ОГЛАВЛЕНИЕ

Оглавление

ВВЕДЕНИЕ

В настоящее время многие процессы нужно автоматизировать, поэтому создаются приложения, которые работают с базой данных. Благодаря таким приложениям работа многих сотрудникам становится проще, так как в один клик можно посмотреть данные из таблиц, добавить, редактировать или удалить данные из база данных.

База данных служит для упрощенного взиамодействия между различными данными, для их редактирования и просмотра. Чаще для работы с базами данных используют SQL. Его часто называют языком эсперанто для систем управления базами данных (СУБД), можно использовать даже при работе с нереляционными СУБД.

SQL — это специализированный непроцедурный язык, позволяющий описывать данные, осуществлять выборку и обработку информации из реляционных СУБД.

Для реализации программного продукта было решено использовать язык C#. Базу данных была реализована в MS SQL SERVER. Для удобства работы с базой данных использовалась графическая программа SQL Server Management Studio. В качестве среды разработки программного продукта использовалась Microsoft Visual Studio 2019.

Данная пояснительная записка состоит из следующих разделов:

- Проектирование базы данных. В этом разделе проектируется база данных согласно предметной области проекта.
- Проектирование формы. В этом разделе проектируются формы приложения, элементы управления, свойства, события и методы.
- Руководство пользователя. В этом разделе описывается весь процесс работы с реализованным программным продуктом.

1 Проектирование базы данных

При проектировании базы данных следует выполнить следующие этапы:

- концептуальное проектирование;
- логическое проектирование;
- физическое проектирование.

1.1. Проектирование концептуальной модели

Задачей концептуального проектирования является: выделение основных объектов предметной области, которые реализуются в виде сущностей, определение свойств сущностей и связей между ними.

Основные сущности:

- продукт;
- магазин;
- история поставок и продаж;

На рисунке 1.1. представлена модель базы данных.

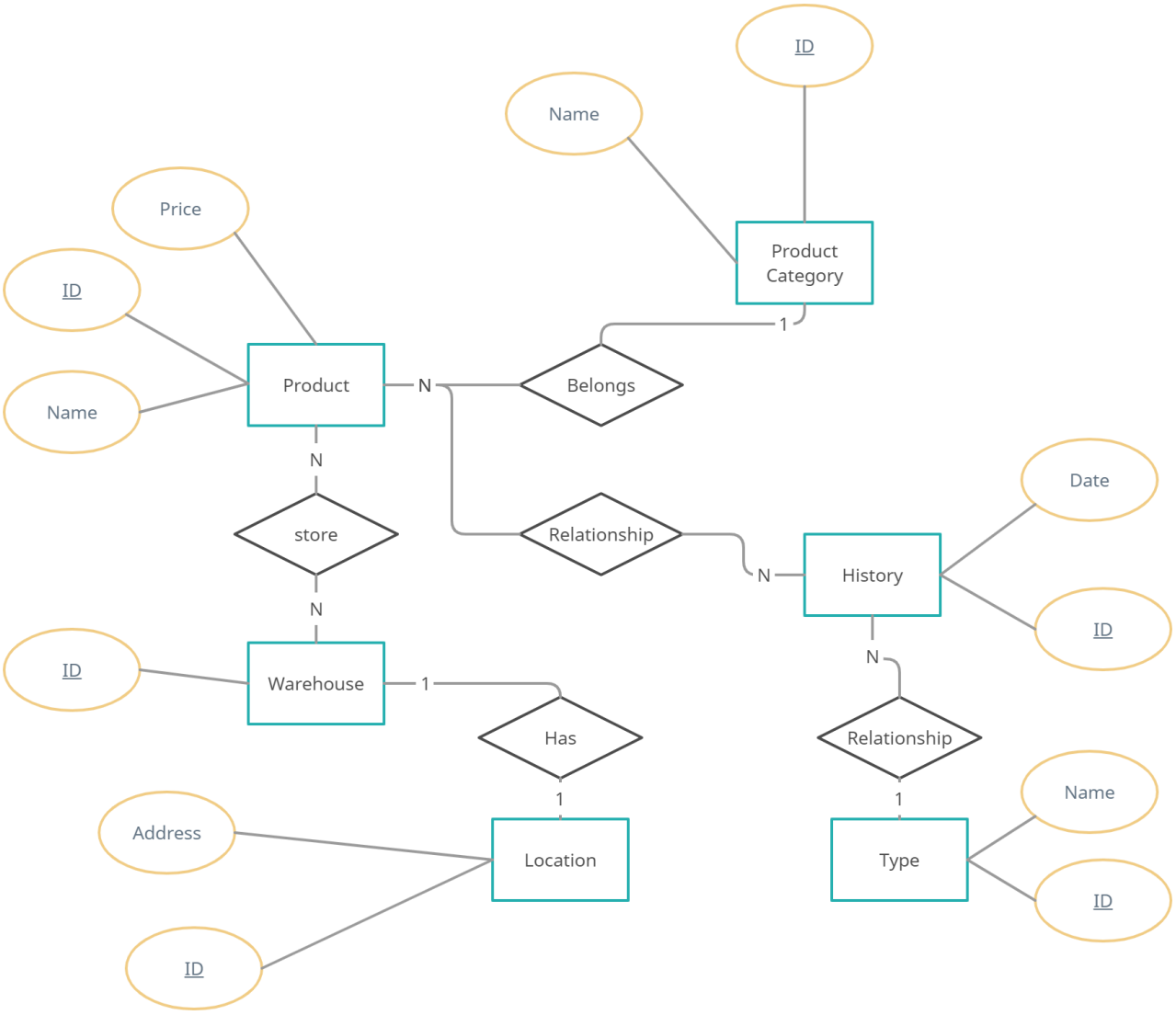

Рисунок 1.1 - концептуальная модель базы данных

1.2. Логическая модель

На логическом уровне модель предметной области представляется в привязке к СУБД определённого вида или к конкретной СУБД и описывает способ организации данных безотносительно их физического размещения.

База данных нормализовано в соответствии с 3 нормальными формами баз данных, а именно:

- все поля атомарные;
- нет частичных функциональных зависимостей;
- нет транзитивных зависимостей не ключевых атрибутов от ключа.

Логическая схема представлена на рисунке 1.2

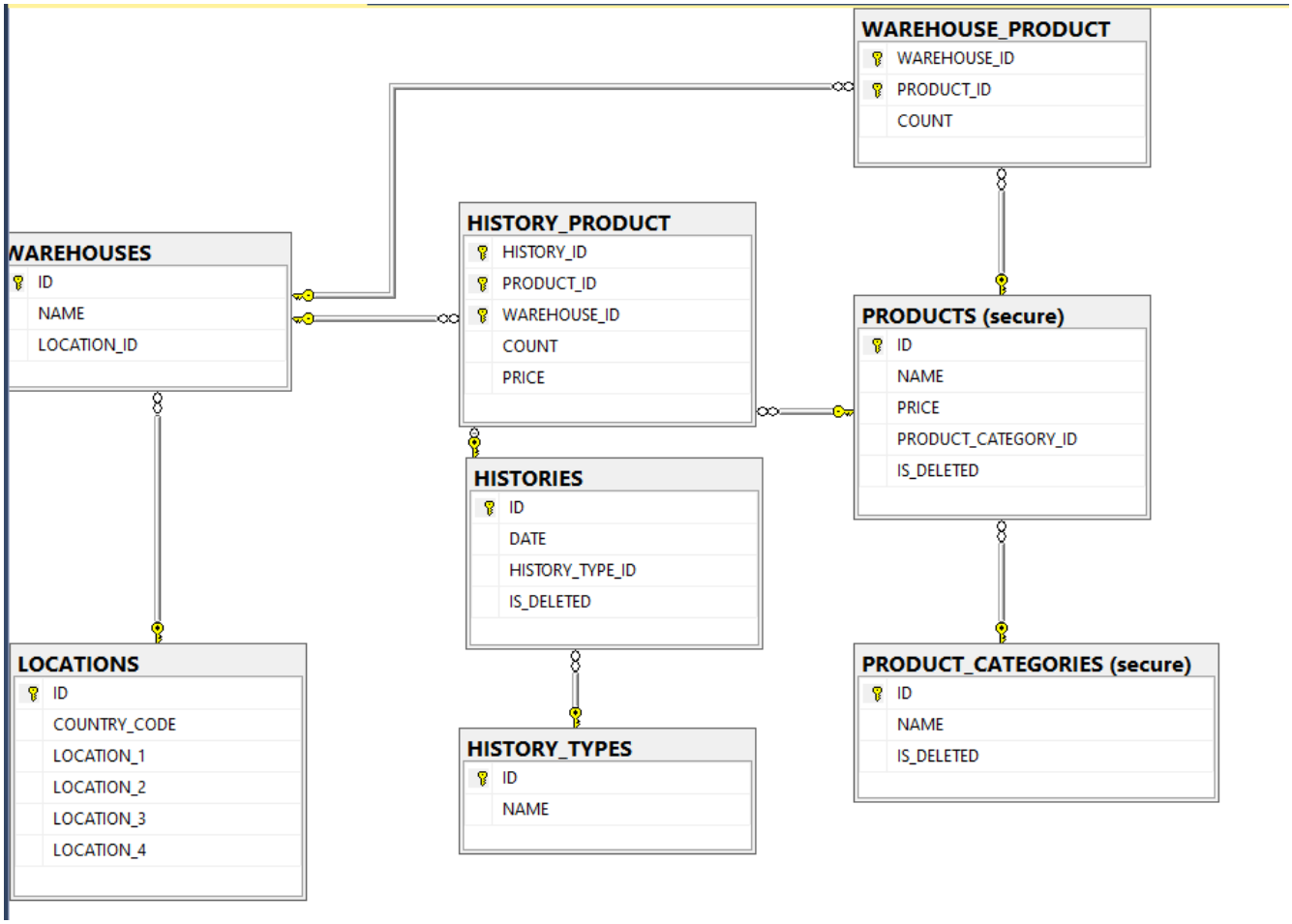

Рисунок 1.2 – логическая модель базы данных

Описание таблицы магазинов (таблица «Warehouses») представлено в таблице 1.3.1. Описание таблицы адресов (таблица «Locations») представлено в таблице 1.3.2. Описание таблицы продуктов (таблица «Product») представлено в таблице 1.3.3. Описание таблицы категории продуктов (таблица

«Product_Categories») представлено в таблице 1.3.4. Описание таблицы истории закупок и продаж (таблица «Histories») представлено в таблице 1.3.5. Описание таблицы типов истории (таблица «History_Types») представлено в таблице 1.3.6. Описание таблицы связи между историей покупок продаж и продуктами (таблица «History_Product») представлено в таблице 1.3.7. Описание таблицы связи продуктов и магазинов (таблица «Warehouse_Product») представлено в таблице 1.3.8.

