой больших массивов информации компьютер применяется для решения широкого круга задач буквально во всех сферах человеческой деятельности. Одновременно с развитием компьютерной техники развивалась и теория баз данных, которые представляют собой наборы взаимосвязанных данных о некоторой предметной области. Такие наборы имеют определенную структуру и постоянно хранятся в памяти компьютера.

В результате развития концепций баз данных были выделены три уровня представления информации: инфологический, даталогический и физический. На каждом уровне проводится структуризация информации таким образом, чтобы на третьем уровне информация могла быть представлена в виде структур данных, реализуемых в памяти ЭВМ. На инфологическом уровне определяется какая информация о предметной области будет храниться и обрабатываться в компьютере, а в результате исследования предметной области строится ее инфологическая модель, которая описывается в терминах классов объектов и их взаимодействий. В инфологической модели информация представляется вне зависимости от того, что представляют собой данные и какие технические средства будут использоваться в дальнейшем для ее хранения и обработки. Даталогическая и физическая модели непосредственно реализуются в системах управления СУБД, а физическая модель в свою очередь определяет структуру хранения данных на физических носителях. Цель инфологического проектирования заключается в представлении семантики (смысла) предметной области.

Для описания предметной области наиболее часто используется модель «сущность – связь», которую сокращенно называют «ER–моделью». «ER–диаграмма» имеет лексикографическую структуру и включает в себя текст и элементы графики. На практике инфологическая модель используется на первой стадии проектирования баз данных. При этом в терминах «ER–модели» описывается концептуальная схема, которая затем преобразуется к реляционной или любой другой схеме. «ER–модели» получили распространение в CASE-системах, поддерживающих проектирование реляционных баз данных [2].

Концептуальная модель проектируемой базы данных представлена на рисунке А.1 приложения А.

## 2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

### 2.1 Построение набора необходимых отношений базы данных

Чтобы построить схему реляционной базы данных необходимо определить совокупность отношений, которые составляют базу данных. Эта совокупность отношений будет содержать всю информацию, которая должна храниться в базе данных.

В предыдущем пункте мы создали инфологическую концептуальную модель базы данных магазина музыкальных инструментов. На основе полученной концептуальной модели можно определить набор необходимых отношений в базе данных.

### 2.2 Задание первичных и внешних ключей определенных отношений

В реляционной базе данных каждому объекту и сущности реального мира соответствуют кортежи отношений. И любое отношение должно обладать первичным ключом. Ключ – это минимальный набор атрибутов, по значениям которых можно однозначно найти требуемый экземпляр сущности.

Минимальность означает, что исключение из набора любого атрибута не позволяет идентифицировать сущность по оставшимся. Каждое отношение должно обладать хотя бы одним ключом. В таблице 2.1 определены первичные и внешние ключи для отношений.

Таблица 2.1 – Первичные и внешние ключи отношений

№	Название таблицы	Первичный ключ	Внешние ключи
1	Provider	id	Отсутствуют
2	Client	id	Отсутствуют
3	Vendor	id	Отсутствуют

## Продолжение таблицы 2.1

№	Название таблицы	Первичный ключ	Внешние ключи
4	Storage	id	Отсутствуют
5	Model	id	Отсутствуют
6	ManufacturerCountry	id	Отсутствуют
7	ProductType	id	Отсутствуют
8	Product	id	productTypeID, manufacturerCountryID, modelID
9	Delivery	id	productID
10	Orders	id	productID
11	PaymentMethod	id	ordersID
12	ClientOrder	clientID, ordersID	clientID, ordersID
13	ProviderDelivery	providerID, deliveryID	providerID, deliveryID



№	Название таблицы	Первичный ключ	Внешние ключи
14	ProductInStorage	productID, storageID	productID, storageID
15	ProductVendor	productID, vendorID	productID, vendorID
16	Admin	id	Отсутствуют

В дальнейшем построении схемы реляционной базы данных ключи будут служить для организации связей между отношениями.

### 2.3 Третья нормальная форма

Процесс преобразования базы данных к виду, отвечающему нормальным формам, называется нормализацией. Нормализация предназначена для приведения структуры базы данных к виду, обеспечивающему минимальную избыточность, то есть нормализация не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение объёма базы данных. Конечной целью нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных, информации.

Для реляционных баз данных необходимо, чтобы все отношения базы данных обязательно находились в первой нормальной форме. Нормальные формы более высокого порядка могут использоваться разработчиками по своему усмотрению. Однако грамотный специалист стремится к тому, чтобы довести уровень нормализации базы данных хотя бы до третьей нормальной формы, тем самым, исключив из базы данных избыточность и аномалии обновления.

Третья нормальная форма – не ключевые атрибуты не должны определять другие не ключевые атрибуты [3].

## 2.4 Определение ограничений целостности для внешних ключей и для отношений в целом

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

## 2.5 Графическое представление связей между внешними ключами

По результатам нормализации, определении первичных и внешних ключей, связей между сущностями, была получена схема реляционной базы данных, представленная в приложении Б на рисунке Б.1. На ней изображаются все отношения базы данных, а также связей между внешними и первичными ключами.

					РДА.508190.ПЗ	Лист
						12
Изм.	Лист	№ докум.	Подпись	Дата		

### 3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Для реализации спроектированной базы данных была выбрана система управления базами данных Microsoft SQL Server 2022. Связано это с тем, что данная СУБД является бесплатной и обладает широким спектром возможностей.

В создаваемой базе данных будут использоваться следующие типы данных:

- INTEGER – Целочисленный тип.
- DATE – Дата временной тип с поддержкой временной зоны.
- NVARCHAR – Строковый тип.
- DECIMAL – Дробное число, хранящееся в виде строки.

Опишем все таблицы, которые будут созданы в базе данных.

Таблица Provider содержит список всех поставщиков. Ее структура приведена в таблице 3.1.

Таблица 3.1 – Характеристика атрибутов таблицы Provider

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор поставщика
name	NVARCHAR(255)	Имя поставщика
phoneNumber	NVARCHAR(255)	Номер телефона поставщика
email	NVARCHAR(255)	Email поставщика

Таблица Client содержит список всех клиентов, сделавших заказ в магазине. Ее структура приведена в таблице 3.2.

Таблица 3.2 – Характеристика атрибутов таблицы Client

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор клиента
name	NVARCHAR(255)	Имя клиента
phoneNumber	NVARCHAR(255)	Номер телефона клиента
email	NVARCHAR(255)	Email клиента

Таблица Vendor содержит список всех производителей, инструменты которых, продаются в магазине. Ее структура приведена в таблице 3.3.

Таблица 3.3 – Характеристика атрибутов таблицы Vendor

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор производителя
name	NVARCHAR(255)	Название производителя

Таблица Storage содержит список всех складов. Ее структура приведена в таблице 3.4.

Таблица 3.4 – Характеристика атрибутов таблицы Storage

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор склада
address	NVARCHAR(255)	Адрес склада
capacity	INTEGER	Общая вместимость склада

Таблица Model содержит список всех моделей товаров. Ее структура приведена в таблице 3.5.

Таблица 3.5 – Характеристика атрибутов таблицы Model

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор модели
label	NVARCHAR(255)	Название модели

Таблица ManufacturerCountry содержит список всех стран, где производят товары, доступных к покупке. Ее структура приведена в таблице 3.6.

Таблица 3.6 – Характеристика атрибутов таблицы ManufacturerCountry

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор склада
country	NVARCHAR(255)	Название страны

Таблица ProductType содержит типы товаров, доступных к покупке. Ее структура приведена в таблице 3.7.

Таблица 3.7 – Характеристика атрибутов таблицы ProductType

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор типа продукта
productType	NVARCHAR(255)	Наименование типа продукта

Таблица Product содержит список всех товаров, доступных к покупке. Ее структура приведена в таблице 3.8.

