государственное автономное профессиональное образовательное учреждение   
Чувашской Республики «Межрегиональный центр компетенций –  
 Чебоксарский электромеханический колледж» Министерства образования и    
молодежной политики Чувашской Республики

Дисциплина МДК.11.01 Технология разработки и защита базы данных

КУРСОВОЙ ПРОЕКТ

Технология разработки и защиты базы данных

автоматизированной информационной системы складского учета

КП.Ип1-21.06.МДК.11.01.ПЗ

Выполнил студент 3 курса, группы Ип5-21

Иваков Алексей Петрович

(Фамилия И. О.)

(подпись) (чч.мм.гггг)

Преподаватель Игнатьева Т.А.

(Фамилия И. О.)

Защищен

(чч.мм.гггг)

с оценкой

Подпись

(подпись) (расшифровка подписи)

2022

СОДЕРЖАНИЕ

[Введение 3](#_Toc121320230)

[1 Инфологическое проектирование 4](#_Toc121320231)

[1.1 Фнализ и описание предметной области 4](#_Toc121320232)

[1.2 Разработка концептуальной модели 5](#_Toc121320233)

[2 Определение требований к операционной системе 9](#_Toc121320234)

[3 Выбор субд и других программных средств 12](#_Toc121320235)

[4 Логическое проектирование реляционной базы данных 13](#_Toc121320236)

[5 Реализация проекта базы данных 29](#_Toc121320237)

[Заключение 37](#_Toc121320238)

[Список использованных источников 38](#_Toc121320239)

[Приложение А ­– Логическая моедель 39](#_Toc121320240)

[Приложение Б – Реалиционная модель 40](#_Toc121320241)

[Приложение В – Код создания таблиц 41](#_Toc121320242)

[Приложение Г – Код приложения 48](#_Toc121320243)

ВВЕДЕНИЕ

В настоящее время эффективное управление магазином бытовых товаров невозможно без грамотной организации и обработки данных. Для этих целей прекрасно подходит использование информационной системы, основанной на базе данных.

База данных для складского учета магазина бытовых товаров – это система, предназначенная для хранения, обработки и управления информацией о товарах, их количестве, поступлении, реализации и других сопутствующих параметрах. Введение такой системы в деятельность магазина позволяет оптимизировать процессы, улучшить точность учета и повысить общую эффективность.

Сфера управления складским учетом в магазинах бытовых товаров также требует внимания к эффективной обработке информации. Необходимо правильно использовать возможности для автоматизации процессов учета товаров, отслеживания поступлений и списаний, а также создания отчетов для эффективного управления складской деятельностью.

Внедрение баз данных в деятельность магазина создает возможность улучшить процессы учета и повысить общую эффективность. Таким образом, актуальной становится тема: "Технология разработки и защиты баз данных для складского учета магазина бытовых товаров".

Создание приложения для управления складским учетом даст возможность в любой момент времени просматривать информацию о товарах, редактировать, добавлять новые товары, удалять устаревшие записи, а также осуществлять фильтрацию данных для удобства работы с ассортиментом товаров.

Цель: разработать базу данных для интернета провайдера.

Задачи:

* произвести анализ и описать предметную область;
* разработать концептуальную и логическую модели; составить список сущностей и атрибутов, построить реляционную модель на основе логической модели;
* спроектировать базу данных с использованием средств SQL;
* разработать пользовательский интерфейс.

1 Инфологическое проектирование

1.1 Анализ и описание предметной области

Концептуальное проектирование – это построение модели предметной   
области, ориентированной на восприятие человека и независимой от типа системы управления базами данных (СУБД) и типа самой базы данных.

Под предметной областью (ПО) понимается совокупность связанных между собой функций, задач управления в некоторой области деятельности предприятия, с помощью которых достигается выполнение поставленной цели.

Множество фрагментов составляют из себя предметную область. Каждый фрагмент предметной области характеризуется множеством объектов и процессов, использующих объекты, а также множеством пользователей, с разными взглядами на ПО.

Модель предметной области представляет собой знания о предметной   
области. Знания могут быть неформальными и формальными. Модель предметной области описывает скорее процессы, происходящие в предметной области и   
данные, используемые этими процессами. От того, насколько правильно   
смоделирована предметная область, зависит успех дальнейшей разработки   
приложений.

Система создается для информационного обеспечения и учета деятельности преподавателей.

1.2 Разработка концептуальной модели

Для разработки базы данных необходимо определить основные сущности и необходимые поля. Так же следует выделить сопоставимые поля для определения связей между таблицами.

База данных должна содержать данные:

* о контрактах, подписанных с пользователями;
* о пользователях;
* о денежных транзакция, совершаемые пользователями;
* о местах подключения пользователей;
* о тарифах, доступных пользователям для подключения.

Учитывая все вышеперечисленное, база данных должна содержать следующие сущности и их атрибуты.

Сущность «Товары»:

* «Код\_товара»;
* «Код\_категории»;
* «Название»;
* «Цена».

Сущность «Категории\_товаров»:

* «Код\_категории»;
* «Название\_категории»;
* «Описание».

Сущность «Поставки»:

* «Код\_поставщиуа»;
* «Код\_товара»;
* «Дата\_поставки».

Сущность «Склады»:

* «Код\_склада»;
* «Название склада»;
* «Адрес».

Сущность «Товары\_на\_складе»:

* «Код\_товара»;
* «Код\_склада»;
* «Количество\_на\_складе».

Сущность «Постащики»:

* «Код\_Поставщика»;
* «Название\_компании»;
* «Адрес»;
* «Контактные\_данные».

Таким образом, были определены основные объекты будущей информационной системы.

Установим связи между сущностями и распишем их с помощью ERD-инструкции:

* «Категории товара» и «Товары», связь «один-ко-многим». Бизнес-правила:

1. каждый товар (Товары) может относиться только к одному категорию товара (Категории товара);
2. каждая категория (Категории товара) может иметь один или несколько товаров (Товары).

* «Склады» и «Товары\_на\_складе», связь «один-ко-многим». Бизнес-правила:

1. каждый товар (Товары\_на\_складе) может относиться только к одному складу (Склады);
2. каждый склад (Склады) может содержать один или несколько товаров (Товары\_на\_складе).

* «Товары» и «Товары\_на\_складе», связь «один-ко-многим». Бизнес-правила:

1. каждый товар (Товары) может относиться к одному или нескольким складам (Товары\_на\_складе);
2. каждый склад (Товары\_на\_складе) может содержать один или несколько товаров (Товары).

* «Поставки» и «Поставщики», связь «один-ко-многим».

Бизнес-правила:

1. каждая поставка (Поставки) может быть связана только с одним поставщиком (Поставщики);
2. каждый поставщик (Поставщики) может осуществлять одну или несколько поставок (Поставки).

* «Поставки» и «Поставщики», связь «один-ко-многим».

Бизнес-правила:

1. каждая поставка (Поставки) может быть связана только с одним товаром (Товары);
2. каждый товар (Товары) может быть связан с одной или несколькими поставками (Поставки).

При осуществлении нормализации текущих отношений было осуществлено приведение к нормальным формам:

* первая нормальная форма (1NF) — все атрибуты должны иметь только одно значение;
* вторая нормальная форма (2NF) — отношения находятся в первой нормальной форме и все не ключевые атрибуты зависят от первичного ключа;
* третья нормальная форма (3NF) — отношения находятся во второй нормальной форме и все неключевые атрибуты не транзитивно зависят от первичного ключа.

Нормализация – это процесс преобразования отношений базы данных к виду, отвечающему нормальным формам.

Нормализация предназначена для приведения структуры баз данных к виду, обеспечивающему минимальную логическую избыточность, и не имеет целью уменьшение или увеличение производительности работы или же уменьшение или увеличение физического объёма базы данных.

Под избыточностью понимают повторение данных в разных строках одной таблицы или в разных таблицах БД. Итогом нормализации является уменьшение потенциальной противоречивости, хранимой в базе данных информации.

В заключение главы выделим все выполненные в процессе создания концептуальной модели действия:

* были определены объекты базы данных и их атрибуты;
* были определены связи между объектами;
* были определены действия необходимые для приведения отношений в базе данных к нормальной форме.

2 Определение требований к операционной системе

По данной формуле будет вычисляться объем памяти, требуемый для хранения данных, на основании данных из таблицы 1:

𝑛

Мд = 2 ∑ 𝑙𝑖 ∗ (𝑁𝑖 + 𝑁𝑎𝑖 )

𝑖=1

где 𝑙𝑖 - длина записи в i-ой таблице (в байтах);

𝑁𝑖 - примерное (максимально возможное) количество записей в i-ой таблице;

𝑁𝑎 – количество записей в архиве i-ой таблицы;

*n* – количество таблиц в БД.

Таблица 2.1 – Атрибуты, отображающие длину записи в каждой таблице

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Отношения | Атрибут | Тип данных | Размер,  байт | Среднее  кол-во | Объем,  байт |
| Товары | Категории\_товара\_Код\_категории | Int | 2 |  |  |
| Код\_товара | int | 2 |
| Название | nvarchar (50) | 50 |
| Цена | int | 2 |
| Цена | int | 2 |
| Категории\_товаров | Код\_категории | Int | 2 |  |  |
| Название\_категории | nvarchar (50) | 100 |
| Описание | nvarchar (50) | 100 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
| Поставки | Поставщики\_Код\_поставщика | Int | 2 |  |  |
| Товары\_Код\_товара | Int | 2 |
| Дата\_поставки | Date | 3 |
| Количество\_товара | Varchar(50) | 50 |
| Склады | Код\_склада | Int | 2 |  |  |
| Название\_склада | Nvarchar (50) | 50 |
| Адрес | Nvarchar (50) | 2 |
| Corpus | Int | 2 |
| Flat | Int | 2 |
| Поставщики | Код\_поставщика | Int | 2 |  |  |
| Название\_компании | Nvarchar (50) | 50 |
| Адрес | Nvarchar (50) | 50 |
| Контактные\_данные | Nvarchar (50) | 50 |
| Товары\_на\_складе | Количество\_на\_складе | Int | 2 |  |  |
| Склады\_Код\_склада | Int | 2 |
| Товары\_Код\_товара | Int | 3 |

Таким образом, объем памяти хранения данных за один год, составит:

МС = 2 × (6000 + 2900 + 525 + 6000 + 4050) = 19474 байт

Полученный объем памяти не будет являться постоянным, потому что он увеличивается или уменьшается по ходу работы с базой данных.

После проведенных вычислений становится ясно, что объем данных в информационной системе небольшой. Следовательно, для реализации проекта подойдет любой современный компьютер.

Объем оперативной памяти, который необходим, будет определяться требованиями выбранной системой управления баз данных, следовательно, СУБД MS SQL подходит для введения данной базы данных.

Для поддержания целостности данных выполняется резервное копирование баз данных. План резервного копирования приведен в таблице 2.2.

Таблица 2.2 – План резервного копирования

|  |  |
| --- | --- |
| День недели | Вид резервного копирования |
| Понедельник | Разностное резервное копирование (поставок, товаров и товаров на складе) |
| Вторник | Разностное резервное копирование (поставок, товаров и товаров на складе) |
| Среда | Разностное резервное копирование (поставок, товаров и товаров на складе) |
| Четверг | Разностное резервное копирование (поставок, товаров и товаров на складе) |
| Пятница | Разностное резервное копирование (поставок, товаров и товаров на складе) |
| Суббота | Разностное резервное копирование (поставок, товаров и товаров на складе) |
| Воскресенье | Полное резервное копирование всей БД |

В Microsoft SQL Server принята практика разных типов резервного копирования. Пользователям доступно:

* полное – делается резервная копия всей БД;
* дифференциальное или разностное – осуществляется копирование данных с того момента, когда осуществлялось ее последнее полное резервирование;
* логов или инкрементальное.