Таблица 1.3.1

Поля	Тип данных	Ограничения
ID	int	PK, Identity(1,1)
NAME	nvarchar(100)	NOT NULL
LOCATION_ID	nvarchar(30)	NOT NULL FK_WAREHOUSE_LOCATION_ID

Таблица 1.3.2

Поля	Тип данных	Ограничения
ID	int	PK, Identity(1,1)
COUNTRY_CODE	nvarchar(100)	NOT NULL
LOCATION_1	nvarchar(100)	NOT NULL
LOCATION_2	nvarchar(100)	NULL
LOCATION_3	nvarchar(100)	NULL
LOCATION_4	nvarchar(100)	NULL

Таблица 1.3.3

Поля	Тип данных	Ограничения
ID	int	PK, Identity(1,1)
NAME	nvarchar(100)	NOT NULL
PRICE	money	NOT NULL, >= 0
IS_DELETED	bit	NOT NULL, default 0
PRODUCT_CATEGORY_ID	int	NOT NULL, FK_PRODUCT_PRODUCT_CATEGORY

Таблица 1.3.4

Поля	Тип данных	Ограничения
ID	int	PK, Identity(1,1)
NAME	nvarchar(100)	NOT NULL

Таблица 1.3.5

Поля	Тип данных	Ограничения
ID	int	PK, Identity(1,1)
DATE	datetime	NOT NULL, DEFAULT = GETDATE()
HISTORY_TYPE_ID	int	NOT NULL, FK_HISTORY_HISTORY_TYPE_ID
IS_DELETED	bit	NOT NULL, DEFAULT = 0

Таблица 1.3.6

Поля	Тип данных	Ограничения
ID	int	PK, Identity(1,1)
NAME	nvarchar(100)	NOT NULL

Таблица 1.3.7

Поля	Тип данных	Ограничения
HISTORY_ID	int	PK, FK_HISTORY_PRODUCT_HISTORY_ID
PRODUCT_ID	int	PK, FK_HISTORY_PRODUCT_PRODUCT_ID
WAREHOUSE_ID	int	PK FK_HISTORY_PRODUCT_WAREHOUSE_ID
COUNT	int	NOT NULL
PRICE	Money	NOT NULL

Таблица 1.3.8

Поля	Тип данных	Ограничения
WAREHOUSE_ID	int	PK, FK_WAREHOUSE_PRODUCT_WAREHOUSE_ID
PRODUCT_ID	int	PK, FK_WAREHOUSE_PRODUCT_PRODUCT_ID
COUNT	int	NOT NULL,

1.3. Физическая модель

Физическая модель данных описывает данные средствами конкретной СУБД. Физическая модель данных реализована средствами реляционной СУБД. Отношения, разработанные на стадии формирования логической модели данных, преобразуются в таблицы, атрибуты становятся столбцами таблиц, для ключевых атрибутов создаются уникальные индексы, домены преображаются в типы данных, принятые в конкретной СУБД.

- Имя базы данных – SweetShop;
- Имя журнала транзакций - SweetShop_log;
- Путь до файлов БД – «D:\New folder\MSSQL15.SQLEXPRESS\MSSQL\DATA»;
- Начальный размер 8мб, размер приращения 64мб, неограниченно.

2 Проектирование форм

2.1. Перечень необходимых форм

Для работы с базой данных необходимы следующие формы:

- главная форма со ссылками на остальные формы;
- форма администрирования;
- форма журнала транзакций;
- форма статистического анализа журнала;
- форма фильтрации журнала;
- форма создания новой поставки/продажи;
- форма создания и редактирования продуктов;
- форма создания и редактирования категорий;
- форма создания и редактирования кондитерских;
- отчет о поставках\продажах за определенный период;
- отчет о наиболее прибыльных товарах и категориях;
- отчет об определенной поставке\продаже.

2.2. Руководство пользователю

При открытии приложения появляется главная форма для работы с таблицами, имеющая несколько кнопок (рисунок 2.1):

- Администрирование
- Журнал
- Статистика

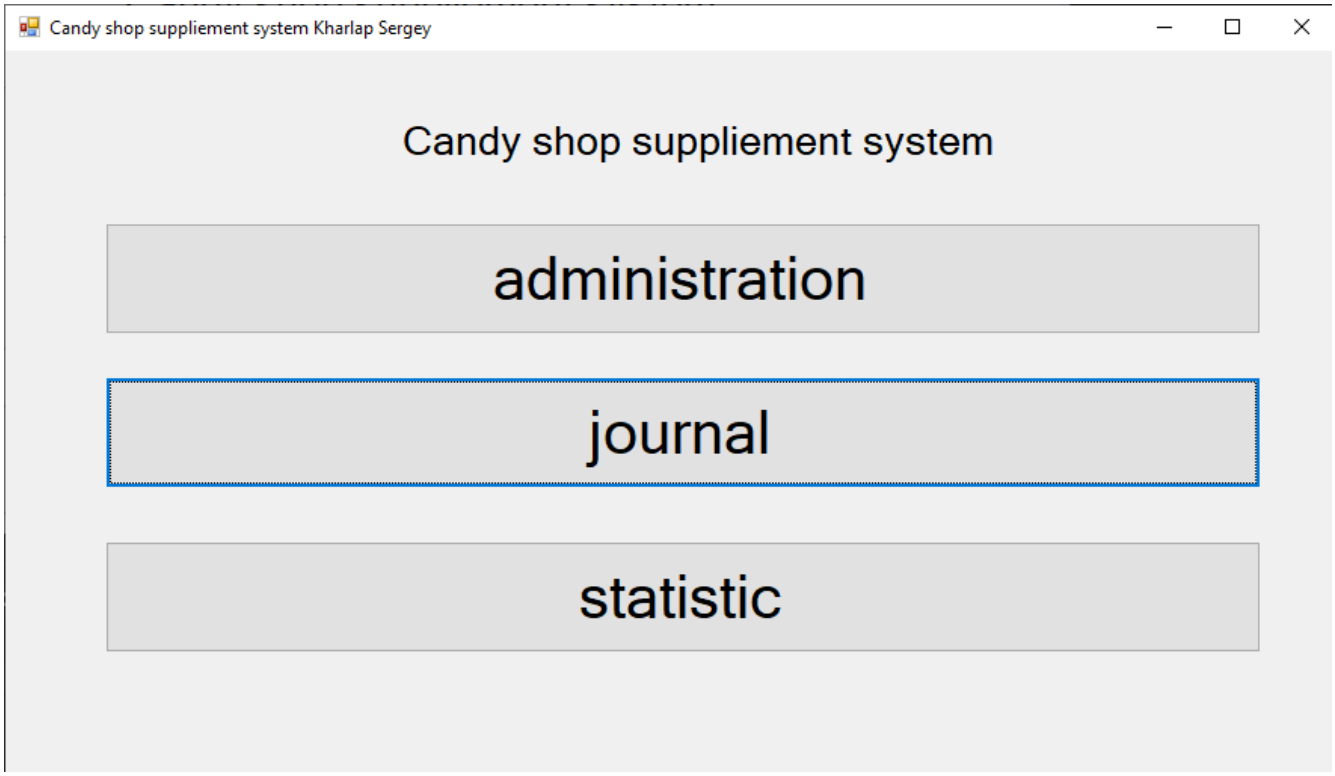

Рисунок 2.1 – главная форма

При нажатии на кнопку “Journal” откроется новая форма (рисунок 2.2). В окне расположена кнопка добавления новой поставки\продаж и журнал поставок продаж отсортированный по дате.

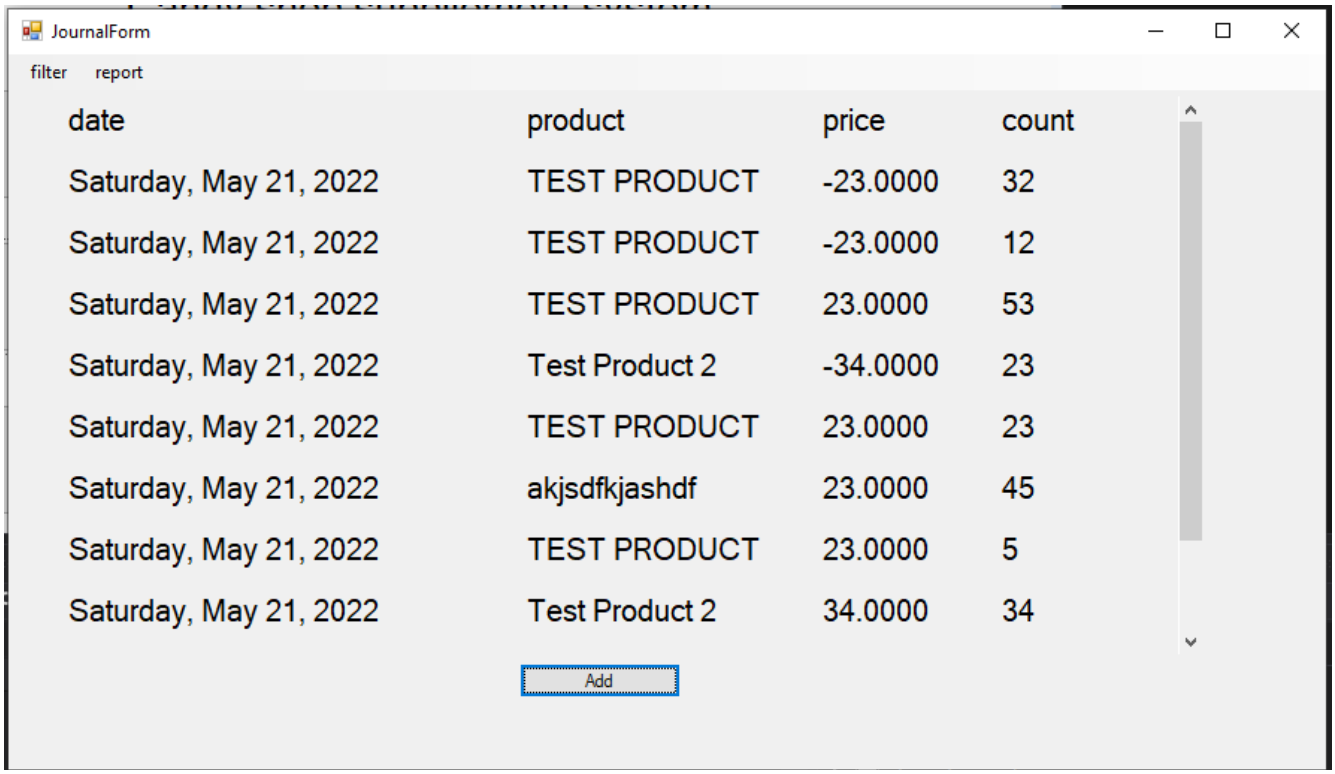

Рисунок 2.2 – форма журнала поставок\продаж

С этой формы можно открыть отчет по поставкам\продажам за период (рисунок 2.3) и форму фильтрации журнала (рисунок 2.4).