Таблица 3.8 – Характеристика атрибутов таблицы Product

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор товара
productTypeID	INTEGER	Идентификатор типа товара

Имя атрибута	Тип	Описание
manufacturerCountryID	INTEGER	Идентификатор страны производителя
modelID	INTEGER	Идентификатор модели товара
price	DECIMAL	Цена товара
warranty	DATE	Дата окончания гарантии на товар

Таблица Delivery содержит список заказов, сделанных у поставщиков. Ее структура приведена в таблице 3.9.

Таблица 3.9 – Характеристика атрибутов таблицы Delivery

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор заказа
productID	NVARCHAR(255)	Идентификатор товара
deliveryDate	DATE	Дата привоза
price	DECIMAL	Цена у поставщика
productCount	INTEGER	Количество заказанного товара

Таблица Orders содержит список заказов, сделанных в магазине. Ее структура приведена в таблице 3.10.

Таблица 3.10 – Характеристика атрибутов таблицы Orders

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор заказа
productID	NVARCHAR(255)	Идентификатор товара
deliveryDate	DATE	Дата доставки
price	DECIMAL	Цена

Таблица PaymentMethod содержит список методов оплаты. Ее структура приведена в таблице 3.11.

Таблица 3.11 – Характеристика атрибутов таблицы PaymentMethod

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор метода оплаты
ordersID	INTEGER	Идентификатор товара
method	NVARCHAR(255)	Название метода оплаты

Таблица ClientOrder является ассоциативной таблицей между Client и Orders. Ее структура приведена в таблице 3.12.

Таблица 3.12 – Характеристика атрибутов таблицы ClientOrder

Имя атрибута	Тип	Описание
clientID	INTEGER	Идентификатор клиента
ordersID	INTEGER	Идентификатор заказа

Таблица ProviderDelivery является ассоциативной таблицей между Provider и Delivery. Ее структура приведена в таблице 3.13.

Таблица 3.13 – Характеристика атрибутов таблицы ProviderDelivery

Имя атрибута	Тип	Описание
providerID	INTEGER	Идентификатор поставщика
deliveryID	INTEGER	Идентификатор заказа у поставщика

Таблица ProductInStorage является ассоциативной таблицей между Product и Storage. Ее структура приведена в таблице 3.14.

Таблица 3.14 – Характеристика атрибутов таблицы ProductInStorage

Имя атрибута	Тип	Описание
productID	INTEGER	Идентификатор товара
storageID	INTEGER	Идентификатор склада

Таблица ProductVendor является ассоциативной таблицей между Product и Vendor. Ее структура приведена в таблице 3.15.

Таблица 3.15 – Характеристика атрибутов таблицы ProductVendor

Имя атрибута	Тип	Описание
productID	INTEGER	Идентификатор товара
vendorID	INTEGER	Идентификатор производителя

Таблица Admin содержит список администраторов, имеющих доступ к администрированию базы данных. Ее структура приведена в таблице 3.16.

Таблица 3.16 – Характеристика атрибутов таблицы Admin

Имя атрибута	Тип	Описание
id	INTEGER	Идентификатор клиента
name	NVARCHAR(255)	Логин администратора
password	NVARCHAR(255)	Пароль администратора

## 4 ЗАПИСЬ ВЫРАЖЕНИЙ, УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL

Описание запросов и их реализация указаны в названии листингов. Их содержании представлено ниже.

**Листинг 4.1 – Получить список поставщиков и их количество для указанной категории**

```
DECLARE @Category NVARCHAR(50) = '{comboBox2.Text}';
-- Получить список поставщиков и их количество для указанной
категории
SELECT
P.ID AS ProviderID,
P.Name AS ProviderName,
COUNT(*) AS TotalProductsSupplied
FROM
    Provider P
JOIN
    ProviderDelivery PD ON P.ID = PD.ProviderID
JOIN
    Delivery D ON PD.DeliveryID = D.ID
JOIN
    Product ON D.ProductID = Product.ID
JOIN
    ProductType PT ON Product.ProductTypeID = PT.ID
WHERE
    PT.ProductType = @Category
GROUP BY
    P.ID, P.Name;
```

**Листинг 4.2 – Получить общее количество поставщиков для указанной категории**

```
DECLARE @Category NVARCHAR(50) = '{comboBox2.Text}';
SELECT
    COUNT(DISTINCT P.ID) AS TotalProviders
FROM
    Provider P
JOIN
    ProviderDelivery PD ON P.ID = PD.ProviderID
JOIN
    Delivery D ON PD.DeliveryID = D.ID
JOIN
    Product ON D.ProductID = Product.ID
JOIN
    ProductType PT ON Product.ProductTypeID = PT.ID
WHERE
    PT.ProductType = @Category;
```

**Листинг 4.3 – Получить сведения о конкретном виде продукта: какими поставщиками поставляется, их расценки, время поставки**

```
DECLARE @Category NVARCHAR(50) = '{comboBox2.Text}';
SELECT
    P.Name AS ProviderName,
    D.Price AS ProviderPrice,
    D.DeliveryDate
FROM
    Provider P
JOIN
    ProviderDelivery PD ON P.ID = PD.ProviderID
JOIN
    Delivery D ON PD.DeliveryID = D.ID
JOIN
    Product ON D.ProductID = Product.ID
JOIN
    ProductType PT ON Product.ProductTypeID = PT.ID
WHERE
    PT.ProductType = @Category;
```

**Листинг 4.4 – Получить перечень и общее число покупателей, купивших указанный вид товара за некоторый период, сделавших покупку товара на сумму не менее указанной**

```
SELECT
    C.Name AS CustomerName,
    COUNT(DISTINCT O.ID) AS TotalPurchases
FROM
    Client C
JOIN
    ClientOrder CO ON C.ID = CO.ClientID
JOIN
    Orders O ON CO.OrdersID = O.ID
JOIN
    Product P ON O.ProductID = P.ID
JOIN
    ProductType PT ON P.ProductTypeID = PT.ID
WHERE
    PT.ProductType = '{productType}' AND
    O.DeliveryDate BETWEEN '{startDate.ToString("yyyy-MM-dd")}'
AND '{endDate.ToString("yyyy-MM-dd")}' AND
    O.Price >= {minPurchaseAmount}
GROUP BY
    C.ID, C.Name;
```

**Листинг 4.5 – Вывести в порядке возрастания десять самых продаваемых моделей продукции**

```
SELECT TOP 10
    M.Label AS ProductLabel, -- Используем ""Label"" из таблицы
    ""Model""
    SUM(D.ProductCount) AS TotalSold FROM
```



```

Product P
JOIN
    Model M ON P.ModelID = M.ID -- Связываем с таблицей ""Model""
по ModelID
JOIN
    Delivery D ON P.ID = D.ProductID
GROUP BY
    M.ID, M.Label -- Группируем по ModelID и Label
ORDER BY
    TotalSold DESC;

```

**Листинг 4.6 – Вывести десять самых дешевых поставщиков**

```

SELECT TOP 10
    Pr.Name AS ProviderName,
    AVG(D.Price) AS AverageDeliveryPrice
FROM
    Provider Pr
JOIN
    ProviderDelivery PD ON Pr.ID = PD.ProviderID
JOIN
    Delivery D ON PD.DeliveryID = D.ID
GROUP BY
    Pr.ID, Pr.Name
ORDER BY
    AverageDeliveryPrice ASC;

```

**Листинг 4.7 – Получить среднее число продаж на месяц по любому виду товара**

```

SELECT
    Subquery.ProductType,
    AVG(SalesPerMonth) AS AvgSalesPerMonth
FROM (
    SELECT
        PT.ProductType,
        DATEPART(MONTH, D.DeliveryDate) AS SaleMonth,
        COUNT(*) AS SalesPerMonth
    FROM
        Product P
    JOIN
        ProductType PT ON P.ProductTypeID = PT.ID
    JOIN
        Delivery D ON P.ID = D.ProductID
    GROUP BY
        PT.ProductType, DATEPART(MONTH, D.DeliveryDate)
) AS Subquery
GROUP BY
    Subquery.ProductType;