3 Выбор СУБД и других программных средств

При создании базы данных и клиентского приложения были решены важные и сложные задачи, для успешного выполнения которых нужно было правильно сделать выбор средства реализации проекта. Для создания концептуальной и логической моделей базы данных Oracle SQL Developer Data Modeler предоставил много возможностей. Приложение является бесплатным графическим программным обеспечением, с помощью которого можно сконструировать различные модели данных, такие как логические, реляционные, многомерные, сетевые и т. д.

СУБД MS SQL Server стала отличным выбором для создания, использования и управления будущей базы данных. Обладая не только всеми необходимыми функциями для создания клиентского приложения, но и имея высокую производительность, она стала самым удачным выбором, так как другие приложения компании Microsoft могут работать с ней без каких-либо проблем.

Следовательно, наилучшим вариантом после выбора СУБД стала программная среда Microsoft Visual Studio с использованием объектно-ориентированного языка программирования C# для разработки пользовательского интерфейса и программного кода, и комфортной работы пользователей с базой данных.

Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows.

Язык программирования C# был выбран из-за его перспективности среди других языков и обстоятельств, облегчающих дальнейшее сопровождение программного обеспечения.

4 Логическое проектирование реляционной базы данных

Логическое проектирование является созданием схемы баз данных на основе конкретной модели данных, то есть в данном случае реляционной модели данных, представленных на рисунках 4.1 и 4.2.

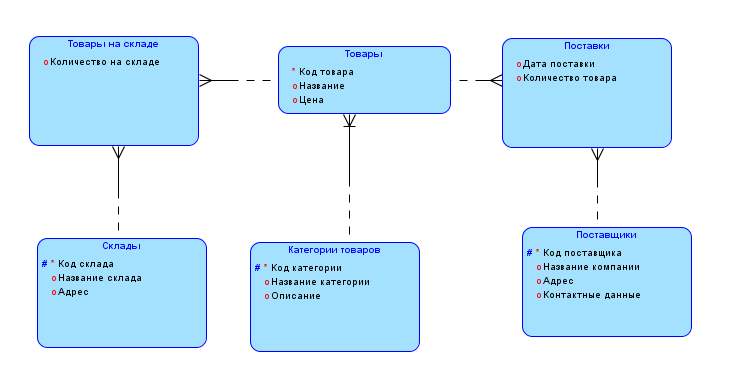


Рисунок 4.1 – Логическая модель базы данных

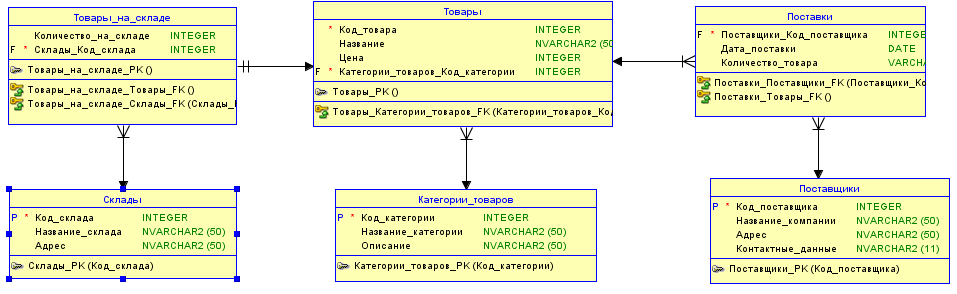


Рисунок 4.2 – Реляционная модель базы данных

Сущностью в базе данных может являться практически любой объект, который возможно логически выделить из предметной области, а ее атрибутом именованный столбец таблицы. Особыми атрибутами считаются первичные и внешние ключи.

Первичный ключ – это столбец, значения которого однозначно определяет строку.

Внешний ключ – это столбец в данной таблице, значения которого совпадает со значениями столбца, являющийся первичным ключем в другой таблице.

Во время разработки структуры базы данных были созданы таблицы, представления, хранимые процедуры и триггеры.

В базе данных таблица является совокупностью, связанной между собой информации, хранящейся в определенном порядке. Любая таблица состоит из кортежей (строк) и атрибутов (столбцов).

Представленная база данных состоит из шести таблиц:

* Таблица «Товары»;
* Таблица «Категории\_товаров»;
* Таблица «Склады»;
* Таблица «Поставщики»;
* Таблица «Товары\_на\_складе»;
* Таблица «Поставки».

Таблица «Товары» содержит информацию о товарах, их характеристиках и связанных с ними данными, такими как цена и название. Таблица «Поставщики» содержит информацию о поставщиках товаров, включая их контактные данные, адреса и другие релевантные сведения, а таблица «Поставки» - о дате поставок. Таблица "Категории товаров" собирает данные о категориях товаров, таких как электроника, одежда, кухонные приборы и т. д. Таблица "Склады" представляет собой данные о складах, включая их местоположение и другие соответствующие сведения, а таблица «Товары на складе» - о том, в каком количестве они хранятся на складе.

Все таблицы созданы в одной базе данных. Описание структуры таблиц и их полей приведены в таблицах 4.1 – 4.5.

Таблица 4.1 – Структура таблицы «Товары»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Код\_товара | Int | 2 | Идентифицирующий  код товара | Первичный  ключ |
| Название | Varchar(50) | 50 | Название товара | NOT NULL |
| Цена | Int | 2 | Стоимость | NOT NULL |
| Категории\_товаров\_Код\_категории | Int | 2 | Идентифицирующий  код товара | Внешний ключ |

Таблица 4.2 – Структура таблицы «Категории товара»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Код\_категории | Int | 2 | Идентифицирующий  код товара | Первичный  ключ |
| Название\_категории | Varchar(50) | 50 | Название категории товара | NOT NULL |
| Описание | Varchar(300) | 300 | Описание товара | NOT NULL |

Таблица 4.3 – Структура таблицы «Товары на складе»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Количество\_на\_складе | Int | 2 | Количестов товаров на складе | Первичный  ключ |
| Код\_склада | Int | 2 | Идентифицирующий  номер склада | NOT NULL |
| Код\_товара | Int | 2 | Идентифицирующий  код товара | Первичный  ключ |

Таблица 4.4 – Структура таблицы «Склады»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Код\_склада | Int | 2 | Идентифицирующий  номер склада | Первичный  ключ |
| Название\_склада | Varchar(50) | 50 | Название склада | NOT NULL |
| Адрес | Varchar(50) | 50 | Местонахождение склада | NOT NULL |

Таблица 4.5 – Структура таблицы «Поставщики»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Код\_поставщика | Int | 2 | Идентифицирующий  код поставщика | Первичный  ключ |
| Название\_компании | Varchar(50) | 50 | Название тарифа | NOT NULL |
| Адрес | Varchar(50) | 50 | Местонахождение | NOT NULL |
| Контактные данные | Varchar(15) | 15 | Номер телефона | NOT NULL |

Таблица 4.6 – Структура таблицы «Поставки»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Имя поля | Тип | Длина | Содержание | Ограничения |
| Поставщики\_Код\_поставщика | Int | 2 | Идентифицирующий  код поставщика | Внешний  ключ |
| Товары\_Код\_товара | Int | 2 | Идентифицирующий  код товара | Внешний  ключ |
| Дата\_поставки | Date | 3 | Местонахождение | NOT NULL |
| Количество\_товара | Varchar(50) | 50 | Номер телефона | NOT NULL |

После создания таблиц и добавления данных в MS SQL Server база данных выглядит следующим образом рисунок 4.3, рисунок 4.4, рисунок 4.5.