JournalReport

1 of 1 100% Find | Next

Report period

from Tuesday,
May 14, 2002

to Sunday,
May 22, 2022

Date	Product Name	Price	Count
Saturday, May 21, 2022	TEST PRODUCT	-23,0000	32
Saturday, May 21, 2022	TEST PRODUCT	-23,0000	12
Saturday, May 21, 2022	TEST PRODUCT	23,0000	53
Saturday, May 21, 2022	Test Product 2	-34,0000	23
Saturday, May 21, 2022	TEST PRODUCT	23,0000	23
Saturday, May 21, 2022	akjsdfkjashdf	23,0000	45
Saturday, May 21, 2022	TEST PRODUCT	23,0000	5
Saturday, May 21, 2022	Test Product 2	34,0000	34
Friday, May 13, 2022	TEST PRODUCT	23,0000	25
Friday, May 13, 2022	Test Product 2	34,0000	32

Total 3923.0000

Рисунок 2.3 – отчет по продажам поставкам за период

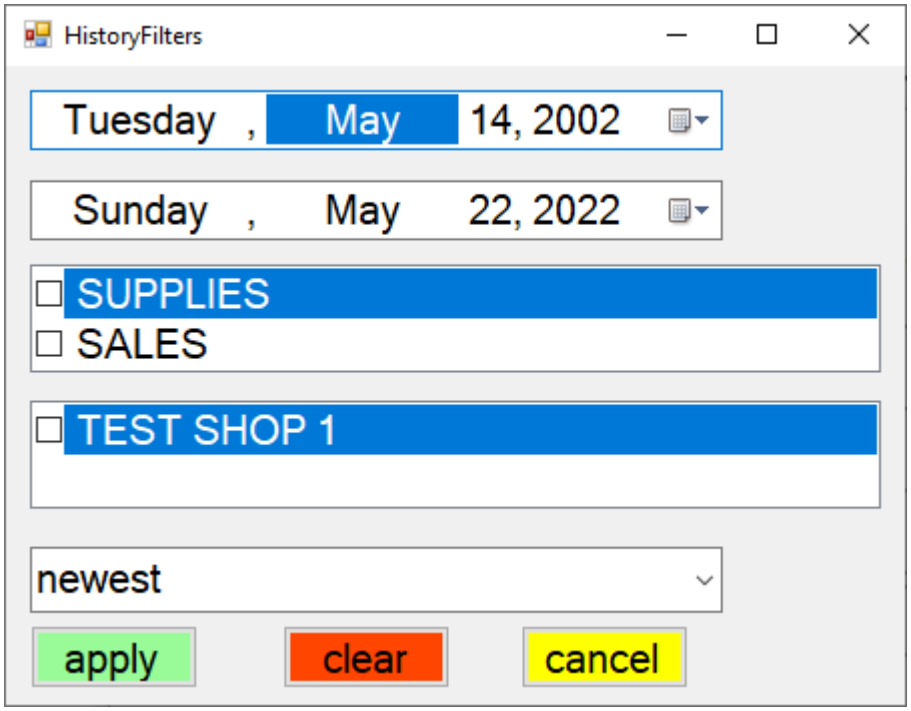

Рисунок 2.4 – форма фильтрации

Для добавления новой записи в журнал необходимо нажать на кнопку “Add” форма журнала поставок\продаж. Откроется новая форма (рисунок 2.5). В окне расположены поля для выбора типа (продажа или поставка), кондитерской, даты, кнопка добавления продуктов “add”, а также кнопка подтверждения покупки \ продажи “submit”. При необходимости отредактировать существующие покупки/продажи необходимо нажать на кнопку “edit” для открытия окна администрирования (рисунок 2.6), это окно так же может быть открыто с главной формы (рисунок 2.1) кнопкой “administration”

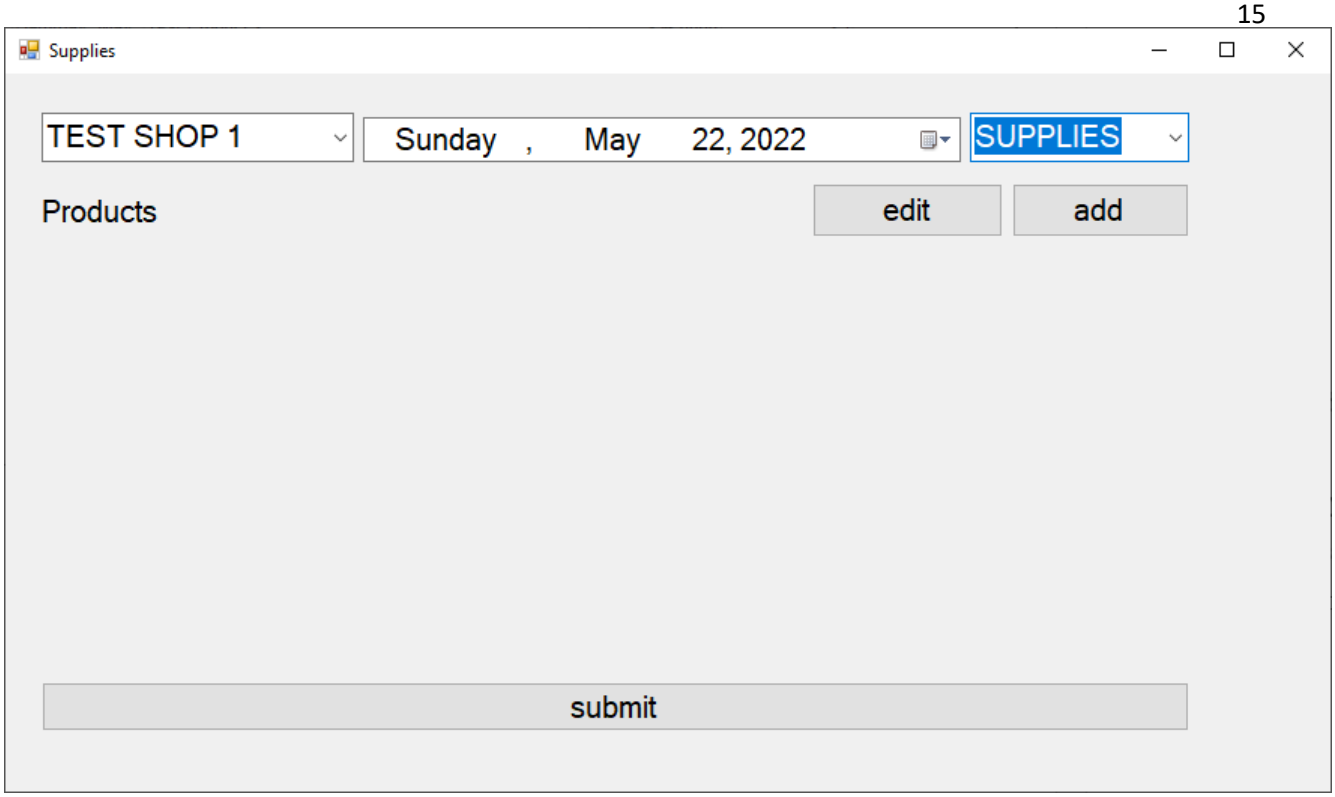

Рисунок 2.5 - форма добавления новой поставки продажи

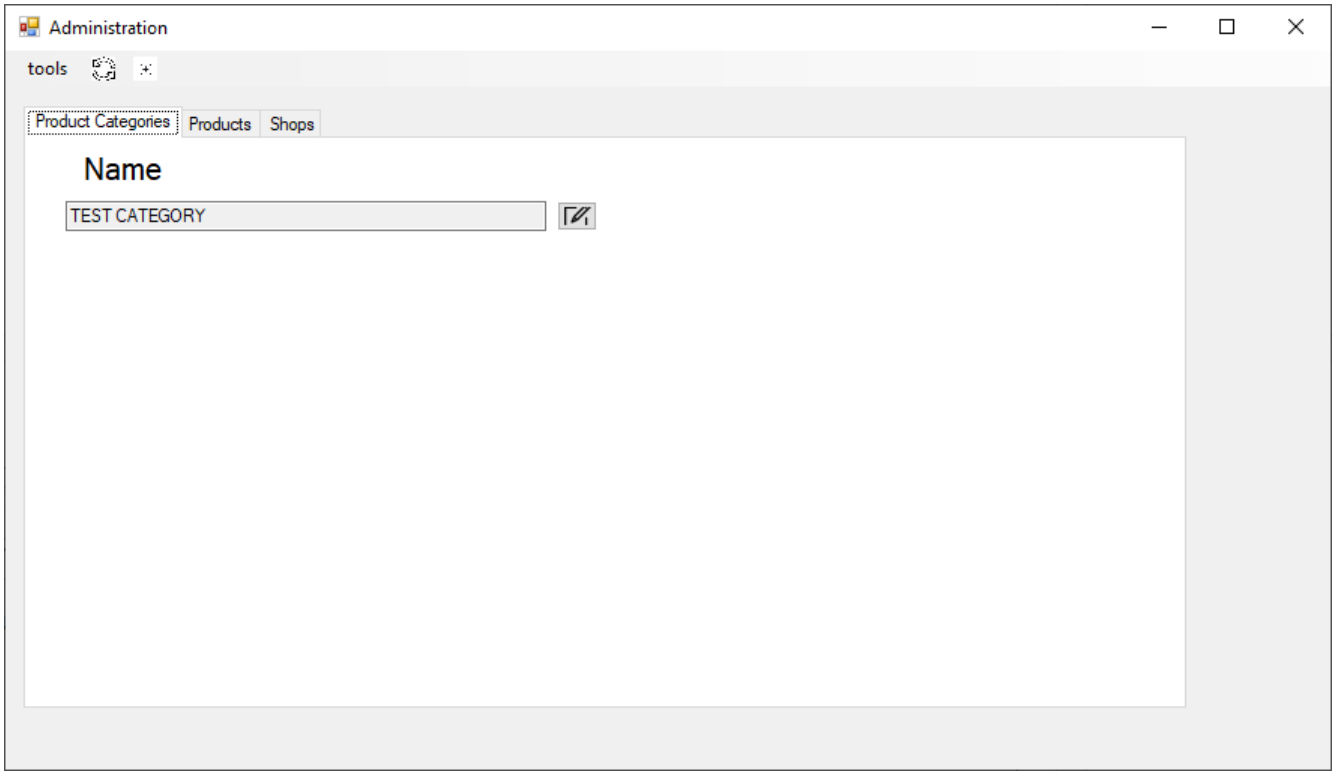

Рисунок 2.6 – форма администрирования

Так же в программе есть возможность просмотреть статистический анализ, для этого из главной формы (рисунок 2.1) необходимо нажать на кнопку “statistics”. Откроется новая форма (рисунок 2.7). В окне расположены истории журнала покупок продаж, сгруппированные по операциям их совершения, при нажатии кнопки “more” откроется детальный отчет той или иной операции (рисунок 2.8). При нажатии на кнопку “report” откроется статистический анализ (рисунок 2.9). Также есть возможность отфильтровать операции для анализа, через нажатие на кнопку “filter” и редактирование окна фильтрации (рисунок 2.4).