```

**Листинг 4.8 – Получить перечень, общее количество и стоимость товара, реализованного за конкретный день**

```
DECLARE @SpecificDate DATE = '{dateTime}';
SELECT
    P.ID AS ProductID,
    P.ProductTypeID,
    P.ManufacturerCountryID,
    P.ModelID,
    P.Price AS ProductPrice,
    P.Warranty,
    D.DeliveryDate,
    D.ProductCount,
    D.Price AS DeliveryPrice
FROM
    Delivery D
JOIN
    Product P ON D.ProductID = P.ID
WHERE
    D.DeliveryDate = @SpecificDate;
```

**Листинг 4.9 – Подсчитать, сколько пустых ячеек имеется на складе**

```
SELECT
    Storage.ID AS StorageID,
    Storage.Address,
    Storage.Capacity - COUNT(ProductInStorage.ProductID) AS
EmptyCells
FROM Storage
LEFT JOIN ProductInStorage ON Storage.ID =
ProductInStorage.StorageID
GROUP BY Storage.ID, Storage.Address, Storage.Capacity;
```

**Листинг 4.10 – Подсчитать, сколько склад сможет вместить товара**

```
SELECT
    Storage.ID AS StorageID,
    Storage.Address,
    COUNT(ProductInStorage.ProductID) AS OccupiedCells,
    Storage.Capacity AS TotalCells
FROM Storage
LEFT JOIN ProductInStorage ON Storage.ID =
ProductInStorage.StorageID
GROUP BY Storage.ID, Storage.Address, Storage.Capacity;
```

**Листинг 4.11 – Получить сведения о клиентских заказах в заданном временном диапазоне**

```
SELECT
    ClientOrder.ClientID,
    Client.Name AS ClientName,
    Orders.ID AS OrderID,
    Orders.DeliveryDate,
    Orders.Price
FROM ClientOrder
```

```
JOIN Orders ON ClientOrder.OrdersID = Orders.ID
JOIN Client ON ClientOrder.ClientID = Client.ID
WHERE Orders.DeliveryDate BETWEEN '{startDateTime}' AND
'{endDateTime}';
```

**Листинг 4.12 – Получить чистую прибыль за определенный период времени**

```
SELECT
    SUM(Orders.Price) AS TotalProfit
FROM Orders
WHERE Orders.DeliveryDate BETWEEN '{startDateTime}' AND
'{endDateTime}';
```

**Листинг 4.13 – Получить, в процентном соотношении, прибыль с каждого товара**

```
SELECT
    Product.ID AS ProductID,
    ProductType.ProductType,
    ManufacturerCountry.Country AS ManufacturerCountry,
    Model.Label AS ModelLabel,
    COALESCE((SUM(Orders.Price) - SUM(Delivery.Price)) /
SUM(Orders.Price) * 100, 0) AS ProfitPercentage
FROM Product
LEFT JOIN Orders ON Product.ID = Orders.ProductID
LEFT JOIN Delivery ON Product.ID = Delivery.ProductID
JOIN ProductType ON Product.ProductTypeID = ProductType.ID
JOIN ManufacturerCountry ON Product.ManufacturerCountryID =
ManufacturerCountry.ID
JOIN Model ON Product.ModelID = Model.ID
GROUP BY
    Product.ID,
    ProductType.ProductType,
    ManufacturerCountry.Country,
    Model.Label;
```

**Листинг 4.14 – Получить список клиентов по суммарному количеству потраченных денег**

```
SELECT
    Client.ID AS ClientID,
    Client.Name AS ClientName,
    SUM(Orders.Price) AS TotalSpent
FROM Client
LEFT JOIN ClientOrder ON Client.ID = ClientOrder.ClientID
LEFT JOIN Orders ON ClientOrder.OrdersID = Orders.ID
GROUP BY Client.ID, Client.Name
ORDER BY TotalSpent DESC;
```

**Листинг 4.15 – Получить скорость оборота денежных средств, вложенных в товар**

```
SELECT
    Product.ID AS ProductID,
    ProductType.ProductType,
    ManufacturerCountry.Country AS ManufacturerCountry,
```

```

Model.Label AS ModelLabel,
SUM(Delivery.ProductCount) AS TotalSold,
Storage.Capacity AS Inventory,
CAST(SUM(Delivery.ProductCount) AS DECIMAL) /
CAST(Storage.Capacity AS DECIMAL) AS TurnoverRate
FROM Product
JOIN ProductType ON Product.ProductTypeID = ProductType.ID
JOIN ManufacturerCountry ON Product.ManufacturerCountryID =
ManufacturerCountry.ID
JOIN Model ON Product.ModelID = Model.ID
LEFT JOIN ProductInStorage ON Product.ID =
ProductInStorage.ProductID
LEFT JOIN Storage ON ProductInStorage.StorageID = Storage.ID
LEFT JOIN Delivery ON Product.ID = Delivery.ProductID
GROUP BY
    Product.ID,
    ProductType.ProductType,
    ManufacturerCountry.Country,
    Model.Label,
    Storage.Capacity;

```

#### Листинг 4.16 – Получить, в днях, остаточную гарантию на товар

```

SELECT
    Product.ID AS ProductID,
    ProductType.ProductType,
    ManufacturerCountry.Country AS ManufacturerCountry,
    Model.Label AS ModelLabel,
    Orders.DeliveryDate AS OrderDeliveryDate,
    Product.Warranty AS WarrantyEndDate,
    DATEDIFF(day, Orders.DeliveryDate, Product.Warranty) AS
DaysBetweenDeliveryAndWarranty,
    Client.Name AS ClientName,
    Client.PhoneNumber AS ClientPhoneNumber,
    Client.Email AS ClientEmail
FROM Product
JOIN ProductType ON Product.ProductTypeID = ProductType.ID
JOIN ManufacturerCountry ON Product.ManufacturerCountryID =
ManufacturerCountry.ID
JOIN Model ON Product.ModelID = Model.ID
JOIN Orders ON Product.ID = Orders.ProductID
JOIN ClientOrder ON Orders.ID = ClientOrder.OrdersID
JOIN Client ON ClientOrder.ClientID = Client.ID;

```

#### Листинг 4.17 – Получение списка методов оплаты и количества оплаченных, данным способом, товаров

```

SELECT
    PaymentMethod.Method AS PaymentMethod,
    COUNT(*) AS NumberOfPayments
FROM PaymentMethod GROUP BY PaymentMethod.Method;

```

## 5 ВЫБОР И ОБОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Выбор средств разработки приложения очень важен. Правильный выбор средств разработки существенно влияет на производительность, масштабируемость и обслуживание приложения в долгосрочной перспективе.

СУБД Microsoft SQL Server 2022 была выбрана в качестве основного инструмента для создания базы данных. Microsoft SQL Server является одной из самых популярных систем управления базами данных, которая обеспечивает высокую производительность, надежность и безопасность. Она поддерживает широкий спектр функций, включая транзакционную обработку, аналитику, интеграцию данных и другие.

C# был выбран в качестве основного языка программирования для разработки приложения. C# является объектно-ориентированным языком программирования, который обеспечивает простоту использования, мощные функции и высокую производительность. Он поддерживает широкий спектр функций, включая обработку исключений, перегрузку операторов, делегаты, события и другие.

WinForms был выбран в качестве фреймворка для создания пользовательского интерфейса. WinForms обеспечивает богатый набор элементов управления, которые позволяют создавать интерактивные и функциональные пользовательские интерфейсы.

Microsoft Visual Studio 2022 была выбрана в качестве интегрированной среды разработки. Visual Studio предлагает мощные функции для написания, отладки и тестирования кода, а также поддерживает широкий спектр языков программирования и фреймворков.

					РДА.508190.ПЗ	Лист
						23
Изм.	Лист	№ докум.	Подпись	Дата		

## 6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ

### 6.1 Разработка и построение интерфейса главной и рабочих форм

Основная форма приложения представлена в виде окна, большую часть площади которого занимает таблица. Навигация между таблицами происходит с помощью комбинированного списка в левой верхней части окна. Кнопки основных функций программы, такие как добавление, редактирование и удаление записей, находятся в верхнем меню основного окна.

