МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«Санкт-Петербургский национальный исследовательский университет

информационных технологий, механики и оптики»

Хранилища данных.  
Итоговый отчёт

Выполнила: студентка группы М3310

Галаева Анастасия

Проверил:

Повышев Владислав Вячеславович

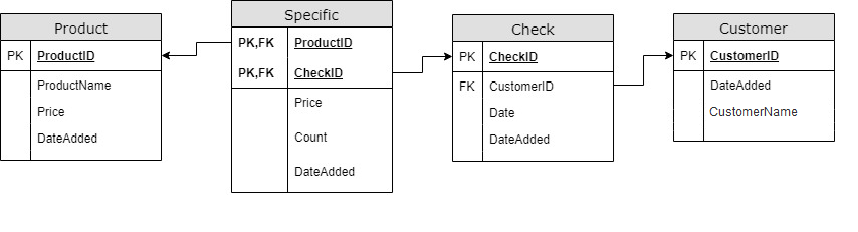
Санкт-Петербург, 2018

*Задача:*

*Создать базы данных для хранения информации о продажах в двух филиалах одной сети и визуализировать статистику продаж с помощью приложения на языке С#.*

*Шаг 1. Создание моделей баз.*

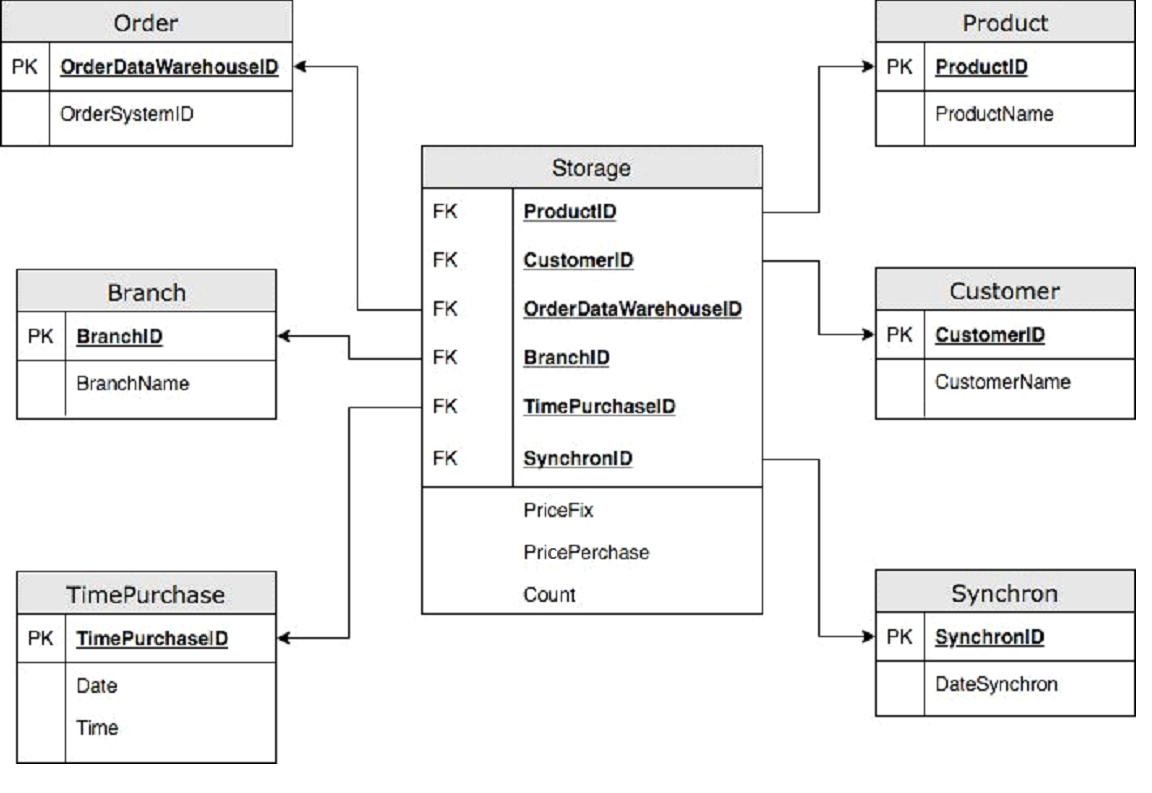
*Модель филиала:*



В таблице Product хранится информация о продуктах: уникальный ID, название, цена, дата ввода данных продукта в базу. *ProductID – первичный ключ.*  
В таблице Specific хранится информация о деталях покупки: ID продукта, ID чека, цена продажи продукта, количество, дата ввода данных в базу. *CheckID и ProductID – внешние ключи.*В таблице Check хранится информация о покупках чеках: уникальный ID, ID покупателя, дата покупки, дата ввода данных покупки в базу. *CheckID – первичный ключ, CustomerID – внешний ключ.*

В таблице Customer хранится информация о покупателях сети: уникальный ID, имя, дата ввода данных пользователя в базу. *CustomerID – первичный ключ.*

*Модель хранилища данных:*



В таблице Customer хранится уникальный ID покупателя и его имя. *CustomerID – первичный ключ.*

В таблице Order хранится информация о чеках: уникальный ID хранилища данных , ID чека в филиале, дата покупки. *OrderDataWarehouseID – первичный ключ.*

В таблице Product хранится информация о продуктах: уникальный ID, название. *ProductID – первичный ключ*.

В таблице Branch хранится информация о филиалах: название филиала, уникальный ID филиала. *BranchID – первичный ключ.*