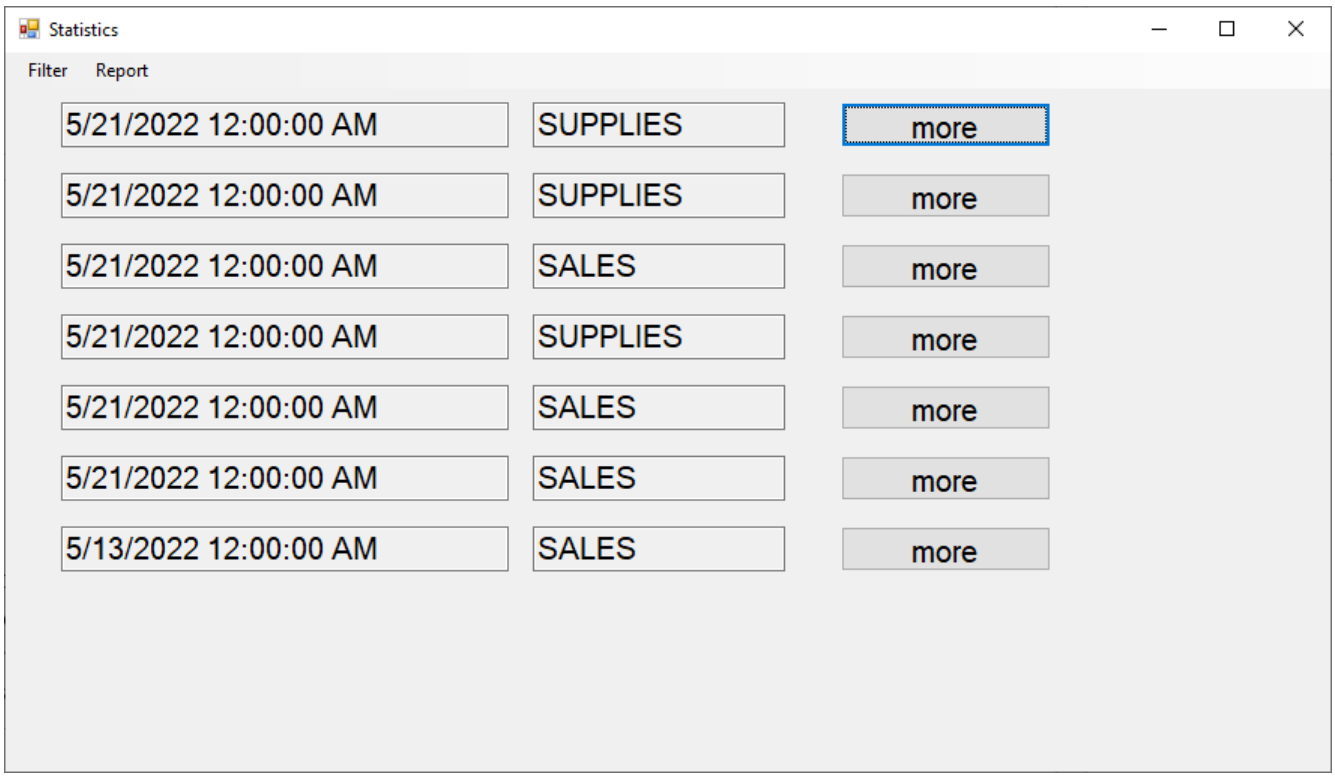

Рисунок 2.7 – форма статистического анализа

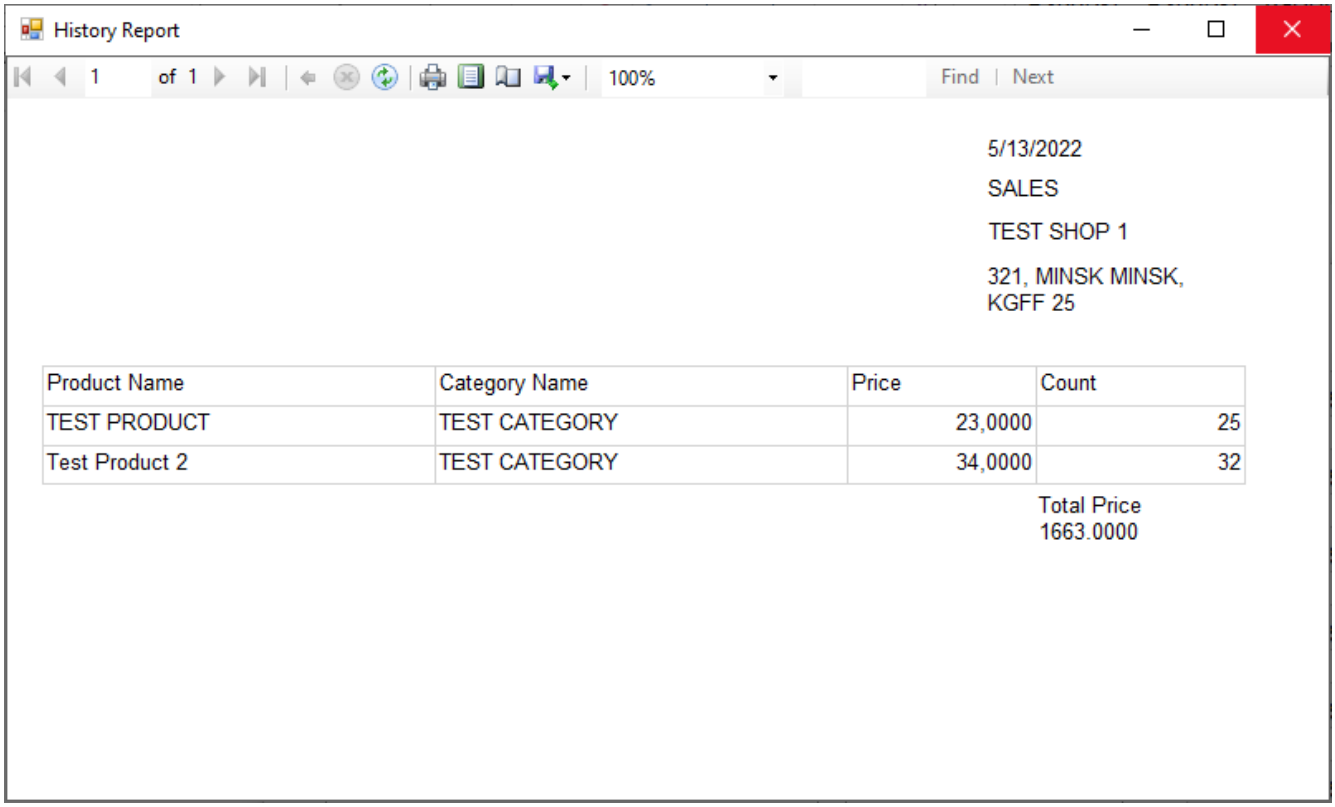

Рисунок 2.8 – форма отчета об операции

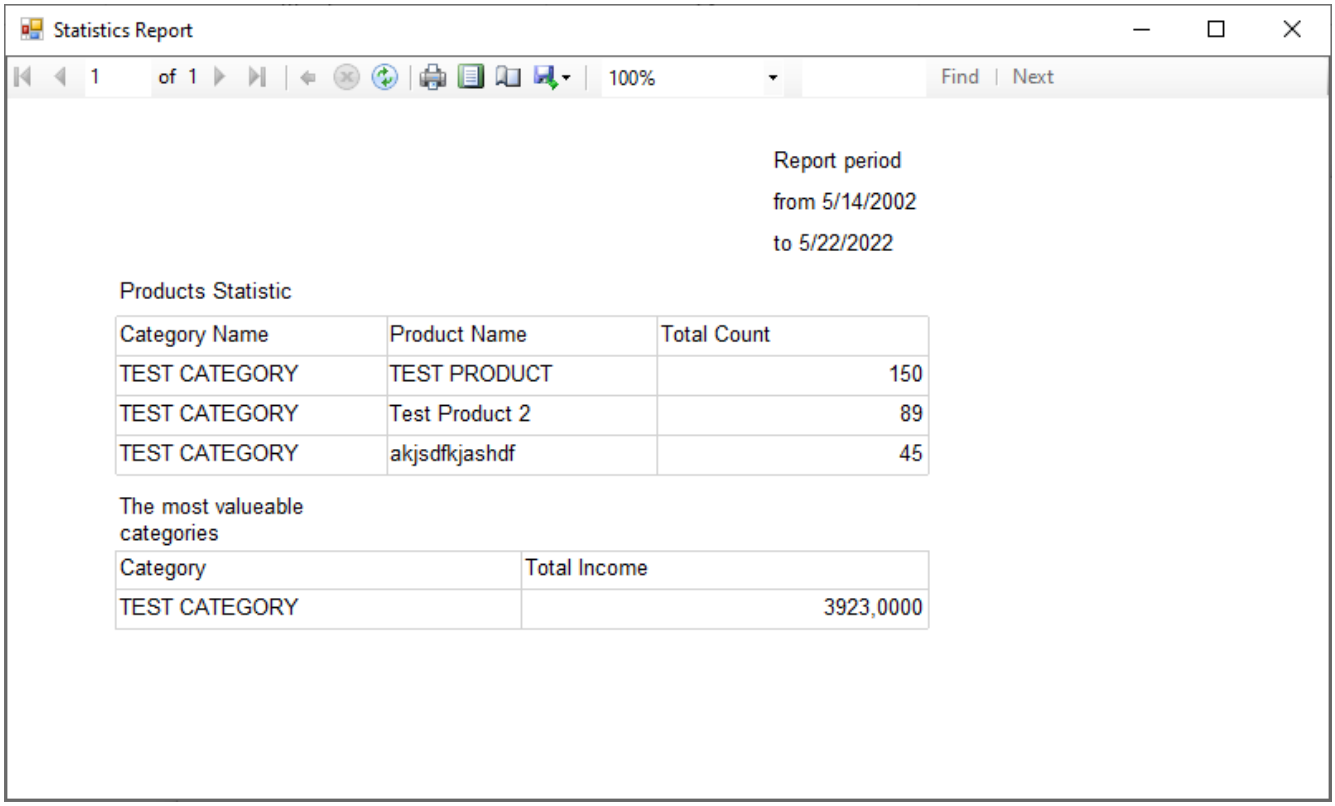

Рисунок 2.9 – отчет статистического анализа

3 Графические материалы

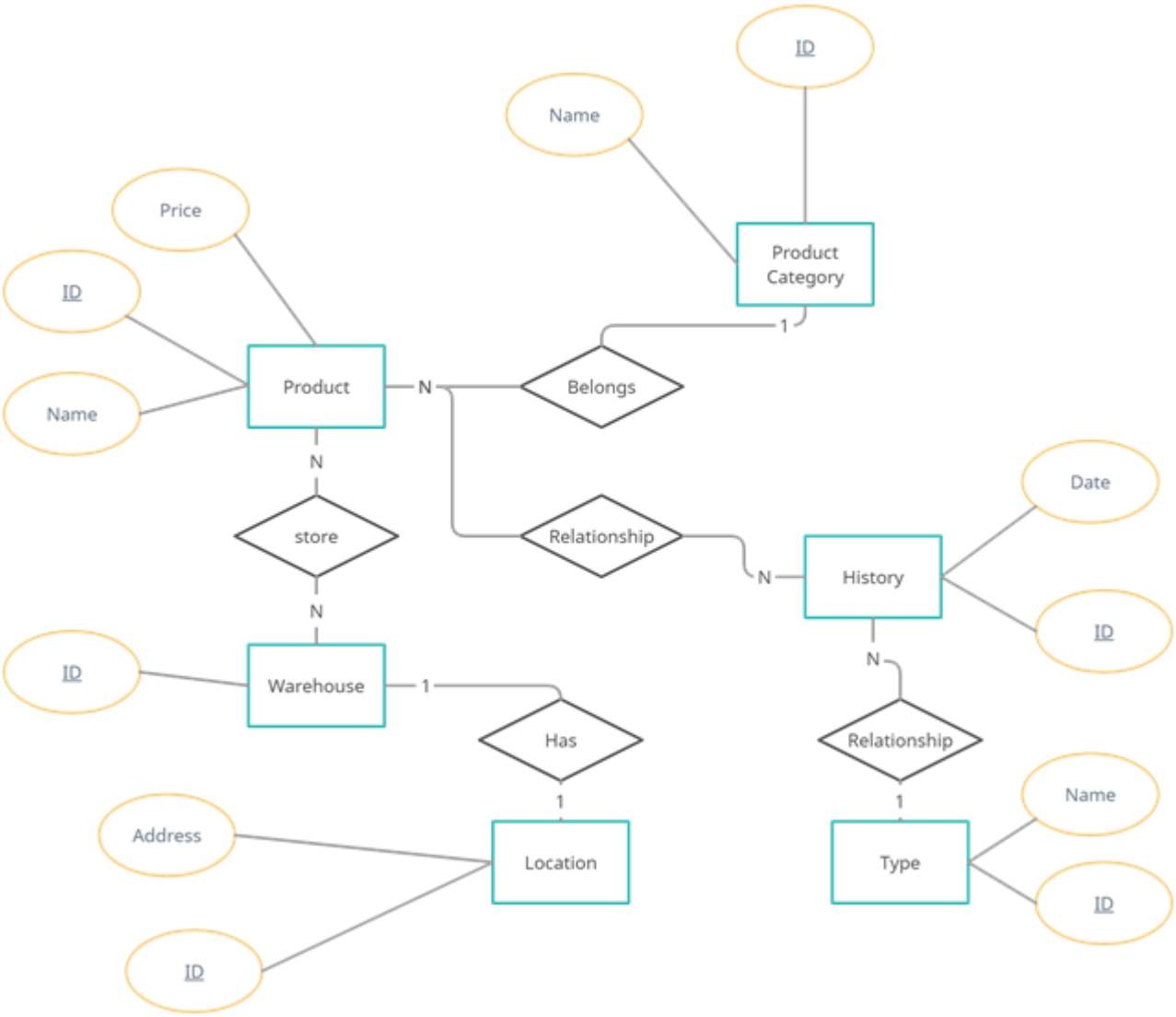

Рисунок 4.1 – концептуальная моедель базы данных

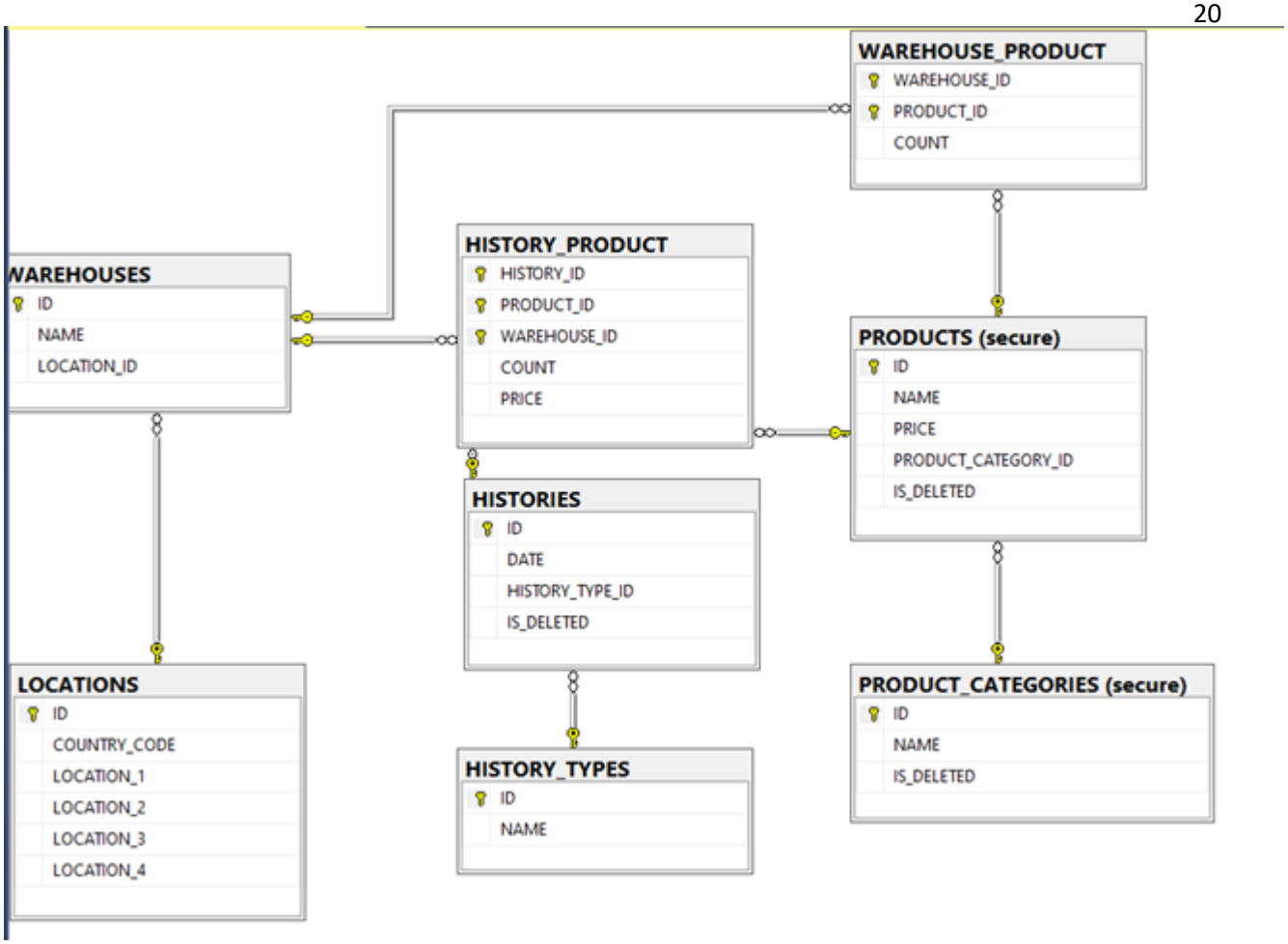

Рисунок 4.2 – логическая модель базы данных

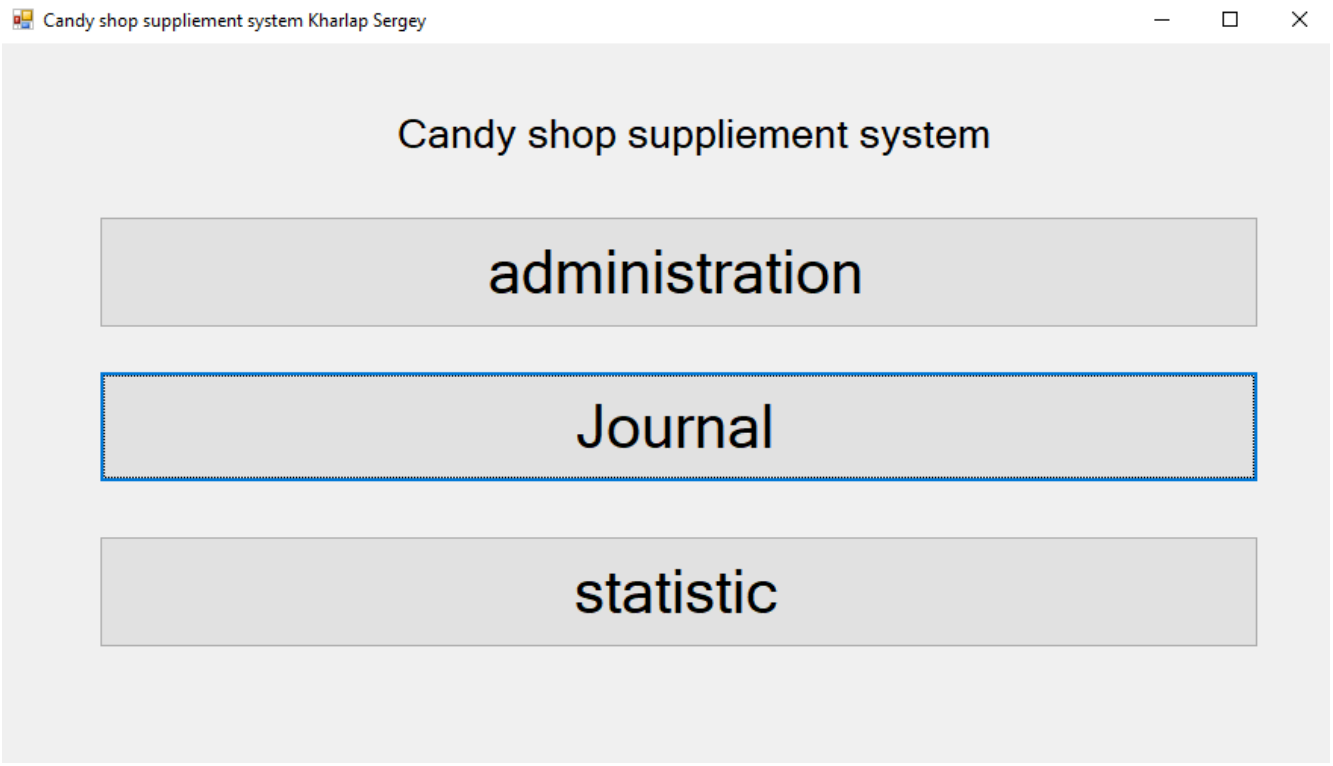

Рисунок 4.3 – главная форма

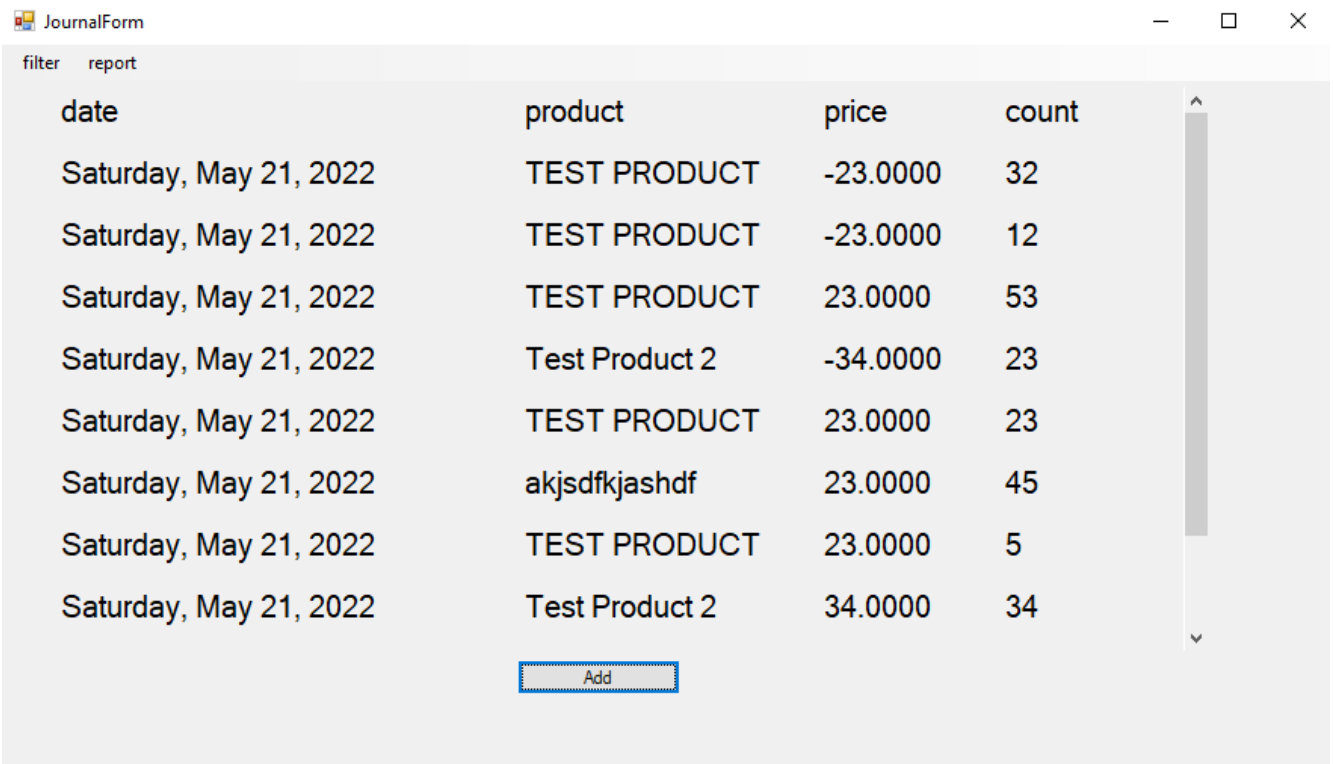

Рисунок 4.4 – форма журнала

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной курсовой работы была спроектирована база данных и разработано приложение для работы с этой базой данных. Исходя из цели, было разработана информационная система контроля закупок и продаж кондитерской Также имеется возможность получить отчет с результирующий прибылью за определенный период.

Приложение имеет пользовательский интерфейс, разработанный посредством использования средств, предоставляемых Microsoft Visual Studio 2022. Приложение написано на языке C#. В связи с этим, данных продукт способен работать на любых машинах, поддерживающих операционную систему Windows.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. «Курсовое проектирование. Общие правила и требования оформления. СТП БНТУ 3.01 – 2003.» — Минск, 2003. – Яз. Русск.

2. «Изучаем SQL» — Алан Бьюли. – Яз. Русск.

3. sql-language [Электронный ресурс]. -sql-language, 20.05.2018 – Язык запросов SQL https://sql-language.ru/query-select.html, свободный, - Загл. С экрана. – Яз. Русск.

4. Джон Шарп. Microsoft Visual C#. Подробное руководство. 8-е издание, 2017. – 845 с.

ПРИЛОЖЕНИЕ

Код создания базы данных

```
IF OBJECT_ID(N'dbo.HISTORY_PRODUCT') IS NOT NULL
	DROP TABLE HISTORY_PRODUCT
GO
IF OBJECT_ID(N'dbo.HISTORIES') IS NOT NULL