При запуске программы пользователь попадает на форму авторизации, с которой можно перейти на основную форму. Главная форма отобразится в случае успешной авторизации в системе, то есть при наличии данного пользователя в соответствующей таблице.

Все формы добавления и редактирования выполнены процедурно, для каждой из таблиц, по причине разного количества столбцов и типов данных. Процедурное добавление форм накладывает некоторые сложности в реализации. Однако, позволяет не создавать огромное количество дополнительных форм для каждой функции.

При проектировании приложения были учтены все возможные случаи некорректной работы программы. Например, программа выведет диалоговое окно с ошибкой, если введенные данные не соответствуют нужному типу.

Скриншоты, а также описание работы всех окон приложения, за исключением окон добавления и редактирования для каждой формы, по причине их аналогичности, в плане проектирования, представлены в приложении Г.

### 6.2 Построение главного меню и кнопок панели инструментов

Навигационная панель программы представлена пунктами: «Управление», «Готовые запросы», «Свой запрос», «О программе». Данные пункты выполнены в виде меню, находящегося сверху основного окна. В свою очередь, пункт меню «Управление» состоит из 4 подпунктов: «Добавить», «Изменить», «Удалить», «Отобразить все данные».

Для навигации по записям базы данных используются два комбинированных меню, таблица, отображающая данные, и графический элемент – список, отображающий записи в выбранной таблице.

### 6.3 Выполнение программного кода на Microsoft Visual C#

Далее следует описание работы программы с базой данных. Основные методы, для работы с базами данных, описаны в классе MainForm. Они связываются с базой данных при помощи класса SqlConnection. Подключение к базе данных начинается с формирования строки подключения. Код класса MainForm представлен в листинге 6.1.

Листинг 6.1 – Класс MainForm

```
public partial class MainForm : Form
{
    private const string ConnectionString = "Data
Source=LAPTOP\\SQLEXPRESS;Initial Catalog=shop;Integrated
Security=True";
    string currentItemComboBox1;

    public MainForm()
    {
        InitializeComponent();
        LoadTableNames();
        comboBox1.SelectedIndex = 0;
        comboBox2.SelectedIndex = 0;
    }

    private void LoadTableNames()
    {
        using (SqlConnection connection = new
SqlConnection(ConnectionString))
        {
            connection.Open();

            string query = "SELECT TABLE_NAME FROM
INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE'";
            SqlDataAdapter adapter = new SqlDataAdapter(query,
connection);
            DataTable tableNameTable = new DataTable();
            adapter.Fill(tableNameTable);

            foreach (DataRow row in tableNameTable.Rows)
            {
                if (row["TABLE_NAME"].ToString() != "sysdiagrams")
                {
                    comboBox1.Items.Add($"{row["TABLE_NAME"]}");
                }
            }
        }
    }
}
```

```

private void comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
{
    listBox1.Enabled = false;

    string selectedItem = comboBox1.SelectedItem.ToString();

    currentItemComboBox1 = selectedItem;

    using (SqlConnection connection = new
SqlConnection(connectionString))
    {
        connection.Open();

        string query = $"SELECT * FROM {selectedItem}";
        SqlDataAdapter adapter = new SqlDataAdapter(query,
connection);
        DataTable dataTable = new DataTable();
        adapter.Fill(dataTable);

        listBox1.Items.Clear();

        foreach (DataRow row in dataTable.Rows)
        {
            listBox1.Items.Add($"{row[0]}");
        }

        query = $"SELECT COLUMN_NAME FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME = '{selectedItem}'";
        adapter = new SqlDataAdapter(query, connection);
        DataTable columnNamesTable = new DataTable();
        adapter.Fill(columnNamesTable);

        comboBox2.Items.Clear();

        foreach (DataRow row in columnNamesTable.Rows)
        {
            comboBox2.Items.Add(row["COLUMN_NAME"].ToString());
        }

        comboBox2.SelectedIndex = 0;
    }

    listBox1.Enabled = true;
}

private void comboBox2_SelectedIndexChanged(object sender,
EventArgs e)
{
    string selectedItem = comboBox2.SelectedItem.ToString();

```



```

        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            connection.Open();

            string query = $"SELECT * FROM {comboBox1.Text}";
            SqlDataAdapter adapter = new SqlDataAdapter(query,
connection);
            DataTable dataTable = new DataTable();
            adapter.Fill(dataTable);

            listBox1.Items.Clear();

            foreach (DataRow row in dataTable.Rows)
            {
                listBox1.Items.Add($"{row[selectedItem]}");
            }
        }

        private void listBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            if (listBox1.SelectedItem != null) {
                string selectedColumnName =
listBox1.SelectedItem.ToString();

                using (SqlConnection connection = new
SqlConnection(connectionString))
                {
                    connection.Open();

                    string query = $"SELECT * FROM {comboBox1.Text}
WHERE {comboBox2.Text} = '{selectedColumnName}'";
                    SqlDataAdapter adapter = new SqlDataAdapter(query,
connection);

                    DataTable dataTable = new DataTable();
                    adapter.Fill(dataTable);

                    dataGridView1.DataSource = null;

                    dataGridView1.DataSource = dataTable;
                }
            }
        }

        private void MainForm_FormClosed(object sender,
FormClosedEventArgs e)
        {

```

```

        Application.Exit();
    }

    private void добавитьToolStripMenuItem_Click(object sender,
    EventArgs e)
    {
        DataTable dataTable = new DataTable();

        using (SqlConnection connection = new
        SqlConnection(ConnectionString))
        {
            connection.Open();

            string query = $"SELECT COLUMN_NAME FROM
        INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME =
        '{currentItemComboBox1}'";
            SqlDataAdapter adapter = new SqlDataAdapter(query,
        connection);
            adapter.Fill(dataTable);

            AddProcedForm addProcedForm = new AddProcedForm(dataTable,
        comboBox1.Text);
        }

        private void изменитьToolStripMenuItem_Click_1(object sender,
        EventArgs e)
        {
            DataTable dataTable = new DataTable();

            using (SqlConnection connection = new
            SqlConnection(ConnectionString))
            {
                connection.Open();

                string query = $"SELECT COLUMN_NAME FROM
            INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME =
            '{currentItemComboBox1}'";
                SqlDataAdapter adapter = new SqlDataAdapter(query,
            connection);
                adapter.Fill(dataTable);

                List<string> comboBox2Items = new List<string>();

                foreach (var item in comboBox2.Items)
                {
                    comboBox2Items.Add(item.ToString());
                }
            }
        }
    }

```

```

        EditProcedForm editProcedForm = new
EditProcedForm(dataTable, comboBox1.Text, comboBox2,
listBox1.SelectedIndex);
    }

    private void удалитьToolStripMenuItem_Click_1(object sender,
EventArgs e)
    {
        string deleteQuery = $"DELETE FROM {comboBox1.Text} WHERE
{comboBox2.Text} = {listBox1.SelectedItem}";

        using (SqlConnection connection = new
SqlConnection(ConnectionString))
        {
            connection.Open();

            using (SqlCommand command = new SqlCommand(deleteQuery,
connection))
            {
                try
                {
                    int rowsAffected = command.ExecuteNonQuery();
                    if (rowsAffected > 0)
                    {
                        MessageBox.Show($"Данные успешно удалены из
базы данных.", "", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                    }
                    else
                    {
                        MessageBox.Show($"Не удалось найти запись с
указанным первичным ключом.", "", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    }
                }
                catch (Exception ex)
                {
                    MessageBox.Show($"Не удалось удалить данные из
базы данных.", ex.Message.ToString(), MessageBoxButtons.OK,
MessageBoxIcon.Error);
                }
            }
        }

        private void отобразитьВсеДанныеToolStripMenuItem_Click(object
sender, EventArgs e)
    {