В таблице Synchron хранится время последнего переноса данных из баз филиалов в хранилище данных: время, присваивается уникальный ID синхронизации. *SynchronizeID – первичный ключ.*

В таблице TimePurchase хранится время покупки: уникальный ID, дата, время. *TimePurchaseID– первичный ключ.*

Таблица Storage представляет сборную числовую информацию:  
Внешние ключи *CustomerID, OrderDataWarehouseID, ProductID, BranchID, SynchronizeID, TimePurchaseID.*

*Шаг 2. Создание баз.*

Скрипт создания базы данных первого филиала:

CREATE DATABASE Branch\_1

CREATE TABLE [dbo].[Customer](

[CustomerID] [int] PRIMARY KEY NOT NULL,

[CustomerName] [varchar](50) NOT NULL,

[DateAdded] [datetime] NOT NULL

)

CREATE TABLE [dbo].[Check](

[CheckID] [int] IDENTITY PRIMARY KEY NOT NULL,

[CustomerID] [int] NOT NULL,

[Date] [datetime] NOT NULL,

[DateAdded] [datetime] NOT NULL

)

CREATE TABLE [dbo].[Product](

[ProductID] [int] PRIMARY KEY NOT NULL,

[ProductName] [varchar](50) NOT NULL,

[Price] [money] NOT NULL,

[DateAdded] [datetime] NOT NULL

)

CREATE TABLE [dbo].[Specific](

[ProductID] [int] NOT NULL,

[CheckID] [int] NOT NULL,

[Price] [money] NOT NULL,

[Count] [int] NOT NULL,

[DateAdded] [datetime] NOT NULL

)

ALTER TABLE [dbo].[Specific]

ADD CONSTRAINT [PK\_CheckID\_ProductID]

PRIMARY KEY

CLUSTERED(CheckID,ProductID)

ALTER TABLE [dbo].[Check]

ADD CONSTRAINT [FK\_CustomerID]

FOREIGN KEY([CustomerID])

REFERENCES [dbo].[Customer] ([CustomerID])

ALTER TABLE [dbo].[Specific]

ADD CONSTRAINT [FK\_CheckID]

FOREIGN KEY([CheckID])

REFERENCES [dbo].[Check] ([CheckID])

ALTER TABLE [dbo].[Specific]

ADD CONSTRAINT[FK\_ProductID]

FOREIGN KEY([ProductID])

REFERENCES [dbo].[Product] ([ProductID])

Скрипт создания базы данных второго филиала:

CREATE DATABASE Branch\_2

CREATE TABLE [dbo].[Customer](

[CustomerID] [int] PRIMARY KEY NOT NULL,

[CustomerName] [varchar](50) NOT NULL,

[DateAdded] [datetime] NOT NULL

)