	DROP TABLE HISTORIES
GO
IF OBJECT_ID(N'dbo.HISTORY_TYPES') IS NOT NULL
	DROP TABLE HISTORY_TYPES
GO
IF OBJECT_ID(N'dbo.WAREHOUSE_PRODUCT') IS NOT NULL
	DROP TABLE WAREHOUSE_PRODUCT
GO
IF OBJECT_ID(N'dbo.WAREHOUSES') IS NOT NULL
	DROP TABLE WAREHOUSES
GO
IF OBJECT_ID(N'dbo.LOCATIONS') IS NOT NULL
	DROP TABLE LOCATIONS
GO
IF OBJECT_ID(N'secure.PRODUCTS') IS NOT NULL
	DROP TABLE secure.PRODUCTS
GO
IF OBJECT_ID(N'secure.PRODUCT_CATEGORIES') IS NOT NULL
	DROP TABLE secure.PRODUCT_CATEGORIES
GO

IF EXISTS (SELECT * FROM sys.schemas WHERE name = 'secure')
	DROP SCHEMA secure;
GO
CREATE SCHEMA secure
Go

CREATE TABLE secure.PRODUCT_CATEGORIES (
	ID INT IDENTITY(1,1) NOT NULL,

	NAME NVARCHAR(100) NOT NULL,
	IS_DELETED BIT NOT NULL DEFAULT 0,

	CONSTRAINT PK_PRODUCT_CATEGORY_ID PRIMARY KEY CLUSTERED (ID)
)
GO

CREATE TABLE secure.PRODUCTS (
	ID INT IDENTITY(1,1) NOT NULL,

	NAME VARCHAR(100) NOT NULL,
	PRICE MONEY NOT NULL DEFAULT 0,

	PRODUCT_CATEGORY_ID INT NOT NULL,
	IS_DELETED BIT NOT NULL DEFAULT 0,

	CONSTRAINT PK_PRODUCT_ID PRIMARY KEY CLUSTERED (ID),
	CONSTRAINT FK_PRODUCT_PRODUCT_CATEGORY FOREIGN KEY (PRODUCT_CATEGORY_ID)
		REFERENCES secure.PRODUCT_CATEGORIES (ID)
		ON DELETE CASCADE
		ON UPDATE CASCADE,

	CONSTRAINT CHK_PRODUCT_PRICE
		CHECK (PRICE > 0)
)
GO

CREATE TABLE LOCATIONS (
	ID INT IDENTITY(1,1) NOT NULL,

	COUNTRY_CODE NVARCHAR(100) NOT NULL,
	LOCATION_1 NVARCHAR(100) NULL,
	LOCATION_2 NVARCHAR(100) NULL,
	LOCATION_3 NVARCHAR(100) NULL,

	CONSTRAINT PK_LOCATION_ID PRIMARY KEY CLUSTERED (ID),
)

ALTER TABLE LOCATIONS
ADD LOCATION_4 NVARCHAR(100) NULL
```

```
GO
CREATE TABLE WAREHOUSES (
	ID INT IDENTITY(1,1) NOT NULL,

	NAME NVARCHAR(100) NOT NULL,
	LOCATION_ID INT NOT NULL,

	CONSTRAINT PK_WAREHOUSE_ID PRIMARY KEY CLUSTERED (ID),
	CONSTRAINT FK_WAREHOUSE_LOCATION_ID FOREIGN KEY (LOCATION_ID)
		REFERENCES LOCATIONS (ID)
		ON DELETE NO ACTION