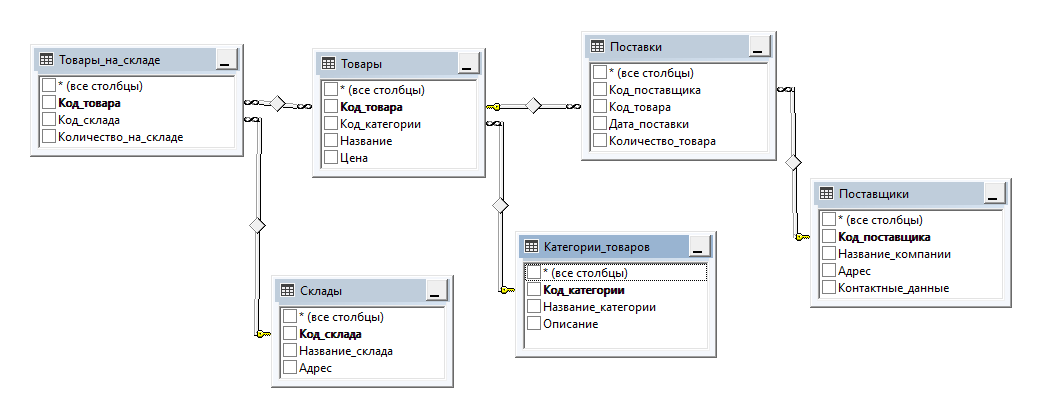


Рисунок 4.3 – Диаграмма базы данных

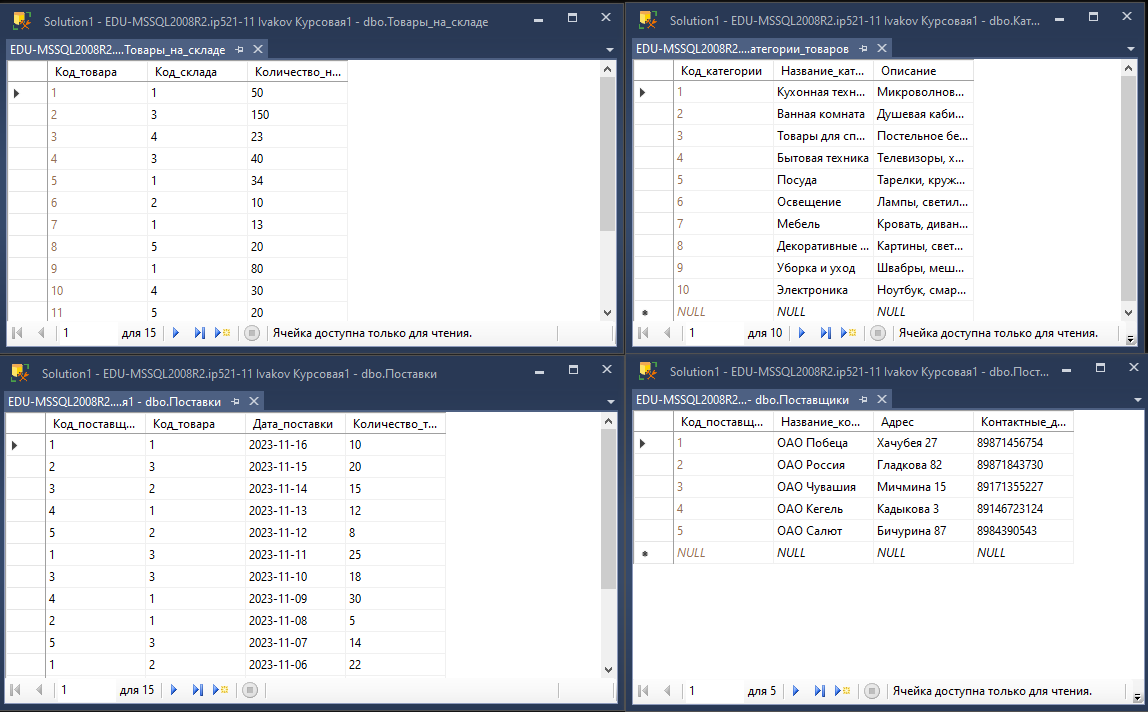


Рисунок 4.4 – Таблицы базы данных

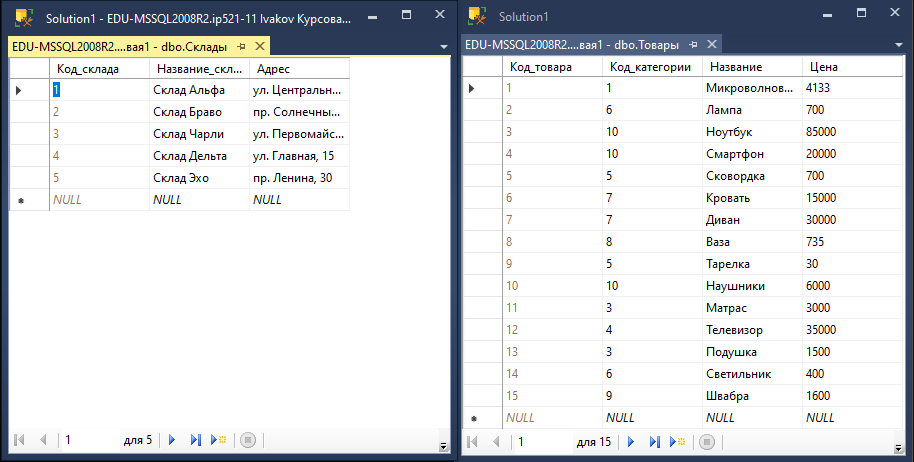


Рисунок 4.5 – Продолжение таблиц базы данных

Представление по своему внешнему виду напоминает таблицу, но на самом деле является не самостоятельным объектом, а результатом запроса на выборку к существующей в базе данных таблице и полностью зависит от нее, так как само по себе представление не может содержать никаких данных.

Созданные представления в данном проекте выглядят следующим образом:

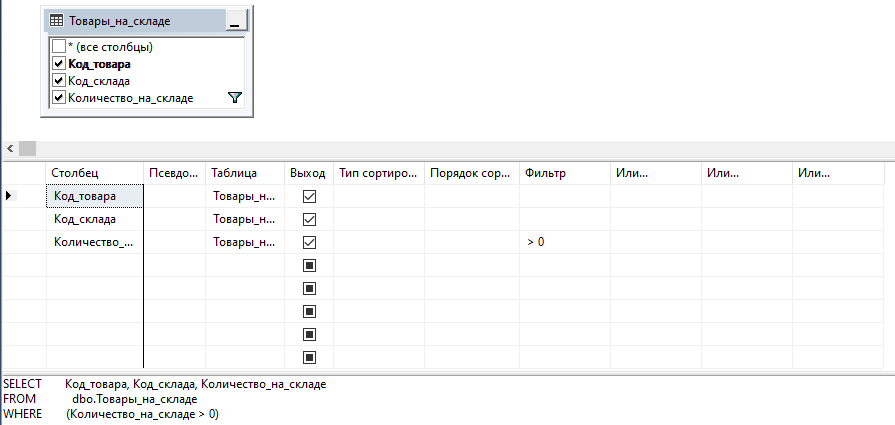


Рисунок 4.5 – Представление «Доступные\_товары\_на\_складе»

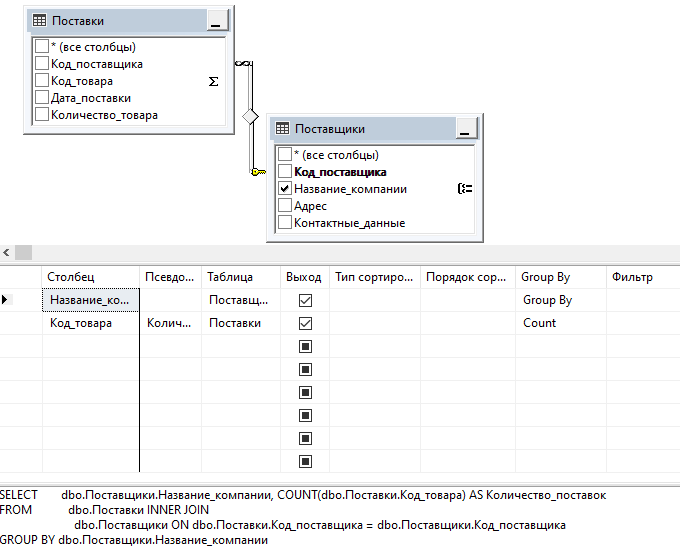


Рисунок 4.6 – Представление «Количество товаров по поставщикам»

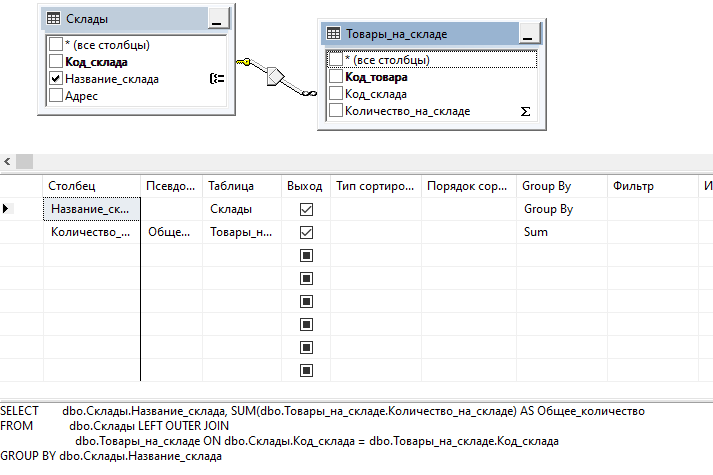


Рисунок 4.7 – Представление «Просмотр общего количества на складах»

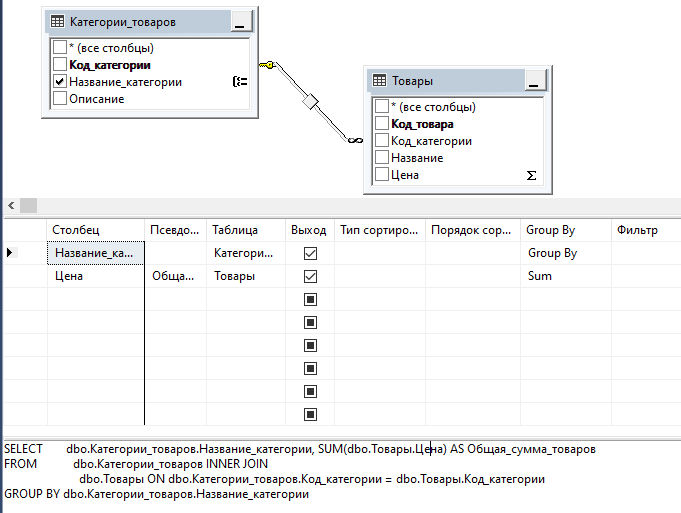


Рисунок 4.8 – Представление «Просмотр общей суммы товаров по категориям»

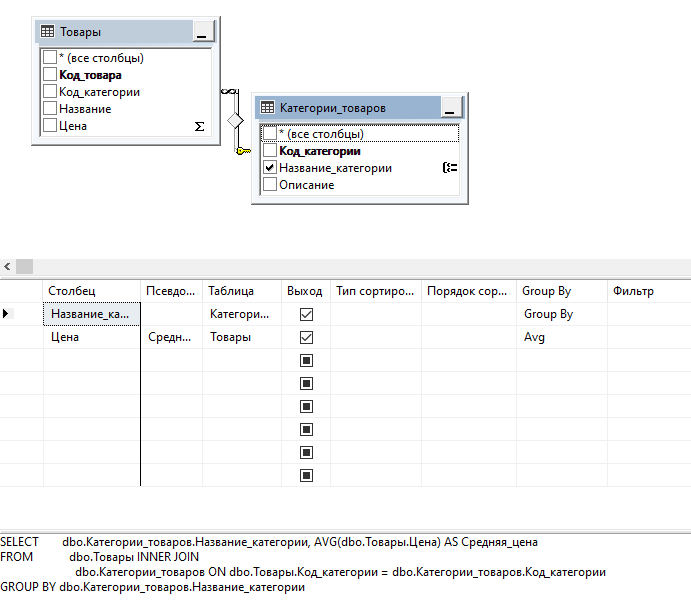


Рисунок 4.9 – Представление «Средняя цена товаров по категориям»

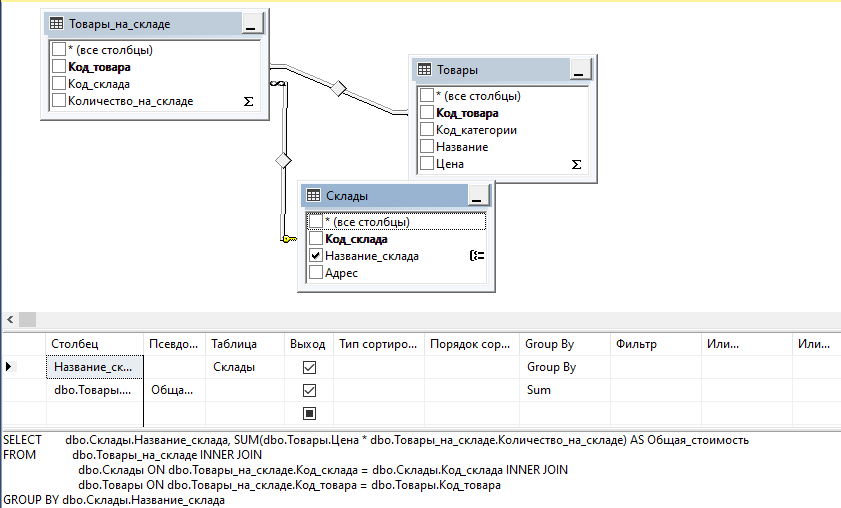


Рисунок 4.10 – Представление «Стоимость товаров на складе»

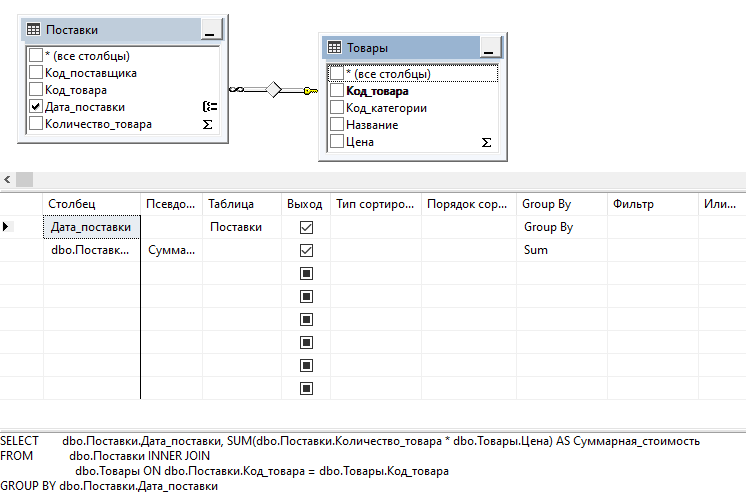


Рисунок 4.11 – Представление «Суммарная стоимость поставок по дате»