```

```

        using (SqlConnection connection = new
SqlConnection(connectionString))
        {
            connection.Open();

            string query = $"SELECT COLUMN_NAME FROM
INFORMATION_SCHEMA.COLUMNS WHERE TABLE_NAME =
'{comboBox1.SelectedItem.ToString()}'";
            SqlDataAdapter adapter = new SqlDataAdapter(query,
connection);
            DataTable columnNamesTable = new DataTable();
            adapter.Fill(columnNamesTable);

            query = $"SELECT * FROM {comboBox1.Text}";
            adapter = new SqlDataAdapter(query, connection);
            DataTable dataTable = new DataTable();
            adapter.Fill(dataTable);

            dataGridView1.DataSource = null;

            dataGridView1.DataSource = dataTable;
        }
    }

    private void oПрограммеToolStripMenuItem_Click_1(object sender,
EventArgs e)
    {
        MessageBox.Show("Программа была разработана студентом
группы 20-ИТ-1 Резниченко Олегом, в рамках курсового проекта.", "О
программе", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    private void запросыToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        QueriesForm queriesForm = new QueriesForm(this);
        this.Hide();
        queriesForm.ShowDialog();
    }

    private void свойЗапросToolStripMenuItem_Click(object sender,
EventArgs e)
    {
        CustomQuery customQuery = new CustomQuery(this);
        this.Hide();
        customQuery.ShowDialog();
    }
}

```

## ЗАКЛЮЧЕНИЕ

В процессе выполнения данной курсовой работы были закреплены навыки проектирования баз данных и реализации их в MS SQL Server 2022.

Были определены основные цели системы в соответствии с выбранным вариантом и выделены требования, которым должна удовлетворять система.

Спроектированная база данных соответствует всем требованиям, которые предъявляются в задании. Данная база предоставляет возможность просмотра, редактирования, удаления и добавления записей. Разработанное приложение имеет интуитивно понятный интерфейс, учитывает возможные ошибки пользователя, а также имеет высокую отзывчивость.

Информационная система магазина музыкальных инструментов предоставляет информацию о клиентах, поставщиках, заказах и товарах.

В результате выполнения курсовой работы были спроектированы и реализованы: база данных информационной системы магазина музыкальных инструментов, программа, которая эффективно и корректно взаимодействует с базой данных. Программное средство, для управления базой данных, реализовано с помощью языка программирования C#.

Таким образом, выполнение данной курсовой работы позволило не только закрепить знания по проектированию баз данных и их реализации в MS SQL Server 2022, но также разработать полнофункциональную информационную систему, обеспечивающую удобное управление данными магазина музыкальных инструментов. Полученные навыки в проектировании и программировании с использованием C#, MS SQL Server и Winforms предоставляют отличную основу для создания эффективных и надежных приложений в будущем.

					РДА.508190.ПЗ	Лист
Изм.	Лист	№ докум.	Подпись	Дата		31

## СПИСОК ЛИТЕРАТУРЫ

1 Webonto [Электронный ресурс] – Режим доступа: <https://webonto.ru/kontseptualnaya-model-bazyi-dannyih/>. Дата обращения: 15.11.2023.

2 BestProg [Электронный ресурс] – Режим доступа: <https://www.bestprog.net/ru/2019/01/27/er-model-the-concept-of-relationship>. Дата обращения: 18.11.2023.