		ON UPDATE CASCADE,
)
GO

GO
CREATE TABLE HISTORY_TYPES (
	ID INT IDENTITY(1,1) NOT NULL,

	NAME VARCHAR(100) NOT NULL,

	CONSTRAINT PK_HISTORY_TYPE_ID PRIMARY KEY CLUSTERED (ID),
)
GO

INSERT INTO HISTORY_TYPES(NAME)
VALUES ('SUPPLIES'),('SALES')
GO

CREATE TABLE HISTORIES (
	ID INT IDENTITY(1,1) NOT NULL,

	DATE DATETIME NOT NULL DEFAULT GETDATE(),
	HISTORY_TYPE_ID INT NOT NULL,
	IS_DELETED BIT NOT NULL DEFAULT 0,

	CONSTRAINT PK_HISOTRY_ID PRIMARY KEY CLUSTERED (ID),
	CONSTRAINT FK_HISTORY_HISTORY_TYPE_ID FOREIGN KEY (HISTORY_TYPE_ID)
		REFERENCES HISTORY_TYPES (ID)
		ON DELETE NO ACTION
		ON UPDATE CASCADE,
)
GO
CREATE TABLE HISTORY_PRODUCT (
	HISTORY_ID INT NOT NULL,
	PRODUCT_ID INT NOT NULL,
	WAREHOUSE_ID INT NOT NULL,

	COUNT INT NOT NULL DEFAULT 1,
	PRICE MONEY NOT NULL DEFAULT 1,

	CONSTRAINT PK_HISTORY_PRODUCT PRIMARY KEY CLUSTERED (HISTORY_ID, PRODUCT_ID, WAREHOUSE_ID),
	CONSTRAINT FK_HISTORY_PRODUCT_HISTORY_ID FOREIGN KEY (HISTORY_ID)
		REFERENCES HISTORIES (ID)
		ON DELETE NO ACTION
		ON UPDATE CASCADE,
	CONSTRAINT FK_HISTORY_PRODUCT_PRODUCT_ID FOREIGN KEY (PRODUCT_ID)
		REFERENCES secure.PRODUCTS (ID)
		ON DELETE NO ACTION
		ON UPDATE CASCADE,
	CONSTRAINT FK_HISTORY_PRODUCT_WAREHOUSE_ID FOREIGN KEY (WAREHOUSE_ID)
		REFERENCES WAREHOUSES (ID)
		ON DELETE NO ACTION
		ON UPDATE CASCADE,
)
GO
CREATE TABLE WAREHOUSE_PRODUCT (
	WAREHOUSE_ID INT NOT NULL,
	PRODUCT_ID INT NOT NULL,

	CONSTRAINT PK_WAREHOUSE_PRODUCT_I PRIMARY KEY CLUSTERED (WAREHOUSE_ID, PRODUCT_ID),
	CONSTRAINT FK_WAREHOUSE_PRODUCT_WAREHOUSE_ID FOREIGN KEY (WAREHOUSE_ID)
		REFERENCES WAREHOUSES (ID)
		ON DELETE NO ACTION
		ON UPDATE CASCADE,
	CONSTRAINT FK_WAREHOUSE_PRODUCT_PRODUCT_ID FOREIGN KEY (PRODUCT_ID)
		REFERENCES secure.PRODUCTS (ID)
		ON DELETE NO ACTION
		ON UPDATE CASCADE,
)

GO
```

```
ALTER TABLE WAREHOUSE_PRODUCT
	ADD COUNT INT NOT NULL DEFAULT 0;

GO
```