CREATE TABLE [dbo].[Check](

[CheckID] [int] IDENTITY PRIMARY KEY NOT NULL,

[CustomerID] [int] NOT NULL,

[Date] [datetime] NOT NULL,

[DateAdded] [datetime] NOT NULL

)

CREATE TABLE [dbo].[Product](

[ProductID] [int] PRIMARY KEY NOT NULL,

[ProductName] [varchar](50) NOT NULL,

[Price] [money] NOT NULL,

[DateAdded] [datetime] NOT NULL

)

CREATE TABLE [dbo].[Specific](

[ProductID] [int] NOT NULL,

[CheckID] [int] NOT NULL,

[Price] [money] NOT NULL,

[Count] [int] NOT NULL,

[DateAdded] [datetime] NOT NULL

)

ALTER TABLE [dbo].[Specific]

ADD CONSTRAINT [PK\_CheckID\_ProductID]

PRIMARY KEY

CLUSTERED(CheckID,ProductID)

ALTER TABLE [dbo].[Check]

ADD CONSTRAINT [FK\_CustomerID]

FOREIGN KEY([CustomerID])

REFERENCES [dbo].[Customer] ([CustomerID])

ALTER TABLE [dbo].[Specific]

ADD CONSTRAINT [FK\_CheckID]

FOREIGN KEY([CheckID])

REFERENCES [dbo].[Check] ([CheckID])

ALTER TABLE [dbo].[Specific]

ADD CONSTRAINT[FK\_ProductID]

FOREIGN KEY([ProductID])

REFERENCES [dbo].[Product] ([ProductID])

Скрипт создания хранилища данных:

CREATE DATABASE Storage

CREATE TABLE [dbo].[Branch](

[BranchID] [int] IDENTITY PRIMARY KEY NOT NULL,

[BranchName] [varchar](50) NOT NULL

)

CREATE TABLE [dbo].[Customer](

[CustomerID] [int] PRIMARY KEY NOT NULL,

[CustomerName] [varchar](50) NOT NULL

)

CREATE TABLE [dbo].[Order](

[OrderDataWarehouseID] [int] IDENTITY PRIMARY KEY NOT NULL,

[OrderSystemID] [int] NOT NULL,  
 [DatePurchase] [datetime] NOT NULL

)

CREATE TABLE [dbo].[Product](

[ProductID] [int] PRIMARY KEY NOT NULL,

[ProductName] [varchar](50) NOT NULL

)

CREATE TABLE [dbo].[TimePurchase](

[TimePurchaseID] [int] IDENTITY PRIMARY KEY NOT NULL,

Year smallint NULL,

Month tinyint NULL,

Day tinyint NULL,

[DatePurchase] [datetime] NOT NULL

)

CREATE TABLE [dbo].[Synchronizes](

[SynchronizeID] [int] IDENTITY PRIMARY KEY NOT NULL,

[DateSynchron] [datetime] NOT NULL

)

CREATE TABLE [dbo].[Storage](

[ProductID] [int] NOT NULL,

[CustomerID] [int] NOT NULL,

[OrderDataWarehouseID] [int] NOT NULL,

[BranchID] [int] NOT NULL,

[TimePurchaseID] [int] NOT NULL,

[SynchronizeID] [int] NOT NULL,

[PriceFix] [money] NOT NULL,

[PricePurchase] [money] NOT NULL,

[Count] [int] NOT NULL

)

ALTER TABLE [dbo].[Storage]

ADD CONSTRAINT [FK\_Storage\_BranchID]

FOREIGN KEY([BranchID])

REFERENCES [dbo].[Branch] ([BranchID])

ALTER TABLE [dbo].[Storage]

ADD CONSTRAINT [FK\_Storage\_CustomerID]

FOREIGN KEY([CustomerID])

REFERENCES [dbo].[Customer] ([CustomerID])

ALTER TABLE [dbo].[Storage]

ADD CONSTRAINT [FK\_Storage\_ SynchronizeID]

FOREIGN KEY([SynchronizeID])

REFERENCES [dbo].[Synchronizes] ([SynchronizeID])

ALTER TABLE [dbo].[Storage]

ADD CONSTRAINT [FK\_Storage\_OrderDataWarehouseID]