3 Habr [Электронный ресурс] – Режим доступа: <https://habr.com/ru/articles/254773/>. Дата обращения: 22.11.2023.

					<i>РДА.508190.ПЗ</i>	Лист
						32
Изм.	Лист	№ докум.	Подпись	Дата		

## Концептуальная схема базы данных

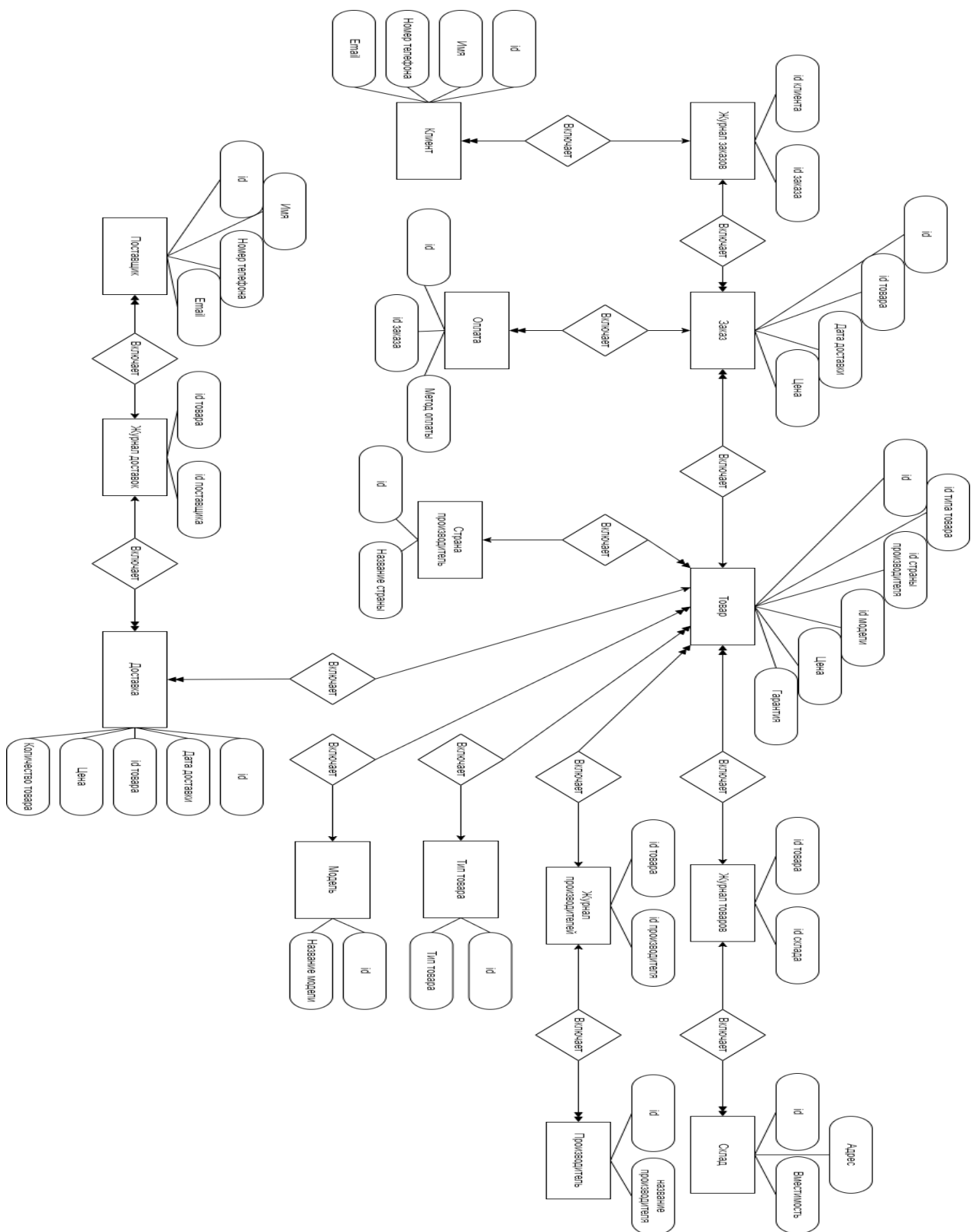


Рисунок А.1 – Концептуальная модель проектируемой базы данных

# **ПРИЛОЖЕНИЕ Б** (обязательное) **Схема реляционной базы данных**

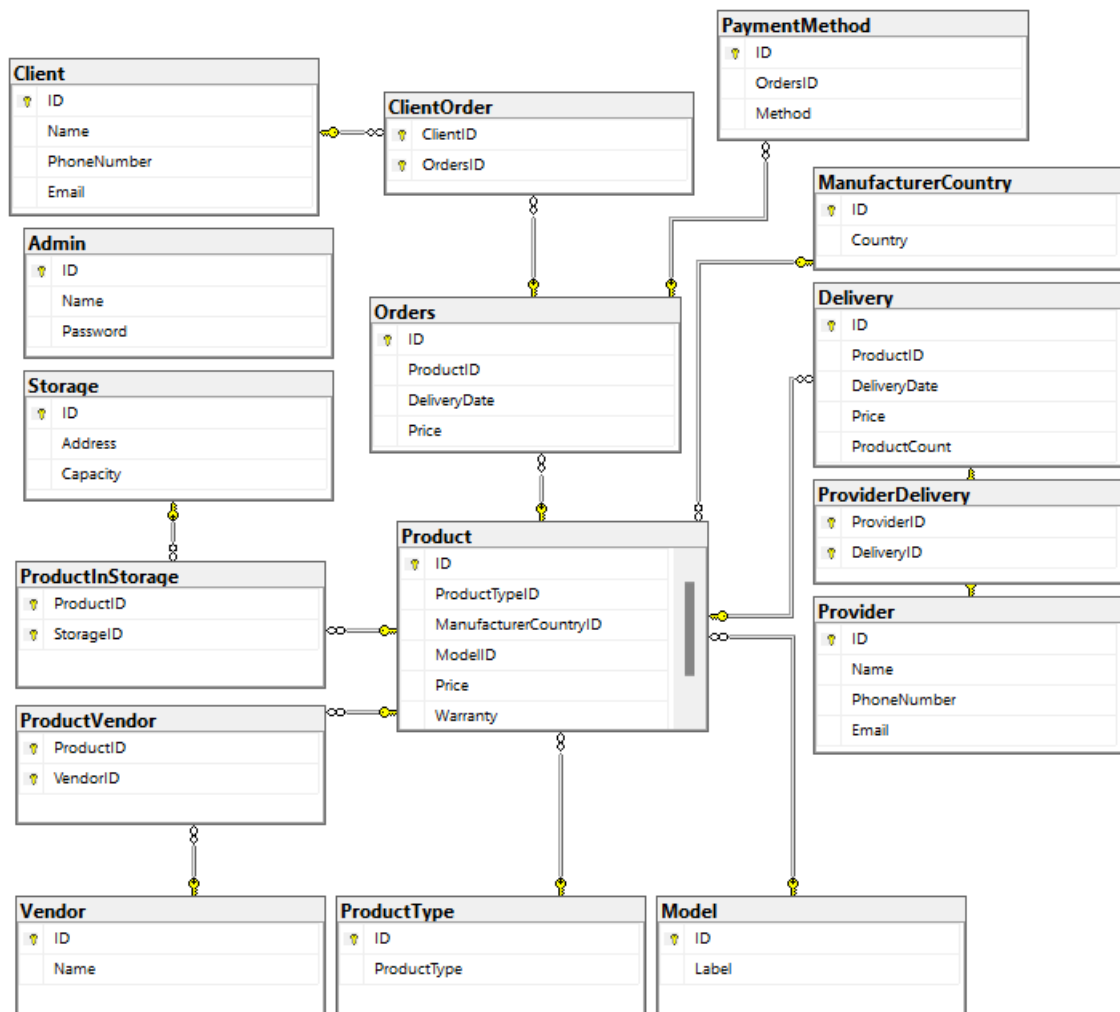


Рисунок Б.1 – Схема реляционной базы данных



## ПРИЛОЖЕНИЕ В

(обязательное)

### Главная и рабочие формы приложения

В ходе проектирования было определено, что при запуске приложения будет открываться форма авторизации в системе с текстовыми полями для логина и пароля, а также с маркером, который позволяет скрыть или показать вводимый пароль. Форма авторизации представлена на рисунке В.1.

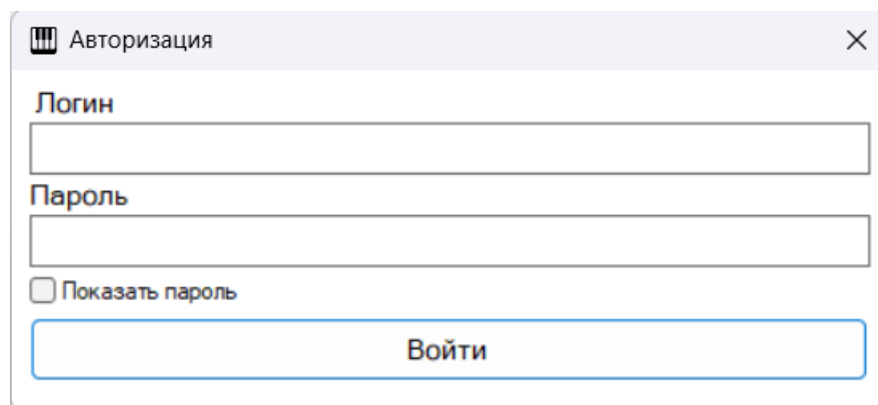


Рисунок В.1 – Окно авторизации

В случае успешной авторизации, появится главное окно. Главная форма приложения представлено на рисунке В.2.

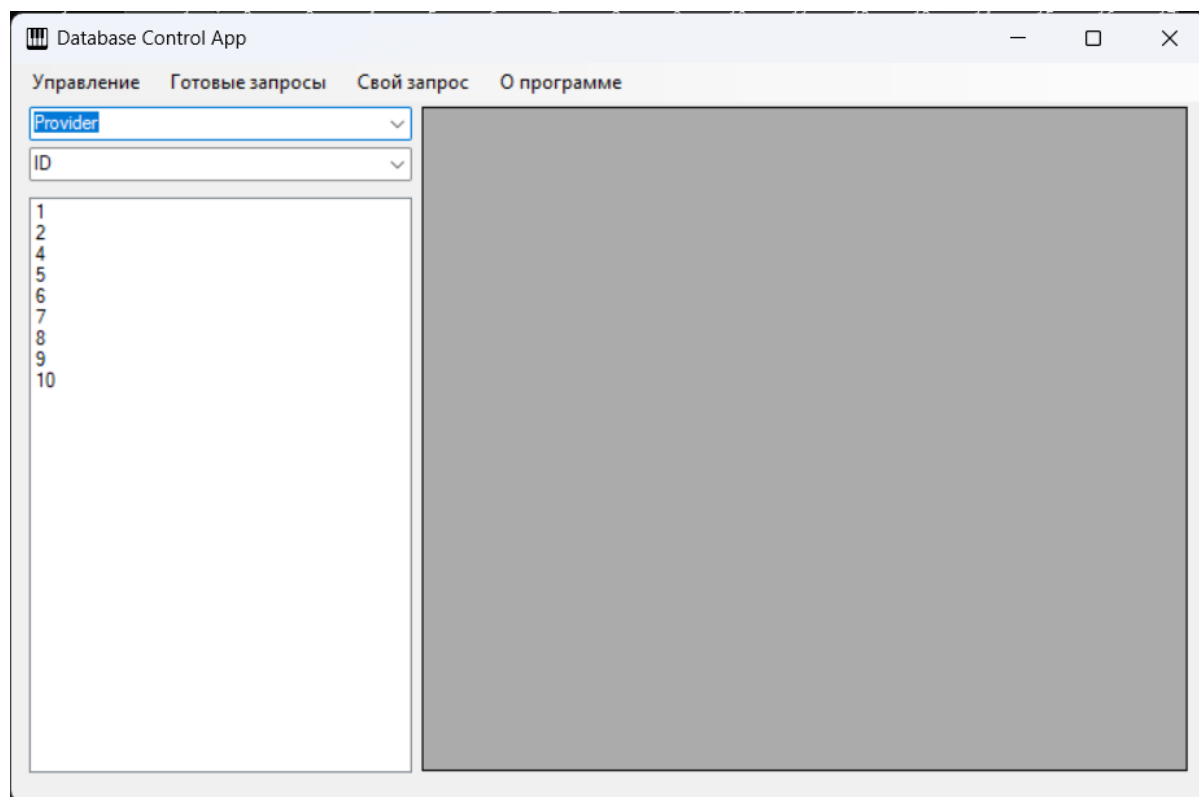


Рисунок В.2 – Главная форма приложения

Выбор таблицы, по которой необходимо получить данные, производится с помощью верхнего поля со списком. Нижнее поле со списком позволяет отфильтровать записи по названию столбца и вывести данные в основной список.