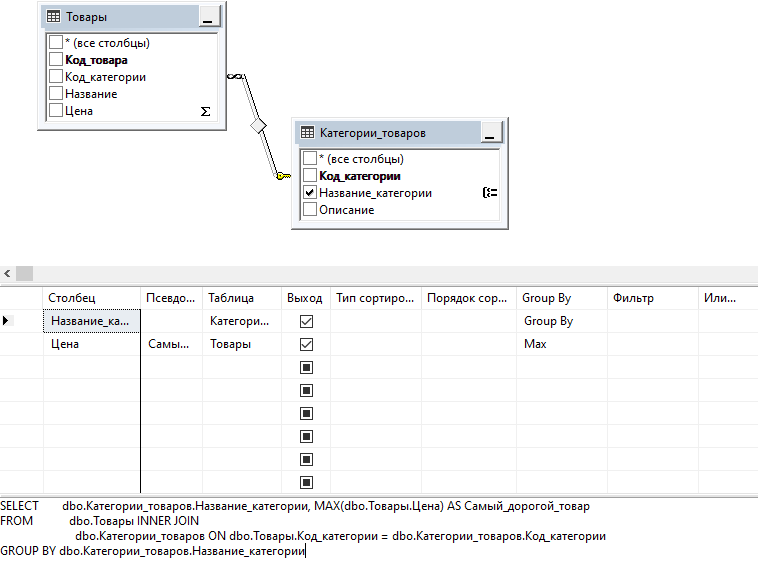


Рисунок 4.12 – Представление «Товары самый дорогой по категориям»

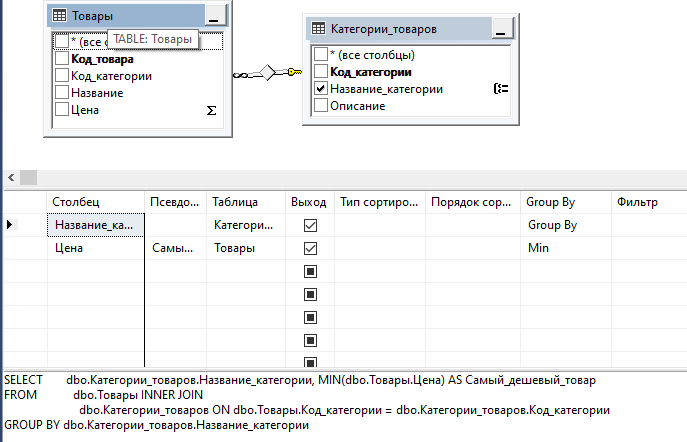


Рисунок 4.13 – Представление «Товар самы дешевый по категориям»

Хранимая процедура – это набор команд, состоящий из одного или нескольких операторов SQL, сохраняющий в БД в откомпилированном виде. Такие процедуры могут иметь входные и выходные параметры, а также локальные переменные.

При разработке курсового проекта были созданы следующие хранимые процедуры:

* Найти товар по названию рисунок 4.14. Данная хранимая процедура позволяет найти товар по названию.

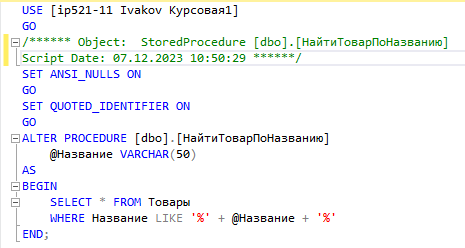


Рисунок 4.14 – Скрипт на создание хранимой процедуры «Найти товар по названию»

* Обновить товар рисунок 4.15. Данная хранимая процедура позволяет обновлять данные в таблице Товары .

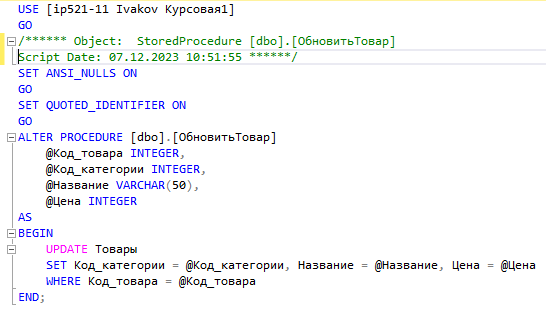


Рисунок 4.15 – Скрипт на создание хранимой процедуры «Обновить товар»

* Получить все товары рисунок 4.16. Данная хранимая процедура позволяет посмотреть все товары.

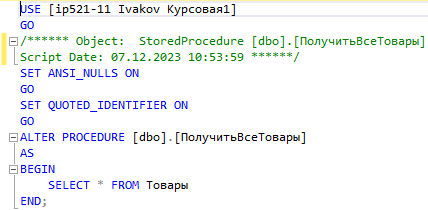


Рисунок 4.16 – Скрипт на создание хранимой процедуры «Получить все товары»

Получить товар по идентификатору рисунок 4.17. Данная хранимая процедура позволяет получить товар по номеру идентификатора.

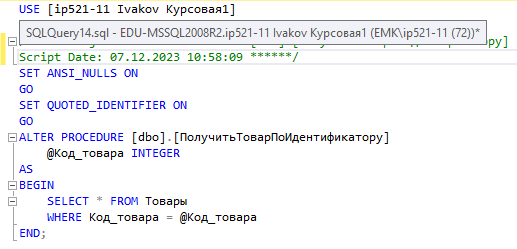


Рисунок 4.17 – Скрипт на создание хранимой процедуры «Получить товар по идентификатору»

Получить товар по категории 4.18. Данная хранимая процедура позволяет получить товар по номеру категории.

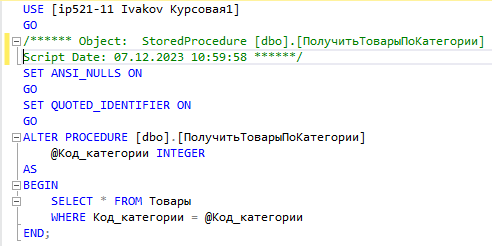


Рисунок 4.18 – Скрипт на создание хранимой процедуры «Получить товар по категории»

Создать товар рисунок 4.19. Данная хранимая процедура позволяет создавать товар.

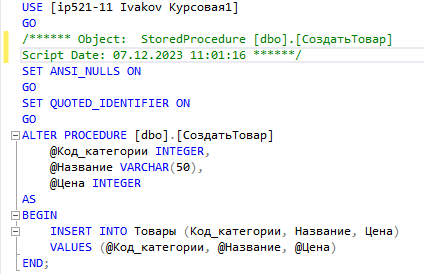
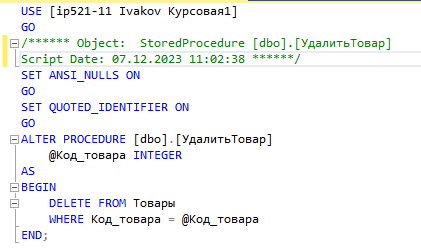


Рисунок 4.17 – Скрипт на создание хранимой процедуры «Создать товар»

Удалить товар рисунок 4.19. Данная хранимая процедура позволяет создавать товар.



Триггер – это вид хранимой процедуры, выполняющийся автоматически при возникновении событий языка обработки данных в БД. Триггеры используются для проверки целостности данных, а также для отката транзакций. Существует несколько видов: вставка, удаление или редактирование информации   
в определенной таблице или во всей базе данных.

Также в курсовом проекте были созданы триггеры:

* Предотвращение удаления категории, тригер не дает удалить категорию, код создания триггера представлен на рисунке 4.20.

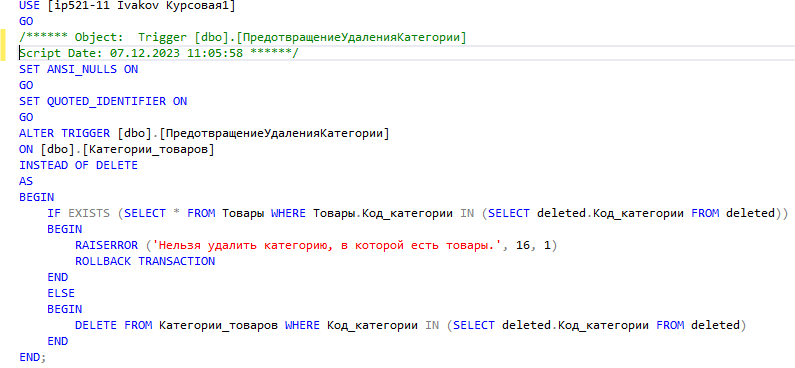


Рисунок 4.20 – Скрипт на создание триггера «Предотвращение удаления категории»

* Обновления количества на складе при удалении товара, обновляет количество товара на складе при его уалении, код данного триггера показан на рисунке 4.21.

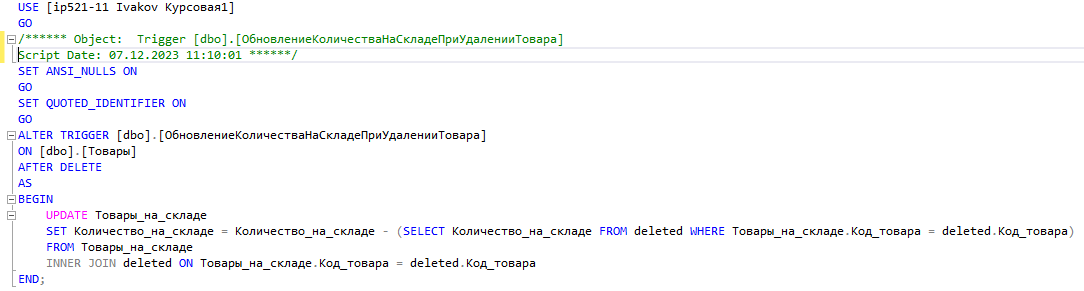


Рисунок 4.21 – Скрипт на создание триггера «

Обновления количества на складе при удалении товара»

* Проверка уникального названия в категории, проверяет есть ли такое название в категории, рисунок 4.22

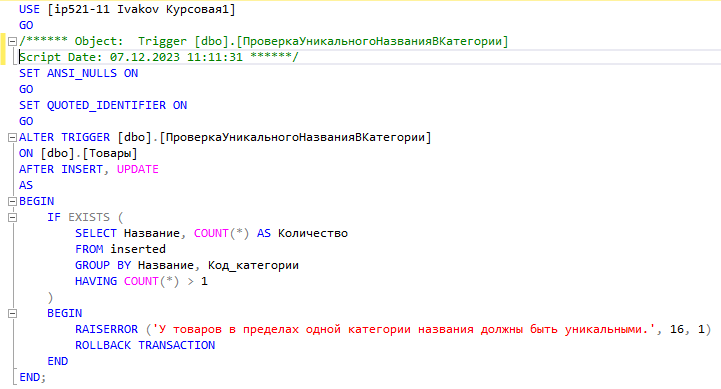


Рисунок 4.22 – Скрипт на создание триггера «Проверка уникального названия в категории»

* Проверка цены товара, проверяет правильно ли введена цена, рисунок 4.23

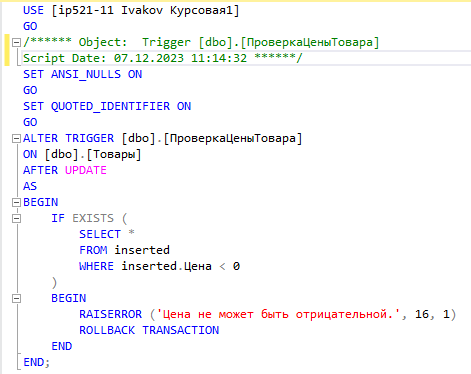


Рисунок 4.23 – Скрипт на создание триггера Update\_balance

Фрагменты создания таблиц, представлений, хранимых процедур и триггеров приведены в приложении А.