Код создания представлений базы данных

```
CREATE OR ALTER VIEW PRODUCT_CATEGORIES
AS
	SELECT ID, NAME
	FROM secure.PRODUCT_CATEGORIES
	WHERE IS_DELETED = 0
GO

CREATE OR ALTER VIEW PRODUCTS
AS
	SELECT ID, NAME, PRICE, PRODUCT_CATEGORY_ID
	FROM secure.PRODUCTS
	WHERE IS_DELETED = 0
GO
CREATE OR ALTER VIEW HISTORIES
AS
	SELECT ID, DATE, HISTORY_TYPE_ID
	FROM secure. HISTORIES
	WHERE secure.IS_DELETED = 0
GO
```

Код создания процедур

```
CREATE OR ALTER PROCEDURE SOFT_PRODUCT_CATEGORY_DELETE
	@CATEGORY_ID INT
AS
BEGIN

	UPDATE secure.PRODUCT_CATEGORIES
	SET IS_DELETED = 1
	WHERE ID = @CATEGORY_ID

	UPDATE secure.PRODUCTS
	SET IS_DELETED = 1
	WHERE @CATEGORY_ID = PRODUCT_CATEGORY_ID

END
```

Код создания тригеров

```
CREATE OR ALTER TRIGGER INSTEAD_DELETE_PRODUCT_CATEGORIES_TRIGGER ON PRODUCT_CATEGORIES
INSTEAD OF DELETE
AS
BEGIN

	IF EXISTS(
		SELECT TOP 1 *
		FROM secure.PRODUCTS
		WHERE PRODUCT_CATEGORY_ID IN (SELECT ID FROM DELETED)
		)
		THROW 50001, 'UNABLE TO DELETE PRODUCT CATEGORY THAT HAS PRODUCTS', 1;

	UPDATE secure.PRODUCT_CATEGORIES
	SET IS_DELETED = 1
	WHERE ID IN (SELECT ID FROM DELETED)

END;
GO

CREATE OR ALTER TRIGGER INSTEAD_DELETE_PRODUCTS_TRIGGER ON PRODUCTS
INSTEAD OF DELETE
AS
BEGIN

	UPDATE secure.PRODUCTS
	SET IS_DELETED = 1
	WHERE ID IN (SELECT ID FROM DELETED)

END;
GO
```

Код создания инфраструктуры журнала

```
CREATE TYPE PRODUCT_LIST_TYPE_2 AS TABLE
(
	PRODUCT_ID INT NOT NULL,
	AMOUNT INT NOT NULL,
	PRICE MONEY NOT NULL
);
GO

CREATE OR ALTER PROCEDURE CHANGE_PRODUCT_WAREHOUSE_LINK
	@PRODUCT_ID INT,
	@WAREHOUSE_ID INT,
	@COUNT INT
AS
BEGIN

	IF EXISTS(SELECT * FROM WAREHOUSE_PRODUCT WHERE PRODUCT_ID = @PRODUCT_ID AND WAREHOUSE_ID =
@WAREHOUSE_ID)
		UPDATE WAREHOUSE_PRODUCT
		SET COUNT = COUNT + @COUNT
		WHERE PRODUCT_ID = @PRODUCT_ID AND WAREHOUSE_ID = @WAREHOUSE_ID
	ELSE
		INSERT INTO WAREHOUSE_PRODUCT (WAREHOUSE_ID, PRODUCT_ID, COUNT)
		VALUES (@WAREHOUSE_ID, @PRODUCT_ID, @COUNT)

END;

GO

CREATE OR ALTER PROCEDURE CREATE_HISTORY
	@HISTORY_TYPE_ID INT ,
	@DATE DATETIME,
	@HISTORY_ID INT OUTPUT
AS
BEGIN
	INSERT INTO HISTORIES (DATE, HISTORY_TYPE_ID)
	VALUES (@DATE, @HISTORY_TYPE_ID)

	SET @HISTORY_ID = SCOPE_IDENTITY();

END

GO

GO
CREATE OR ALTER PROCEDURE NEW_HISTORY_EVENT
	@PRODUCT_ITEMS PRODUCT_LIST_TYPE_2 READONLY,
	@HISTORY_TYPE_ID INT,
	@WAREHOUSE_ID INT,
	@DATE DATETIME,
	@UPDATER INT
AS
BEGIN

	DECLARE @HISTORY_ID INT;
	EXECUTE CREATE_HISTORY @HISTORY_TYPE_ID, @DATE, @HISTORY_ID OUTPUT;

	INSERT INTO HISTORY_PRODUCT (PRODUCT_ID, HISTORY_ID, COUNT, WAREHOUSE_ID, PRICE)
	SELECT P.PRODUCT_ID, @HISTORY_ID, @UPDATER * P.AMOUNT, @WAREHOUSE_ID, PRICE*-1*@UPDATER
	FROM @PRODUCT_ITEMS P

END;

GO

CREATE OR ALTER PROCEDURE NEW_SALE
	@PRODUCT_ITEMS PRODUCT_LIST_TYPE_2 READONLY,
	@WAREHOUSE_ID INT,
	@DATE DATETIME
AS
BEGIN
	EXEC NEW_HISTORY_EVENT @PRODUCT_ITEMS, 2, @WAREHOUSE_ID, @DATE, -1
END;

GO

CREATE OR ALTER PROCEDURE NEW_SUPPLY
	@PRODUCT_ITEMS PRODUCT_LIST_TYPE_2 READONLY,
	@WAREHOUSE_ID INT,
	@DATE DATETIME
```

```
AS
BEGIN
	EXEC NEW_HISTORY_EVENT @PRODUCT_ITEMS, 1, @WAREHOUSE_ID, @DATE, 1
END;

GO


GO

CREATE OR ALTER TRIGGER AFTER_INSERT_TRIGGER ON HISTORY_PRODUCT
AFTER INSERT
AS
BEGIN
	DECLARE
		@TEMP AS TABLE(
			ROW_N INT,
			PRODUCT_ID INT,

			WAREHOUSE_ID INT,
			COUNT INT
		)
	DECLARE
		@PRODUCT_ID INT,
		@WAREHOUSE_ID INT,
		@COUNT INT

	INSERT INTO @TEMP
	SELECT ROW_NUMBER() OVER(ORDER BY COUNT ASC), PRODUCT_ID, WAREHOUSE_ID, COUNT FROM INSERTED

	WHILE Exists
	(
		select 1
		from @TEMP
	)
	begin
		declare @TEMP_ID INT;

		SELECT TOP 1
			@TEMP_ID = ROW_N,
			@PRODUCT_ID = PRODUCT_ID,
			@WAREHOUSE_ID = WAREHOUSE_ID,
			@COUNT = COUNT
		from @TEMP

		EXECUTE CHANGE_PRODUCT_WAREHOUSE_LINK @PRODUCT_ID, @WAREHOUSE_ID, @COUNT

		delete
		from @TEMP
		where ROW_N = @TEMP_ID
	end
END;

GO
```

Код создания функций

```
CREATE OR ALTER FUNCTION GET_STATISTICS (
	@FROM_DATE DATE,
	@TO_DATE DATE
	)
RETURNS TABLE
AS
RETURN (
	SELECT H.ID AS HISTORY, H.DATE, ABS(HP.COUNT) AS COUNT, P.ID AS PRODUCT_ID, P.NAME AS PRODUCT, HP.PRICE,
PC.NAME AS CATEGORY, PC.ID AS CATEGORY_ID
	FROM HISTORIES AS H
	JOIN HISTORY_TYPES AS HT ON HT.ID = H.HISTORY_TYPE_ID
	JOIN HISTORY_PRODUCT AS HP ON HP.HISTORY_ID = H.ID
	JOIN PRODUCTS AS P ON P.ID = HP.PRODUCT_ID
	JOIN PRODUCT_CATEGORIES PC ON PC.ID = P.PRODUCT_CATEGORY_ID
	JOIN WAREHOUSES AS W ON W.ID = HP.WAREHOUSE_ID
	JOIN LOCATIONS AS L ON L.ID = W.LOCATION_ID
	WHERE H.DATE BETWEEN @FROM_DATE AND @TO_DATE
)

GO

CREATE OR ALTER FUNCTION GET_STATISTICS_CATEGORY (
	@FROM_DATE DATE,
```

```
	@TO_DATE DATE)
RETURNS TABLE
AS
RETURN
	SELECT TOP(1) WITH TIES CATEGORY_ID, CATEGORY, SUM(COUNT) AS TOTAL_COUNT
	FROM GET_STATISTICS(@FROM_DATE, @TO_DATE)
	GROUP BY CATEGORY_ID, CATEGORY
	ORDER BY SUM(COUNT)

GO

CREATE OR ALTER FUNCTION GET_STATISTICS_CATEGORY_PRICE (
	@FROM_DATE DATE,
	@TO_DATE DATE)
RETURNS TABLE
AS
RETURN
	SELECT TOP(1) WITH TIES CATEGORY_ID, CATEGORY, SUM(COUNT*PRICE) AS TOTAL_SUM
	FROM GET_STATISTICS(@FROM_DATE, @TO_DATE)
	GROUP BY CATEGORY_ID, CATEGORY
	ORDER BY SUM(COUNT*PRICE)

GO

CREATE OR ALTER FUNCTION GET_STATISTICS_PRODUCTS (
	@FROM_DATE DATE,
	@TO_DATE DATE)
RETURNS TABLE
AS
RETURN
	SELECT PRODUCT, CATEGORY, SUM(COUNT) AS TOTAL_COUNT
	FROM GET_STATISTICS(@FROM_DATE, @TO_DATE)
	GROUP BY PRODUCT_ID, PRODUCT, CATEGORY
GO
```