При выборе записи в основном списке, таблица данных покажет все данные этой записи в выбранной таблице. Пример отображения данных по выбранной записи представлен на рисунке В.3.

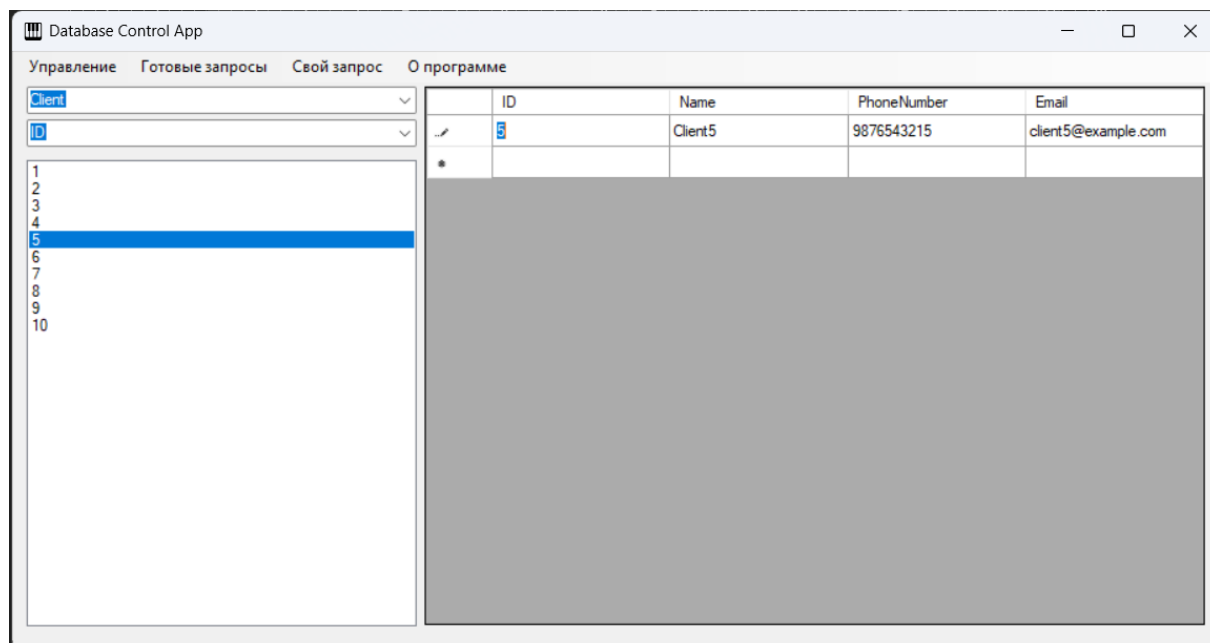


Рисунок В.3 – Отображение данных по выбранной записи

Для отображения всех данных, по выбранной таблице, необходимо перейти в пункт «Управление», а затем нажать кнопку «Отобразить все данные». Результат отображения всех записей выбранной таблицы представлен на рисунке В.4.

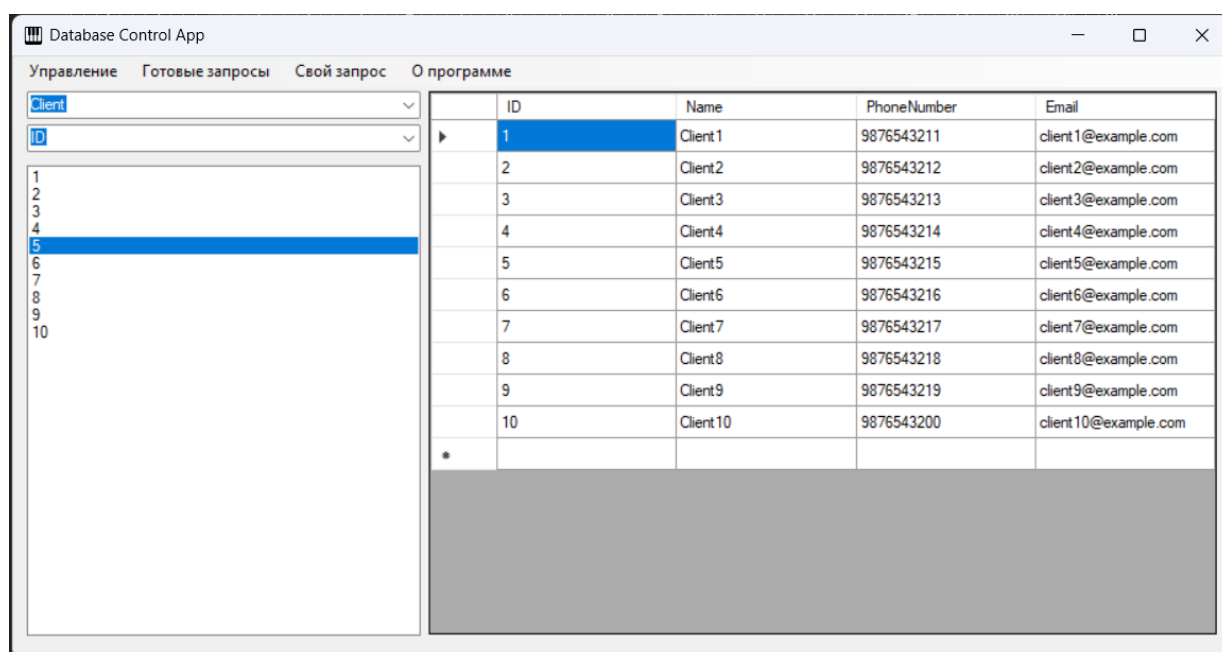


Рисунок В.4 – Отображение всех записей выбранной таблицы

Для добавления записи необходимо перейти в пункт «Управление» и нажать на кнопку «Добавить». После данных манипуляций появится новая форма «Добавление данных». Далее необходимо заполнить предоставленные поля и нажать на кнопку «Сохранить». После чего запись добавится в базу данных и появится информационное окно о успешном добавлении данных. Пример добавления записи в базу данных представлен на рисунке В.5.

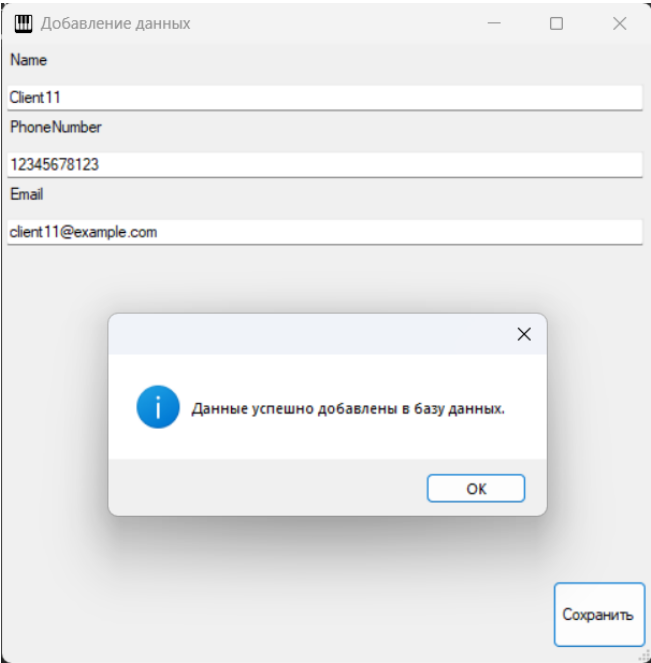


Рисунок В.5 – Добавление записи в базу данных

Для удаления записи необходимо перейти в пункт «Управление» и нажать на кнопку «Удалить», выбрав, заранее, запись из списка. После чего запись удалится из базы данных и появится информационное окно о успешном удалении записи. Пример удаления записи из базы данных представлен на рисунке В.6.