5 Реализация проекта базы данных

Для полного комфорта и удобства пользователя при работе с базой данных было создано пользовательское приложение, представляющее собой форму Windows Forms, созданную в среде разработки Microsoft Visual Studio 2019.

Подключение к базе данных осуществляется с помощью технологии ADO.NET Entity Framework, то есть объектно-ориентированной технологией доступа к данным, являясь object-relational mapping решением для .NET Framework от Microsoft.

При запуске приложения отрывается окно входа рисунок 5.1, в котором предлагается вести свой логин и пароль. Если вводится неправильный пароль, появляется окно с оповещением рисунок 5.2.

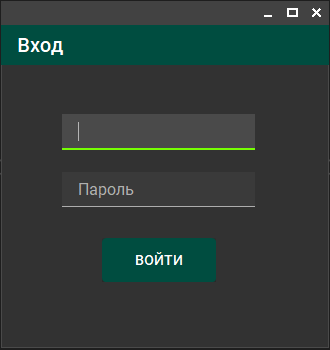


Рисунок 5.1 – Окно авторизации

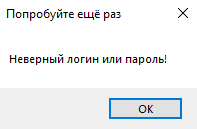


Рисунок 5.2 – Сообщение об ошибке

Если пароль указан верно, то происходит переход на следующую форму рисунок 5.3.

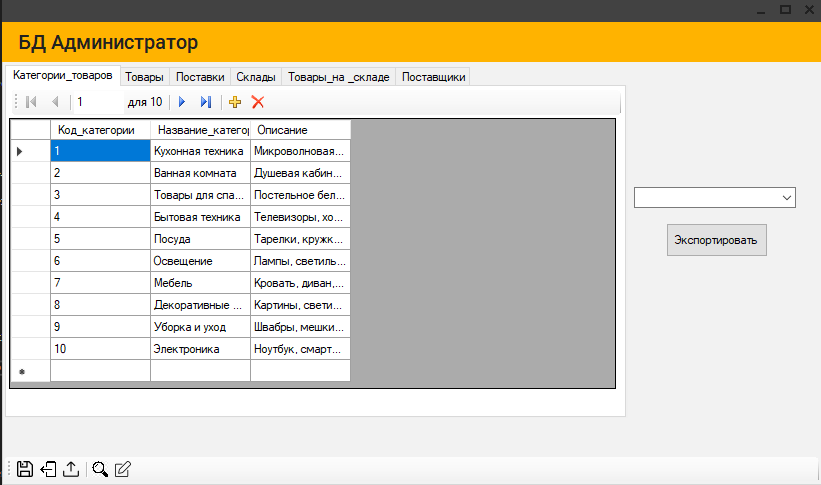


Рисунок 5.3 – Окно работы с таблицами

Переход по существующим таблицам происходит через элемент «tabControl» с помощью кнопок рисунок 5.4, рисунок 5.5, рисунок 5.6, рисунок 5.7, рисунок 5.8, рисунок 5.9, рисунок 5.10.

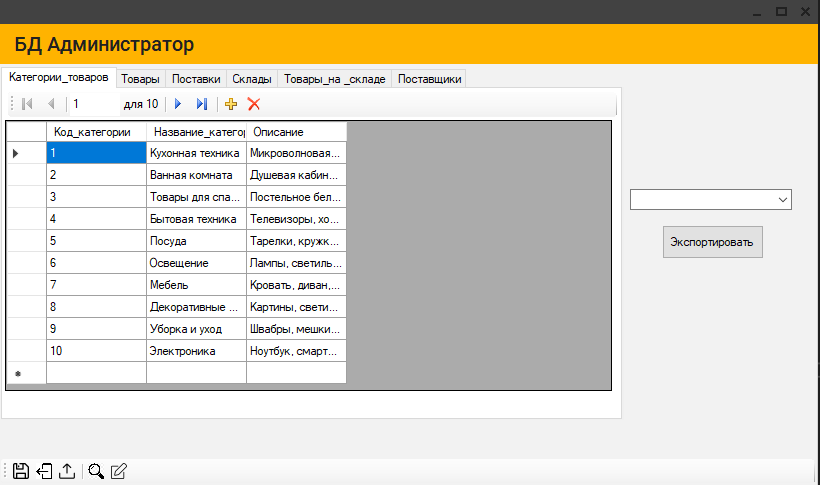


Рисунок 5.4 – Таблица «Категории товаров»

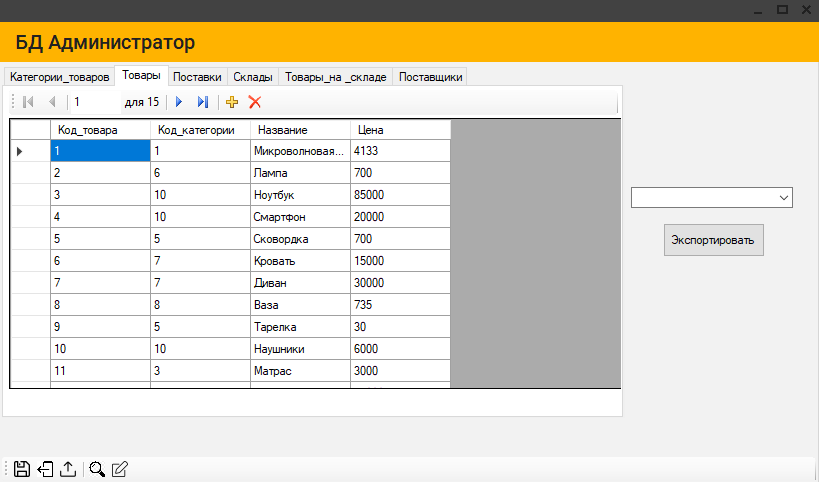


Рисунок 5.5 – Таблица «Товары»

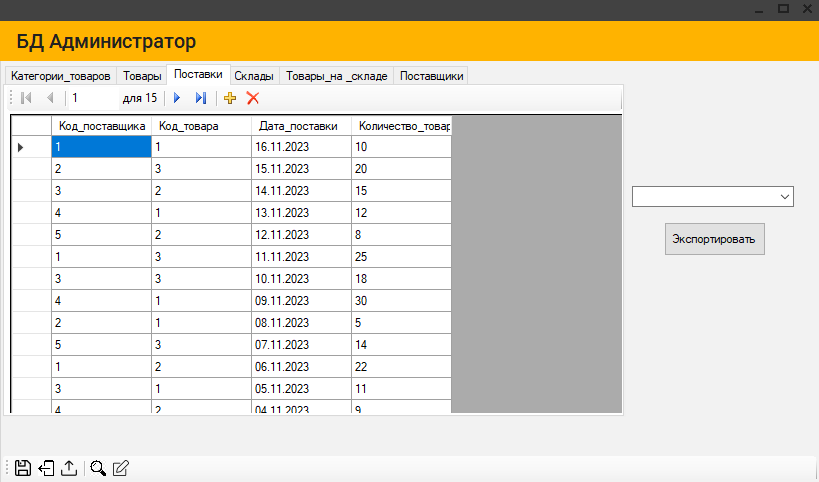


Рисунок 5.6 – Таблица «Поставки»

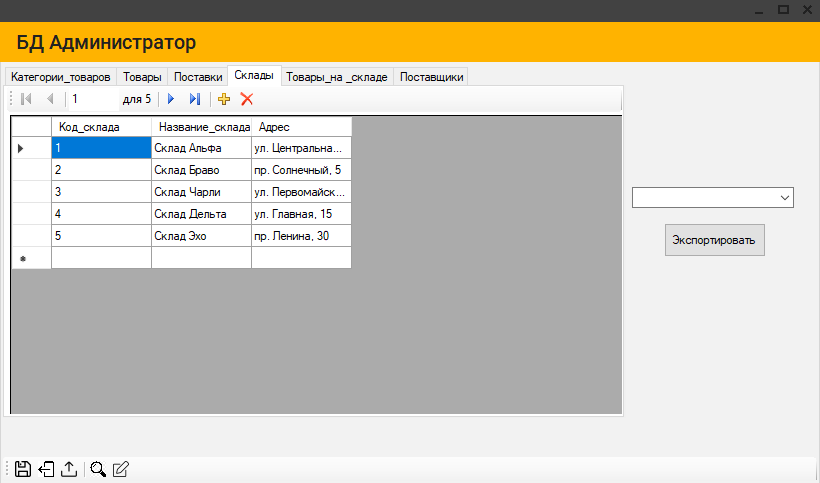


Рисунок 5.7 – Таблица «Склады»

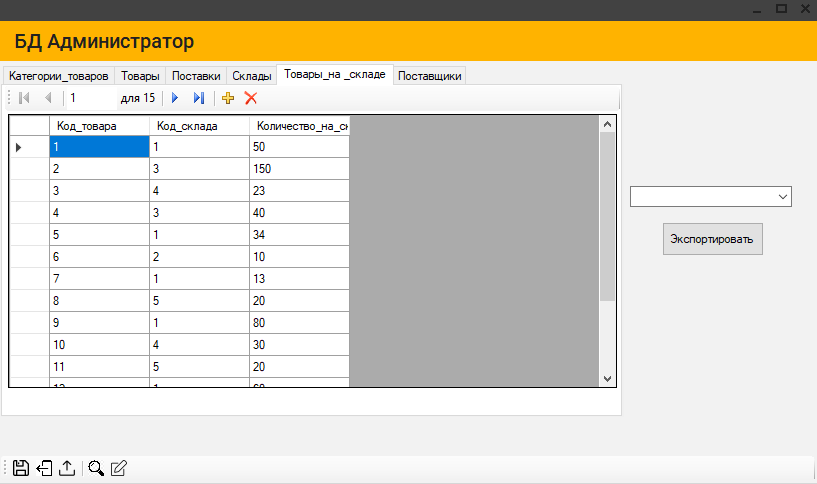


Рисунок 5.8 – Таблица «Товары на складе»

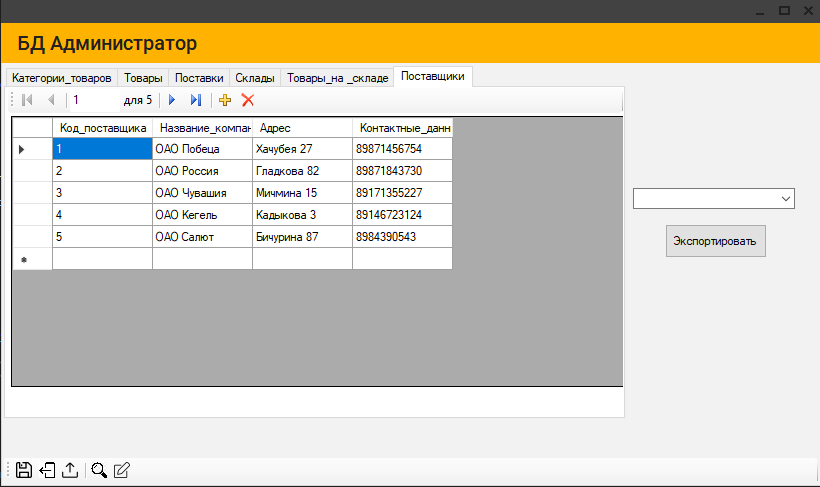


Рисунок 5.9 – Таблица «Поставщики»