Код бизнес логики

```
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Data.Entity.Infrastructure;
using System.Data.Entity.Migrations;
using System.Linq;
using System.Security.Cryptography;
using CourseWork.Domain;
using CourseWork.Domain.StoredProceduresTypes;
using SweetShop.Models.Entities;
using SweetShop.Models.Filters;

namespace SweetShop.BusinessLogic
{
   public class HistoryService
   {
      private readonly SweetShopEntities _context;
      private readonly ProductsService _productsService = new ProductsService();
      public HistoryService()
      {
         _context = new SweetShopEntities();
      }
      public List<HistoryType> GetHistoryTypes()
      {
         var types = _context.HistoryTypes.Select(ConvertHelper.Convert).ToList();
         return types;
      }

      public void CreateNewHistory(
         Warehouse warehouse,
         HistoryType historyType,
         DateTime date,
         List<(Product product, int amount)> selectedProducts
      )
      {
         var selected = selectedProducts.Select(pc => new ProductListType
            {Amount = pc.amount, ProductId = pc.product.Id, Price = pc.product.Price}).ToList();

         BaseHistoryEvent proc;
         if (historyType.Name.EndsWith("SALES", StringComparison.CurrentCultureIgnoreCase))
            proc = new NewSalesProcedure
            {
               Date = date,
               SelectedProducts = selected,
               WarehouseId = warehouse.Id
```

```
        };
    else
        proc = new NewSupplyProcedure()
        {
            Date = date,
            SelectedProducts = selected,
            WarehouseId = warehouse.Id
        };

    _context.NEW_HISTORY_EVENT(proc);
}

public List<History> GetHistories(HistoryFilter filter, bool includeProducts = false)
{
    var query = _context.Histories.Select(h => h).Include(h => h.HistoryType);
    if (filter.FromDate.HasValue)
        query = query.Where(h => h.DATE >= filter.FromDate.Value);
    if (filter.ToDate.HasValue)
        query = query.Where(h => h.DATE <= filter.ToDate.Value);
    if (filter.SelectedTypes.Any())
        query = query.Where(h => filter.SelectedTypes.Contains(h.HISTORY_TYPE_ID));
    if (filter.SelectedStores.Any())
        query = query
            .Join(
                _context.HistoryProducts,
                h => h.ID,
                hp => hp.HISTORY_ID,
                (h, hp) => new
                {
                    h,
                    hp
                }
            )
            .Where(joint => filter.SelectedStores.Contains(joint.hp.WAREHOUSE_ID))
            .Select(joint => joint.h)
            .GroupBy(h => h)
            .Select(h => h.Key);

    query = filter.IsDescending ? query.OrderByDescending(h => h.DATE) : query.OrderBy(h => h.DATE);

    var histories = query.ToList();
    return histories.Select(
        history =>
        {
            var h = new History
            {
                Id = history.ID,
                Date = history.DATE,
                HistoryType = ConvertHelper.Convert(history.HistoryType),
                HistoryTypeId = history.HISTORY_TYPE_ID,
            };

            if (includeProducts)
            {
                h.ProductHistories = history?.HistoryProducts?
                    .Select(
                    historyProduct =>
                    new ProductHistory
                    {
                        Count = Math.Abs(historyProduct.COUNT),
                        History = h,
                        Product = ConvertHelper.Convert(historyProduct.Product),
                        Price = historyProduct.PRICE
                    }
                ).ToList() ?? new List<ProductHistory>();
            }
            return h;
        }
    ).ToList();


    //var histories = query.Join(
    //       _context.HistoryProducts,
    //       h => h.ID,
    //       hp => hp.HISTORY_ID,
    //       (h, hp) => new
    //       {
    //           h,
    //           hp
    //       })
    //    .ToList();

    //var result = histories.GroupBy(joint => joint.h)
```

```
            //    .Select(
            //        g =>
            //        {
            //            var history = g.Key;

            //            return new History
            //            {
            //                Id = history.ID,
            //                Date = history.DATE,
            //                HistoryType = ConvertHelper.Convert(history.HistoryType),
            //                HistoryTypeId = history.HISTORY_TYPE_ID,
            //                ProductHistories = g.Select(
            //                    joint => new ProductHistory
            //                    {
            //                        Count = joint.hp.COUNT,
            //                        HistoryId = history.ID,
            //                        ProductId = joint.hp.PRODUCT_ID,
            //                        WarehouseId = joint.hp.WAREHOUSE_ID
            //                    }
            //                ).ToList()
            //            };
            //        }
            //    )
            //    .ToList();
            //return result;
        }

        public History GetHistory(int id)
        {
            var temp = _context.HistoryProducts.ToList();
            var history = _context.Histories
                .Include(h => h.HistoryType)
                .FirstOrDefault(h => h.ID == id);

            if (history == null)
                return null;

            var historyProducts = _context.HistoryProducts.Include(hp => hp.Product).Include(hp => hp.Warehouse).Include(hp =>
hp.Warehouse.Location)
                .Where(hp => hp.HISTORY_ID == history.ID)
                .ToList();

            return new History
            {
                HistoryType = ConvertHelper.Convert(history.HistoryType),
                Date = history.DATE,
                HistoryTypeId = history.HISTORY_TYPE_ID,
                Id = history.ID,
                ProductHistories = historyProducts.Select(
                    hp =>
                    {
                        var category = _productsService.GetProductCategory(hp.Product.PRODUCT_CATEGORY_ID);
                        var product = ConvertHelper.Convert(hp.Product);
                        product.Category = category;
                        product.CategoryId = category.Id;

                        return new ProductHistory
                        {
                            Count = hp.COUNT,
                            Product = product,
                            Warehouse = ConvertHelper.Convert(hp.Warehouse, hp.Warehouse.Location),
                            ProductId = product.Id,
                            WarehouseId = hp.WAREHOUSE_ID
                        };
                    }
                    ).ToList()
            };
        }

        public List<ProductsStatistics> GetProductsStatistics(DateTime from, DateTime to)
        {
            return _context.GET_STATISTICS_PRODUCTS(from, to).Select(
                sp => new ProductsStatistics
                {
                    CategoryName = sp.CategoryName,
                    ProductName = sp.ProductName,
                    TotalCount = sp.TotalCount
                }
                ).ToList();
        }

        public List<CategoryStatistics> GetCategoryStatistics(DateTime from, DateTime to)
        {
```

```
            return _context.GET_STATISTICS_CATEGORY_PRICE(from, to).Select(
                sp => new CategoryStatistics()
                {
                    Category = sp.CATEGORY,
                    TotalIncome = sp.TOTAL_SUM,
                }
            ).ToList();
        }
    }
}
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.Entity.Migrations;
using System.Linq;
using System.Runtime.Remoting.Contexts;
using System.Text.RegularExpressions;
using CourseWork.Domain;
using SweetShop.Models.Entities;

namespace SweetShop.BusinessLogic
{
    public class ProductsService
    {
        private readonly SweetShopEntities _context;

        public ProductsService()
        {
            _context = new SweetShopEntities();
        }

        #region product

        public List<Product> GetProducts()
        {
            var context = new SweetShopEntities();

            var temp = context.Products.Join(
                context.ProductCategories,
                p => p.PRODUCT_CATEGORY_ID,
                pc => pc.ID,
                (p, pc) =>
                    new
                    {
                        Product = p,
                        Category = pc
                    }
            ).ToList();
            return temp
                .Select(
                    join => ConvertHelper.Convert(join.Product, join.Category)
                )
                .ToList();
        }

        public void AddOrUpdateProduct(Product product)
        {
            _context.Products.AddOrUpdate(ConvertHelper.Convert(product));
            _context.SaveChanges();
        }

        public Product DeleteProduct(int productId)
        {
            var toRemove = _context.Products.First(pcd => pcd.ID == productId);
            var removed = _context.Products.Remove(toRemove);
            _context.SaveChanges();
            return ConvertHelper.Convert(removed);
        }

        #endregion


        #region product category

        public List<ProductCategory> GetProductCategories(bool includeProducts = false)
        {
            var context = new SweetShopEntities();

            if (!includeProducts)
                return context.ProductCategories.ToList().Select(pc => ConvertHelper.Convert(pc)).ToList();
            //return context.PRODUCTS.Join(
            //        context.PRODUCT_CATEGORIES,
            //        p => p.PRODUCT_CATEGORY_ID,
```

```
            //        pc => pc.ID,
            //        (p, pc) =>
            //            new
            //            {
            //                Product = p,
            //                Category = pc
            //            }
            //    )
            //    .ToList()
            //    .GroupBy(
            //        joint => joint.Category,
            //        (key, group) => new
            //        {
            //            key,
            //            Items = group.ToList()
            //        }
            //    )
            //    .ToList()
            //    .Select(
            //        group =>
            //        {
            //            var category = ConvertHelper.Convert(group.Key);
            //            category.Products = group.ToList().Select(g => ConvertHelper.Convert(g.Product, group.Key))
            //                .ToList();
            //        }
            //    )
            //    .ToList();
            return null;
        }

        public ProductCategory GetProductCategory(int id, bool includeProducts = false)
        {
            var context = new SweetShopEntities();

            var category = context.ProductCategories.FirstOrDefault(pc => pc.ID == id);
            if (category == null)
                return null;

            var categoryModel = ConvertHelper.Convert(category);
            if (includeProducts)
                categoryModel.Products = context.Products
                    .Where(p => p.PRODUCT_CATEGORY_ID == categoryModel.Id)
                    .ToList()
                    .Select(p => ConvertHelper.Convert(p, category))
                    .ToList();

            return categoryModel;
        }

        public void AddOrUpdateCategory(ProductCategory pc)
        {
            var context = new SweetShopEntities();

            context.ProductCategories.AddOrUpdate(ConvertHelper.Convert(pc));
            context.SaveChanges();
        }

        public ProductCategory DeleteProductCategory(ProductCategory pc)
        {
            var toRemove = _context.ProductCategories.First(pcd => pcd.ID == pc.Id);
            var removed = _context.ProductCategories.Remove(toRemove);
            _context.SaveChanges();
            return ConvertHelper.Convert(removed);
        }

        public void DeleteProductCategoryWithCascade(ProductCategory pc)
        {
            if (pc == null)
                throw new ArgumentNullException(nameof(pc));

            _context.SOFT_PRODUCT_CATEGORY_DELETE(pc.Id);
        }

        #endregion

    }
} using System.Collections.Generic;
using System.Data.Entity;
using System.Data.Entity.Migrations;
using System.Linq;
using CourseWork.Domain;
using SweetShop.Models.Entities;
```

```
namespace SweetShop.BusinessLogic
{
    public class WarehouseService
    {
        private readonly SweetShopEntities _context;
        public WarehouseService()
        {
            _context = new SweetShopEntities();
        }
        public List<Warehouse> GetWarehouses()
        {
            var warehouses = _context.Warehouses.Include(w => w.Location).ToList();
            return warehouses.Select(w => ConvertHelper.Convert(w, w.Location)).ToList();
        }

        public void AddOrUpdate(Warehouse warehouse)
        {
            _context.Locations.AddOrUpdate(ConvertHelper.Convert(warehouse.Location));
            _context.Warehouses.AddOrUpdate(ConvertHelper.Convert(warehouse));
            _context.SaveChanges();
        }

        public void Delete(int warehouseId)
        {
            var toRemove = _context.Warehouses.First(pcd => pcd.ID == warehouseId);
            var removed = _context.Warehouses.Remove(toRemove);
            _context.SaveChanges();
        }
    }
}
```