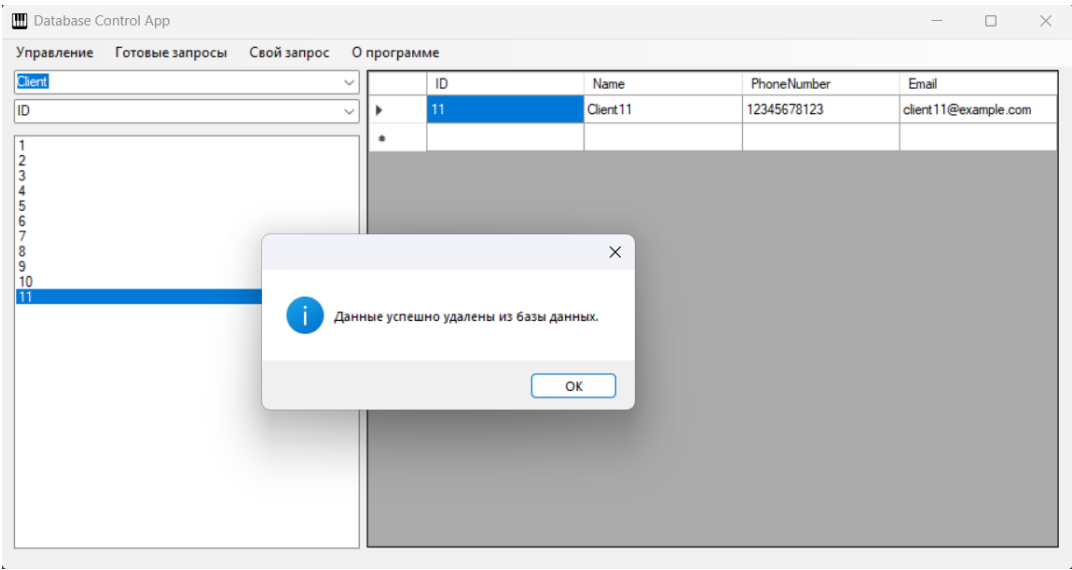


Рисунок В.6 – Удаление записи из базы данных

Для изменения записи необходимо перейти в пункт «Управление» и нажать на кнопку «Изменить», выбрав запись из списка. После, появится новая форма «Изменение данных». Далее необходимо изменить предоставленные поля и нажать на кнопку «Сохранить». После чего запись обновится в базе данных и появится информационное окно о успешном изменении данных. Пример изменения записи в базе данных представлен на рисунке В.7.

Изменение данных в таблице Client

Выберите ID

2

ID

2

Name

Client15

PhoneNumber

9876543212

Email

client2@example.com

Успешно

Данные успешно обновлены

ОК

Сохранить

Рисунок В.7 – Изменение записи в базе данных

При нажатии на кнопку «Готовые запросы». Появится новая форма с двумя полями со списком. В верхнем поле со списком элементы – запросы из задания к данному курсовому проекту. При выборе элемента из поля со списком и нажатии кнопки «Обновить», в таблице будут выведены данные, соответствующие запросу. При этом, если для получения записей из запроса необходимы некоторые дополнительные данные, появится диалоговое окно с необходимыми полями. Пусть программа выполнит первый запрос для примера. Результат выполнения первого запроса, из списка, представлен на рисунке В.8.

Запросы

Обновить

1. Получить список поставщиков и их количество для указанной категории.

Type 1

	ProviderID	ProviderName	TotalProductsSupplied
▶	1	Provider1	1
*			

Рисунок В.8 – Результат выполнения первого запроса

При нажатии, в главной форме, кнопки «Свой запрос» откроется новая форма. Данная форма позволяет написать свой запрос к базе данных. Для этого в поле с текстом введите запрос и нажмите кнопку «Применить». Пример выполнения своего запроса, к базе данных, представлен на рисунке В.9.

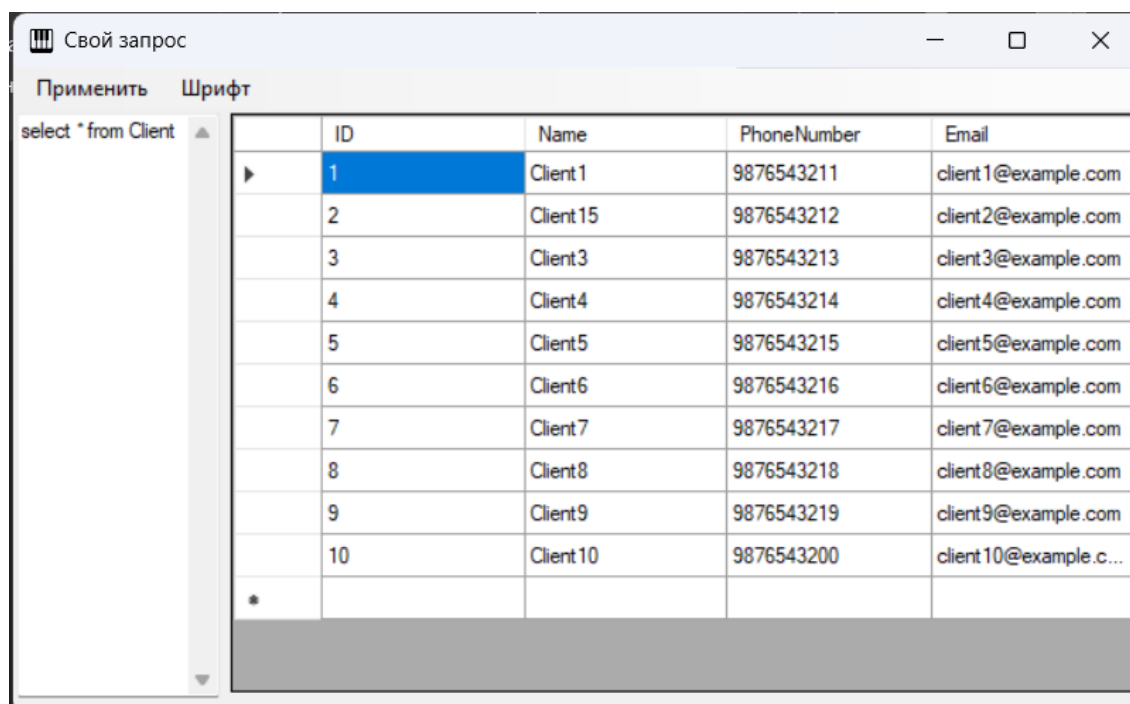


Рисунок В.9 – Выполнение своего запроса

Также можно изменить размер текста в поле, где вводится запрос. Для этого выберите пункт «Шрифт» и с помощью соответствующих кнопок изменяйте размер текста.

Последняя функция – вывод информации о приложении. Для этого перейдите в главную форму и нажмите кнопку «О программе». Появится диалоговое окно с необходимой информацией. Пример получения информации о программе представлен на рисунке В.10.

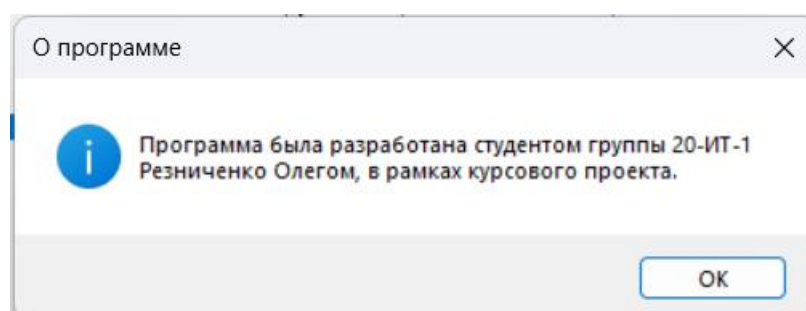


Рисунок В.10 - Получения информации о программе