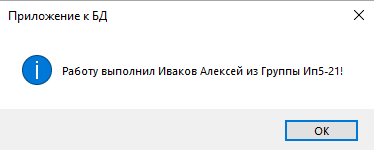


Рисунок 5.10 – Сведения об разработчике

Кнопки «Сохранить» и «Выход» вызывают исполнение соответствующих команд. При нажатии на первую кнопку сохраняются изменения базы данных. При нажатии на вторую кнопку выходит из второй формы и переходит в первую

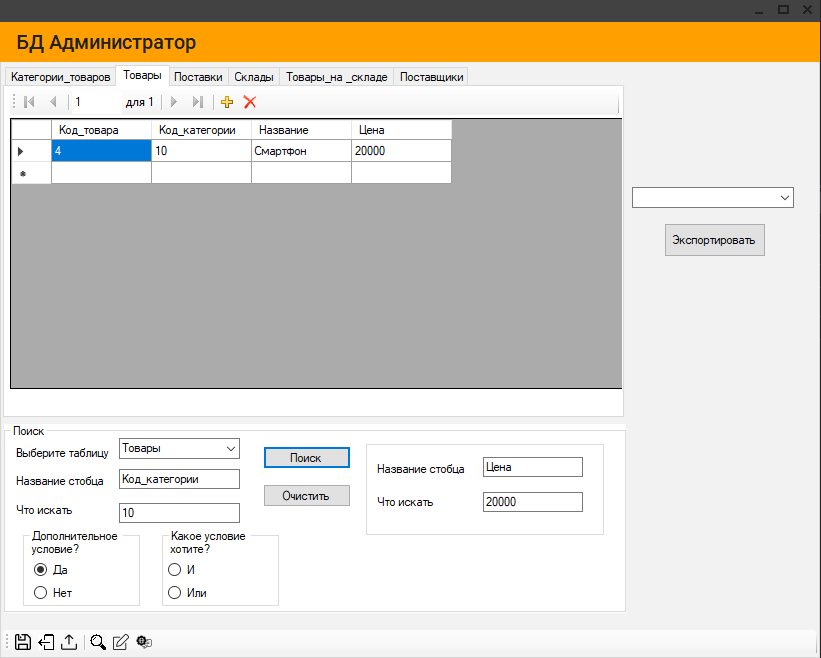


Рисунок 5.11 – Фильтрация

Кнопка «Экспортировать на форме «MONETARY\_TRANSACTION» создает отчет в Microsoft Excel рисунок 5.15

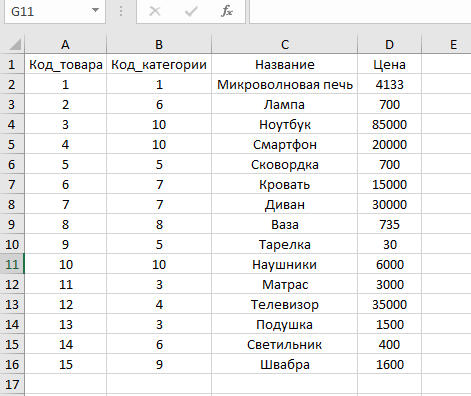


Рисунок 5.12 – Отчет в Microsoft Excel

По итогу результатом пройденного пути стало создание приложения, полностью удовлетворяющим все требования. Данное приложение позволяет свободно работать с созданной ранее БД.

ЗАКЛЮЧЕНИЕ

В ходе разработки курсового проекта были решены все поставленные задачи:

* были произведены анализ и описание предметной области;
* разработаны концептуальная и логическая модель базы данных;
* спроектирована база данных с использованием средств SQL;
* разработан пользовательский интерфейс.

А также получены новые навыки по работе с такими программами как: Oracle SQL Developer Data Modeler, Microsoft Visual Studio 2019 и СУБД Microsoft SQL Server.

Итоговое приложение полностью удовлетворяет запросы, так как позволяет свободно вносить правки в базу данных, создавать информацию о новых товарах, поставках, поставщиков и складов. Программный продукт способен создавать итоговые отчеты и обрабатывать большое количество информации.

После решения всех поставленных задач была достигнута цель курсового проекта – разработана база данных складского учета магазина бытовых товаров и ее защита.

Можно и дальше улучшать приложение, например, создание следующий версии исходя из приобретенного опыта.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Федорова, Г. Н. Разработка, администрирование и защита баз данных: учебник для студ. учреждений сред. проф. Образования / Федорова Г. Н. – Москва: Издательский центр «Академия», 2019. – 288 с.
2. Агальцов, В. П. Базы данных: в 2 книгах. Книга 2. Распределенные и удаленные базы данных: учебник / В.П. Агальцов. — Москва: ФОРУМ: ИНФРА- М, 2021. — 271 с. — URL: https://znanium.com/catalog/product/1514118 (дата обращения: 04.12.2022).
3. Мартишин, С. А. Базы данных. Практическое применение СУБД SQL и NoSQL-типа для проектирования информационных систем: учебное пособие / С. А. Мартишин, В. Л. Симонов, М. В. Храпченко. — Москва: ФОРУМ: ИНФРА-М, 2021. — 368 с. — URL: https://znanium.com/catalog/product/1215513 (дата обращения: 04.12.2022).

## ПРИЛОЖЕНИЕ А

Логическая модель

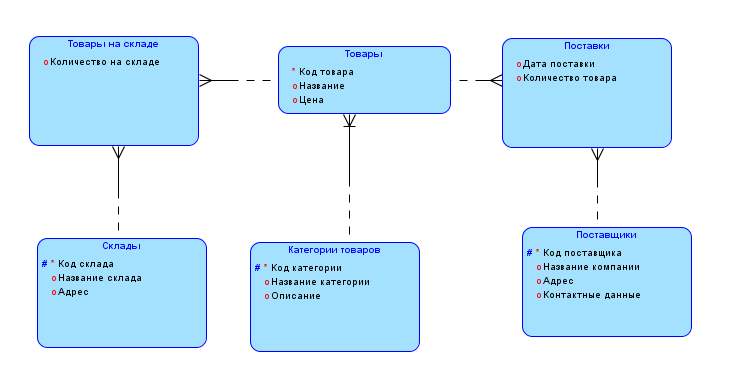


Рисунок А.1 – Логическая модель

# ПРИЛОЖЕНИЕ Б

Реалиционная модель

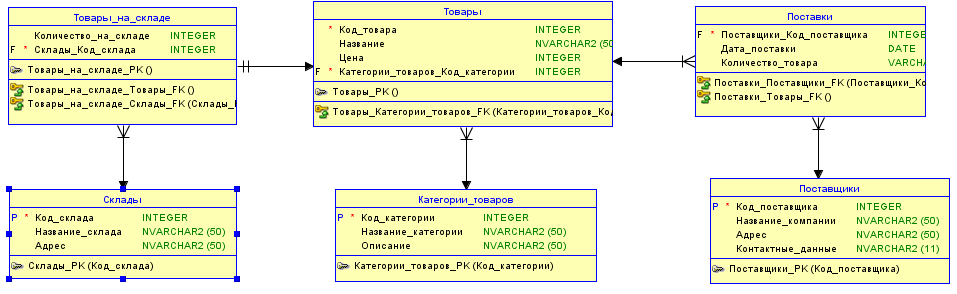


Рисунок Б.1 – Логическая модель

ПРИЛОЖЕНИЕ B

Код создания таблиц

CREATE TABLE Категории\_товаров (

Код\_категории int NOT NULL primary key identity,

Название\_категории varchar(50),

Описание varchar(300)

);

CREATE TABLE Поставщики (

Код\_поставщика INTEGER NOT NULL primary key identity,

Название\_компании varchar(50),

Адрес varchar(50),

Контактные\_данные varchar(15)

);

CREATE TABLE Склады (

Код\_склада INTEGER NOT NULL primary key identity,

Название\_склада varchar(50),

Адрес varchar(50)

);

CREATE TABLE Товары (

Код\_товара INTEGER NOT NULL primary key identity,

Код\_категории INTEGER NOT NULL,

foreign key (Код\_категории) references Категории\_товаров(Код\_категории),

Название varchar(50),

Цена INTEGER

);

CREATE TABLE Поставки (

Код\_поставщика int NOT NULL,

foreign key (Код\_поставщика) references Поставщики(Код\_поставщика),

Код\_товара int NOT NULL,

foreign key (Код\_товара) references Товары(Код\_товара),

Дата\_поставки date,

Количество\_товара varchar(50)

);

CREATE TABLE Товары\_на\_складе (

Код\_товара INTEGER NOT NULL primary key identity,

Код\_склада INTEGER NOT NULL,

foreign key (Код\_склада) references Склады(Код\_склада),

Количество\_на\_складе INTEGER

);

INSERT INTO Категории\_товаров (Название\_категории, Описание)

VALUES

('Кухонная техника', 'Микроволновая печь, тостер, кофеварка, блендер.'),

('Ванная комната', 'Душевая кабина, раковина, зеркало'),

('Товары для спальни ', 'Постельное белье, подушки, одеяла, матрасы'),

('Бытовая техника', 'Телевизоры, холодильники, стиральные машины'),

('Посуда', 'Тарелки, кружки, сковородки, столовые приборы'),

('Освещение', 'Лампы, светильники'),

('Мебель', 'Кровать, диван, стол, стул'),

('Декоративные предметы и аксессуары', 'Картины, светильники, вазы, часы, зеркала, подушки'),

('Уборка и уход', 'Швабры, мешки для мусора, моющие средства'),

('Электроника', 'Ноутбук, смартфон, планшет, наушники')

;

INSERT INTO Поставщики (Название\_компании, Адрес, Контактные\_данные)

VALUES

('ОАО Побеца', 'Хачубея 27', '89871456732'),

('ОАО Россия', 'Гладкова 82', '89871843730'),

('ОАО Чувашия', 'Мичмина 15', '89171355227'),

('ОАО Кегель', 'Кадыкова 3', '89146723124'),

('ОАО Салют', 'Бичурина 87', '8984390543');

insert into Склады(Название\_склада, Адрес)

values

('Склад Альфа','ул. Центральная, 10'),

('Склад Браво', 'пр. Солнечный, 5'),

('Склад Чарли', 'ул. Первомайская, 20');

INSERT INTO Товары (Код\_категории,Название, Цена)

VALUES

(1,'Микроволновая печь', 4133),

(6,'Лампа', 700),

(10,'Ноутбук', 85000),

(10,'Смартфон', 20000),

(5,'Сковордка', 700),

(7,'Кровать', 15000),

(7,'Диван', 30000),

(8,'Ваза', 735),

(5,'Тарелка', 30),

(10,'Наушники', 6000),

(3,'Матрас', 3000),

(4,'Телевизор', 35000),

(3,'Подушка', 1500),

(6,'Светильник', 400),

(9,'Швабра', 1600);

INSERT INTO Склады (Название\_склада, Адрес)

VALUES

('Склад Дельта', 'ул. Главная, 15'),

('Склад Эхо', 'пр. Ленина, 30');

insert into Товары\_на\_складе(Код\_склада, Количество\_на\_складе)

values

(1, 50),

(2, 150),

(1, 23),

(1, 40),

(2, 34),

(1, 10),

(1, 13),

(1, 20),

(1, 80),

(1, 30),

(1, 20),

(1, 60),

(1, 50),

(1, 47),

(1, 23);

insert into Поставки(Код\_поставщика,Код\_товара,Дата\_поставки,Количество\_товара)

values

(1, 1, '2023-11-16', 10),

(2, 3, '2023-11-15', 20),

(3, 2, '2023-11-14', 15),

(4, 1, '2023-11-13', 12),

(5, 2, '2023-11-12', 8),

(1, 3, '2023-11-11', 25),

(3, 3, '2023-11-10', 18),

(4, 1, '2023-11-09', 30),

(2, 1, '2023-11-08', 5),

(5, 3, '2023-11-07', 14),

(1, 2, '2023-11-06', 22),

(3, 1, '2023-11-05', 11),

(4, 2, '2023-11-04', 9),

(5, 3, '2023-11-03', 21),

(3, 3, '2023-11-02', 17);

delete from Товары\_на\_складе

where Код\_товара between 16 and 30;

Код создания представлений:

CREATE VIEW Средняя\_цена\_товаров\_по\_категориям AS

SELECT

Категории\_товаров.Название\_категории,

AVG(Товары.Цена) AS Средняя\_цена

FROM Товары

JOIN Категории\_товаров ON Товары.Код\_категории = Категории\_товаров.Код\_категории

GROUP BY Категории\_товаров.Название\_категории;

CREATE VIEW Количество\_товаров\_по\_поставщикам AS

SELECT

Поставщики.Название\_компании,

COUNT(Поставки.Код\_товара) AS Количество\_поставок

FROM Поставки

JOIN Поставщики ON Поставки.Код\_поставщика = Поставщики.Код\_поставщика

GROUP BY Поставщики.Название\_компании;

CREATE VIEW Суммарная\_стоимость\_поставок\_по\_дате AS

SELECT

Дата\_поставки,

SUM(Поставки.Количество\_товара \* Товары.Цена) AS Суммарная\_стоимость

FROM Поставки

JOIN Товары ON Поставки.Код\_товара = Товары.Код\_товара

GROUP BY Дата\_поставки;

CREATE VIEW Товары\_самый\_дорогой\_и\_дешевый AS

SELECT

Категории\_товаров.Название\_категории,

MAX(Товары.Цена) AS Самый\_дорогой\_товар,

MIN(Товары.Цена) AS Самый\_дешевый\_товар

FROM Товары

JOIN Категории\_товаров ON Товары.Код\_категории = Категории\_товаров.Код\_категории

GROUP BY Категории\_товаров.Название\_категории;

CREATE VIEW Количество\_товаров\_на\_складе AS

SELECT

Склады.Название\_склада,

COUNT(\*) AS Общее\_количество\_товаров

FROM Товары\_на\_складе

JOIN Склады ON Товары\_на\_складе.Код\_склада = Склады.Код\_склада

GROUP BY Склады.Название\_склада;

CREATE VIEW Стоимость\_товаров\_на\_складе AS

SELECT

Склады.Название\_склада,

SUM(Товары.Цена \* Товары\_на\_складе.Количество\_на\_складе) AS Общая\_стоимость

FROM Товары\_на\_складе

JOIN Склады ON Товары\_на\_складе.Код\_склада = Склады.Код\_склада

JOIN Товары ON Товары\_на\_складе.Код\_товара = Товары.Код\_товара

GROUP BY Склады.Название\_склада;

CREATE VIEW Просмотр\_ОбщегоКоличестваНаСкладах AS

SELECT Склады.Название\_склада, SUM(Товары\_на\_складе.Количество\_на\_складе) AS Общее\_количество

FROM Склады

LEFT JOIN Товары\_на\_складе ON Склады.Код\_склада = Товары\_на\_складе.Код\_склада

GROUP BY Склады.Название\_склада;

CREATE VIEW Просмотр\_ОбщейСуммыТоваровПоКатегориям AS

SELECT Категории\_товаров.Название\_категории, SUM(Товары.Цена) AS Общая\_сумма\_товаров

FROM Категории\_товаров

inner JOIN Товары ON Категории\_товаров.Код\_категории = Товары.Код\_категории

GROUP BY Категории\_товаров.Название\_категории;

CREATE VIEW Товар\_самый\_дешевый AS

SELECT

Категории\_товаров.Название\_категории,

MIN(Товары.Цена) AS Самый\_дешевый\_товар

FROM Товары

JOIN Категории\_товаров ON Товары.Код\_категории = Категории\_товаров.Код\_категории

GROUP BY Категории\_товаров.Название\_категории;

CREATE VIEW Доступные\_товары\_на\_складе AS

SELECT \*

FROM Товары\_на\_складе

WHERE Количество\_на\_складе > 0;

Код создания хранимых процедур:

USE [ip521-11 Ivakov Курсовая1]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[НайтиТоварПоНазванию] Script Date: 07.12.2023 11:35:49 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[НайтиТоварПоНазванию]

@Название VARCHAR(50)

AS

BEGIN

SELECT \* FROM Товары

WHERE Название LIKE '%' + @Название + '%'

END;

USE [ip521-11 Ivakov Курсовая1]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ОбновитьТовар] Script Date: 07.12.2023 11:36:25 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[ОбновитьТовар]

@Код\_товара INTEGER,

@Код\_категории INTEGER,

@Название VARCHAR(50),

@Цена INTEGER

AS

BEGIN

UPDATE Товары

SET Код\_категории = @Код\_категории, Название = @Название, Цена = @Цена

WHERE Код\_товара = @Код\_товара

END;

USE [ip521-11 Ivakov Курсовая1]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ПолучитьВсеТовары] Script Date: 07.12.2023 11:36:48 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[ПолучитьВсеТовары]

AS

BEGIN

SELECT \* FROM Товары

END;

USE [ip521-11 Ivakov Курсовая1]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ПолучитьТоварПоИдентификатору] Script Date: 07.12.2023 11:37:01 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[ПолучитьТоварПоИдентификатору]

@Код\_товара INTEGER

AS

BEGIN

SELECT \* FROM Товары

WHERE Код\_товара = @Код\_товара

END;

USE [ip521-11 Ivakov Курсовая1]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[ПолучитьТоварыПоКатегории] Script Date: 07.12.2023 11:37:16 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[ПолучитьТоварыПоКатегории]

@Код\_категории INTEGER

AS

BEGIN

SELECT \* FROM Товары

WHERE Код\_категории = @Код\_категории

END;

USE [ip521-11 Ivakov Курсовая1]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[СоздатьТовар] Script Date: 07.12.2023 11:37:27 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[СоздатьТовар]

@Код\_категории INTEGER,

@Название VARCHAR(50),

@Цена INTEGER

AS

BEGIN

INSERT INTO Товары (Код\_категории, Название, Цена)

VALUES (@Код\_категории, @Название, @Цена)

END;

USE [ip521-11 Ivakov Курсовая1]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[УдалитьТовар] Script Date: 07.12.2023 11:37:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[УдалитьТовар]

@Код\_товара INTEGER

AS

BEGIN

DELETE FROM Товары

WHERE Код\_товара = @Код\_товара

END;

Код создания триггеров:

CREATE TRIGGER ПроверкаЦеныТовара

ON Товары

AFTER UPDATE

AS

BEGIN

IF EXISTS (

SELECT \*

FROM inserted

WHERE inserted.Цена < 0

)

BEGIN

RAISERROR ('Цена не может быть отрицательной.', 16, 1)

ROLLBACK TRANSACTION

END

END;

CREATE TRIGGER ПредотвращениеУдаленияКатегории

ON Категории\_товаров

INSTEAD OF DELETE

AS

BEGIN

IF EXISTS (SELECT \* FROM Товары WHERE Товары.Код\_категории IN (SELECT deleted.Код\_категории FROM deleted))

BEGIN

RAISERROR ('Нельзя удалить категорию, в которой есть товары.', 16, 1)

ROLLBACK TRANSACTION

END

ELSE

BEGIN

DELETE FROM Категории\_товаров WHERE Код\_категории IN (SELECT deleted.Код\_категории FROM deleted)

END

END;

CREATE TRIGGER ПроверкаУникальногоНазванияВКатегории

ON Товары

AFTER INSERT, UPDATE

AS

BEGIN

IF EXISTS (

SELECT Название, COUNT(\*) AS Количество

FROM inserted

GROUP BY Название, Код\_категории

HAVING COUNT(\*) > 1

)

BEGIN

RAISERROR ('У товаров в пределах одной категории названия должны быть уникальными.', 16, 1)

ROLLBACK TRANSACTION

END

END;

CREATE TRIGGER ОбновлениеКоличестваНаСкладеПриУдаленииТовара

ON Товары

AFTER DELETE

AS

BEGIN

UPDATE Товары\_на\_складе

SET Количество\_на\_складе = Количество\_на\_складе - (SELECT Количество\_на\_складе FROM deleted WHERE Товары\_на\_складе.Код\_товара = deleted.Код\_товара)

FROM Товары\_на\_складе

INNER JOIN deleted ON Товары\_на\_складе.Код\_товара = deleted.Код\_товара

END;

CREATE TRIGGER триггер\_ОбновлениеЦеныТовара

ON Товары

AFTER UPDATE

AS

BEGIN

UPDATE Товары

SET Цена = i.Цена

FROM Товары t

JOIN inserted i ON t.Код\_товара = i.Код\_товара;

END;

ПРИЛОЖЕНИЕ Г

Код создания приложения.

Код Form1:

using MaterialSkin.Controls;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MaterialSkin;

namespace Курсовая

{

public partial class Form1 : MaterialForm

{

static public string Who;

public Form1()

{

InitializeComponent();

// Create a material theme manager and add the form to manage (this)

MaterialSkinManager materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.AddFormToManage(this);

materialSkinManager.Theme = MaterialSkinManager.Themes.DARK;

materialSkinManager.ColorScheme = new ColorScheme

(

Primary.Teal900,

Primary.Grey800,

Primary.Amber900,

Accent.LightGreen400,

TextShade.WHITE

);

}

private void materialFlatButton1\_Click(object sender, EventArgs e)

{

}

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

{

}

private void materialButton1\_Click(object sender, EventArgs e)

{

if (materialTextBox21.Text == "employer" & materialTextBox22.Text == "employer")

{

Who = "Работник";

Form2 a = new Form2();

a.Show();

Hide();

}

else if (materialTextBox21.Text == "admin" & materialTextBox22.Text == "admin")

{

Who = "Администратор";

Form2 a = new Form2();

a.Show();

Hide();

}

else MessageBox.Show("Неверный логин или пароль!", "Попробуйте ещё раз");

}

private void materialTextBox21\_Click(object sender, EventArgs e)

{

}

}

}

Код Form2:

using MaterialSkin.Controls;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MaterialSkin;

using System.IO;

using OfficeOpenXml;

using OfficeOpenXml.Style;

namespace Курсовая

{

public partial class Form2 : MaterialForm

{

bool Поиск = false;

bool редактировать = false;

public Form2()

{

InitializeComponent();

MaterialSkinManager materialSkinManager = MaterialSkinManager.Instance;

materialSkinManager.Theme = MaterialSkinManager.Themes.LIGHT;

materialSkinManager.ColorScheme = new ColorScheme

(

Primary.Amber600,

Primary.Grey800,

Primary.Amber900,

Accent.LightGreen400,

TextShade.BLACK

);

}

private void Form2\_Load(object sender, EventArgs e)

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "\_ip521\_11\_Ivakov\_Курсовая1DataSet.Поставщики". При необходимости она может быть перемещена или удалена.

this.поставщикиTableAdapter.Fill(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Поставщики);

// TODO: данная строка кода позволяет загрузить данные в таблицу "\_ip521\_11\_Ivakov\_Курсовая1DataSet.Товары\_на\_складе". При необходимости она может быть перемещена или удалена.

this.товары\_на\_складеTableAdapter.Fill(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Товары\_на\_складе);

// TODO: данная строка кода позволяет загрузить данные в таблицу "\_ip521\_11\_Ivakov\_Курсовая1DataSet.Склады". При необходимости она может быть перемещена или удалена.

this.складыTableAdapter1.Fill(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Склады);

// TODO: данная строка кода позволяет загрузить данные в таблицу "\_ip521\_11\_Ivakov\_Курсовая1DataSet.Поставки". При необходимости она может быть перемещена или удалена.

this.поставкиTableAdapter.Fill(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Поставки);

// TODO: данная строка кода позволяет загрузить данные в таблицу "\_ip521\_11\_Ivakov\_Курсовая1DataSet.Товары". При необходимости она может быть перемещена или удалена.

this.товарыTableAdapter.Fill(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Товары);

// TODO: данная строка кода позволяет загрузить данные в таблицу "\_ip521\_11\_Ivakov\_Курсовая1DataSet.Категории\_товаров". При необходимости она может быть перемещена или удалена.

this.категории\_товаровTableAdapter.Fill(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Категории\_товаров);

Size = new Size(820, 610);

groupBox3.Visible = false;

if (Form1.Who == "Работник")

{

this.Text = "БД Работник";

tabControl2.Visible = true;

dataGridView6.Visible = false;

dataGridView7.Visible = false;

toolStripButton5.Visible = false;

}

else

if (Form1.Who == "Администратор")

{

//tabControl1.Visible = true;

//tabControl2.Visible = false;

this.Text = "БД Администратор";

tabControl1.Visible = true;

}

}

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

{

Form1 a = new Form1();

a.Show();

Hide();

}

private void label4\_Click(object sender, EventArgs e)

{

}

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

{

//клиентыBindingSource.AddNew();

}

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

{

категориитоваровBindingSource.EndEdit();

категории\_товаровTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

товарыBindingSource.EndEdit();

товарыTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

поставкиBindingSource.EndEdit();

поставкиTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

складыBindingSource1.EndEdit();

складыTableAdapter1.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

товарынаскладеBindingSource.EndEdit();

товары\_на\_складеTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

поставщикиBindingSource.EndEdit();

поставщикиTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

this.Validate();

this.категориитоваровBindingSource.EndEdit();

this.категории\_товаровTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Категории\_товаров);

this.Validate();

this.товарыBindingSource.EndEdit();

this.товарыTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Товары);

this.Validate();

this.поставкиBindingSource.EndEdit();

this.поставкиTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Поставки);

this.Validate();

this.складыBindingSource1.EndEdit();

this.складыTableAdapter1.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Склады);

this.Validate();

this.товарынаскладеBindingSource.EndEdit();

this.товары\_на\_складеTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Товары\_на\_складе);

this.Validate();

this.поставщикиBindingSource.EndEdit();

this.поставщикиTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Поставщики);

}

private void tabPage3\_Click(object sender, EventArgs e)

{

}

private void tabControl2\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void toolStripButton3\_Click(object sender, EventArgs e)

{

string columnName = textBox1.Text;

string filterValue = textBox2.Text;

string a = comboBox1.SelectedItem.ToString();

if (radioButton3.Checked == true)

{

switch (a)

{

case "Категории\_товаров": категориитоваровBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "')"; break;

case "Товары": товарыBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "')"; break;

case "Поставки": поставкиBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "')"; break;

case "Склады": складыBindingSource1.Filter = textBox1.Text + " = ('" + textBox2.Text + "')"; break;

case "Товары\_на\_складе": товарынаскладеBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "')"; break;

case "Поставщики": поставщикиBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "')"; break;

default: MessageBox.Show("Таблица не найдена", "Ошибка"); break;

}

}

else

{

if (radioButton2.Checked == false)

{

switch (a)

{

case "Категории\_товаров": категориитоваровBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') and " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Товары": товарыBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') and " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Поставки": поставкиBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') and " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Склады": складыBindingSource1.Filter = textBox1.Text + " = ('" + textBox2.Text + "') and " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Товары\_на\_складе": товарынаскладеBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') and " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Поставщики": поставщикиBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') and " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

default: MessageBox.Show("Таблица не найдена", "Ошибка"); break;

}

}

else

{

switch (a)

{

case "Категории\_товаров": категориитоваровBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') or " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Товары": товарыBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') or " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Поставки": поставкиBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') or " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Склады": складыBindingSource1.Filter = textBox1.Text + " = ('" + textBox2.Text + "') or " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Товары\_на\_складе": товарынаскладеBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') or " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

case "Поставщики": поставщикиBindingSource.Filter = textBox1.Text + " = ('" + textBox2.Text + "') or " + textBox5.Text + " =('" + textBox4.Text + "')"; break;

default: MessageBox.Show("Таблица не найдена", "Ошибка"); break;

}

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

if (radioButton3.Checked == false)

{

groupBox2.Visible = true;

groupBox5.Visible = true;

}

else groupBox2.Visible = false;

if (Поиск == true & редактировать == true)

{

Size = new Size(774, 600);

groupBox3.Visible = true;

groupBox4.Visible = true;

}

else if (Поиск == true)

{

Size = new Size(820, 660);

groupBox3.Size = new Size(622, 189);

groupBox3.Visible = true;

groupBox4.Visible = false;

}

else if (редактировать == true)

{

Size = new Size(774, 600);

groupBox3.Visible = false;

groupBox4.Visible = true;

}

else

{

Size = new Size(820, 487);

groupBox3.Visible = false;

groupBox4.Visible = false;

}

}

private void toolStripButton3\_Click\_1(object sender, EventArgs e)

{

if (Поиск == true)

{

Поиск = false;

}

else Поиск = true;

}

private void toolStripButton5\_Click(object sender, EventArgs e)

{

if (редактировать == true)

{

редактировать = false;

}

else редактировать = true;

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

}

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

{

//fd

}

private void button3\_Click\_1(object sender, EventArgs e)

{

товарыBindingSource.Filter = string.Empty;

категориитоваровBindingSource.Filter = string.Empty;

складыBindingSource1.Filter = string.Empty;

поставщикиBindingSource.Filter = string.Empty;

товарынаскладеBindingSource.Filter = string.Empty;

поставкиBindingSource.Filter = string.Empty;

textBox1.Clear();

textBox2.Clear();

}

private void toolStripButton4\_Click(object sender, EventArgs e)

{

string filepath = "";

string table = comboBox2.Text;

switch (table)

{

case "Товары":

filepath = $"{comboBox2.Text}.xlsx";

break;

case "Категории\_товаров":

filepath = $"{comboBox2.Text}.xlsx";

break;

case "Склады":

filepath = $"{comboBox2.Text}.xlsx";

break;

case "Поставки":

filepath = $"{comboBox2.Text}.xlsx";

break;

case "Поставщики":

filepath = $"{comboBox2.Text}.xlsx";

break;

case "Товары\_на\_складе":

filepath = $"{comboBox2.Text}.xlsx";

break;

default:

MessageBox.Show("Такой таблицы не существует!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

break;

}

}

private void ExportToExcel(DataGridView dataGridView, string filePath)

{

ExcelPackage.LicenseContext = OfficeOpenXml.LicenseContext.NonCommercial;

using (ExcelPackage excelPackage = new ExcelPackage())

{

ExcelWorksheet worksheet = excelPackage.Workbook.Worksheets.Add("Sheet1");

// Заполнение заголовков столбцов

for (int column = 0; column < dataGridView.ColumnCount; column++)

{

worksheet.Cells[1, column + 1].Value = dataGridView.Columns[column].HeaderText;

}

// Заполнение ячеек данными

for (int row = 0; row < dataGridView.Rows.Count; row++)

{

for (int column = 0; column < dataGridView.ColumnCount; column++)

{

worksheet.Cells[row + 2, column + 1].Value = dataGridView.Rows[row].Cells[column].Value;

}

}

// Применение стилей к таблице

using (ExcelRange range = worksheet.Cells[1, 1, dataGridView.Rows.Count + 1, dataGridView.ColumnCount])

{

range.Style.Font.Bold = true;

range.Style.HorizontalAlignment = ExcelHorizontalAlignment.Center;

range.AutoFitColumns();

}

// Сохранение файла Excel

FileInfo excelFile = new FileInfo(filePath);

excelPackage.SaveAs(excelFile);

}

MessageBox.Show("Таблица была успешна экспортирована", "Экспорт");

}

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

{

}

private void toolStripButton6\_Click(object sender, EventArgs e)

{

this.Validate();

this.поставщикиBindingSource.EndEdit();

this.поставщикиTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Поставщики);

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

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

{

}

private void tabPage6\_Click(object sender, EventArgs e)

{

}

private void menuStrip1\_ItemClicked(object sender, ToolStripItemClickedEventArgs e)

{

}

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

{

}

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

{

}

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

{

tabControl1.Visible = false;

}

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

{

категориитоваровBindingSource.EndEdit();

категории\_товаровTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

товарыBindingSource.EndEdit();

товарыTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

поставкиBindingSource.EndEdit();

поставкиTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

складыBindingSource1.EndEdit();

складыTableAdapter1.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

товарынаскладеBindingSource.EndEdit();

товары\_на\_складеTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

поставщикиBindingSource.EndEdit();

поставщикиTableAdapter.Update(\_ip521\_11\_Ivakov\_Курсовая1DataSet);

this.Validate();

this.категориитоваровBindingSource.EndEdit();

this.категории\_товаровTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Категории\_товаров);

this.Validate();

this.товарыBindingSource.EndEdit();

this.товарыTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Товары);

this.Validate();

this.поставкиBindingSource.EndEdit();

this.поставкиTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Поставки);

this.Validate();

this.складыBindingSource1.EndEdit();

this.складыTableAdapter1.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Склады);

this.Validate();

this.товарынаскладеBindingSource.EndEdit();

this.товары\_на\_складеTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Товары\_на\_складе);

this.Validate();

this.поставщикиBindingSource.EndEdit();

this.поставщикиTableAdapter.Update(this.\_ip521\_11\_Ivakov\_Курсовая1DataSet.Поставщики);

}

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

{

Application.Exit();

}

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

{

}

private void groupBox3\_Enter(object sender, EventArgs e)

{

}

private void label4\_Click\_1(object sender, EventArgs e)

{

}

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

{

tabControl1.TabPages[5].Visible = true;

}

private void radioButton4\_CheckedChanged(object sender, EventArgs e)

{

groupBox2.Visible = true;

}

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

{

}

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

{

}

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

{

}

private void button4\_Click\_1(object sender, EventArgs e)

{

string selectedState = comboBox2.SelectedItem.ToString();

switch (selectedState)

{

case "Категории\_товаров": ExportToExcel(dataGridView1, "BD.xlsx"); break;

case "Товары": ExportToExcel(dataGridView2, "BD.xlsx"); break;

case "Поставки": ExportToExcel(dataGridView4, "BD.xlsx"); break;

case "Склады": ExportToExcel(dataGridView5, "BD.xlsx"); break;

case "Товары\_на\_складе": ExportToExcel(dataGridView6, "BD.xlsx"); break;

case "Поставщики": ExportToExcel(dataGridView7, "BD.xlsx"); break;

default: MessageBox.Show("Таблица не найдена", "Ошибка"); break;

}

}

private void radioButton3\_CheckedChanged(object sender, EventArgs e)

{

groupBox5.Visible = false;

}

}

}