FOREIGN KEY([OrderDataWarehouseID])

REFERENCES [dbo].[Order] ([OrderDataWarehouseID])

ALTER TABLE [dbo].[Storage]

ADD CONSTRAINT [FK\_Storage\_ProductID]

FOREIGN KEY([ProductID])

REFERENCES [dbo].[Product] ([ProductID])

ALTER TABLE [dbo].[Storage]

ADD CONSTRAINT [FK\_Storage\_TimePurchaseID]

FOREIGN KEY([TimePurchaseID])

REFERENCES [dbo].[TimePurchase] ([TimePurchaseID])

*Шаг 3. Заполнение баз данных филиалов с помощью запросов.*

USE Branch\_1

INSERT INTO Product (ProductID, ProductName, Price, DateAdded) VALUES (1, 'Yogurt', 65, getdate()), (2, 'Juice', 45, getdate()), (3, 'Milk', 70, getdate())

INSERT INTO Customer (CustomerID, CustomerName, DateAdded) VALUES (1, 'Misha', getdate()), (2,'Masha', getdate()), (3, 'Lena', getdate())

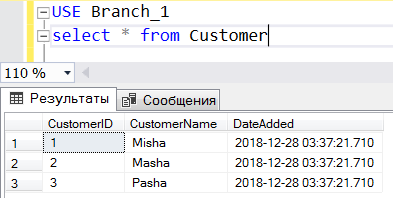
INSERT INTO [dbo].[Check] (CustomerID, Date, DateAdded) VALUES

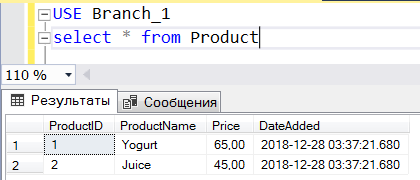
(1, '10-10-2018 10:15:17', getdate()), (2, '10-10-2018 10:20:00', getdate()), (2, '11-10-2018 10:20:00', getdate())

INSERT INTO Specific (ProductID, CheckID, Price, Count, DateAdded) VALUES (1, 1, 65, 2, getdate()),(1, 2, 65, 3, getdate()), (2, 2, 45, 1, getdate())

DELETE FROM Product WHERE Price > 68

UPDATE Customer SET CustomerName = 'Pasha' WHERE CustomerID = 3





USE Branch\_2

INSERT INTO Product (ProductID, ProductName, Price, DateAdded) VALUES (4, 'Apple', 65, getdate()), (5, 'Banana', 45, getdate())

INSERT INTO Customer (CustomerID, CustomerName, DateAdded) VALUES (4, 'Sasha', getdate()), (5,'Ivan', getdate())

INSERT INTO [dbo].[Check] (CustomerID, Date, DateAdded) VALUES

(4, '10-10-2018 19:15:17', getdate())

INSERT INTO Specific (ProductID, CheckID, Price, Count, DateAdded) VALUES (4, 1, 65, 3, getdate())

USE Storage

INSERT INTO Branch (BranchName) VALUES ('Branch\_1'), ('Branch\_2')

*Шаг 4 .Скрипт процедуры переноса данных из филиалов в хранилище.*

USE Storage

GO

CREATE PROCEDURE InsertFromBranche

AS

BEGIN

BEGIN

INSERT INTO Storage.dbo.Customer (CustomerID, CustomerName)

SELECT CustomerID, CustomerName

FROM Branch\_1.dbo.Customer

WHERE Branch\_1.dbo.Customer.DateAdded > (select MAX(DateSynchron) from Storage.dbo.Synchronizes)

UNION ALL

SELECT CustomerID, CustomerName

FROM Branch\_2.dbo.Customer

WHERE Branch\_2.dbo.Customer.DateAdded > (select MAX(DateSynchron) from Storage.dbo.Synchronizes)

END

BEGIN

INSERT INTO Storage.dbo.[Order](OrderSystemID, DatePurchase)

SELECT CheckID, Date

FROM Branch\_1.dbo.[Check]

WHERE Branch\_1.dbo.[Check].DateAdded > (select MAX(DateSynchron) from Storage.dbo.Synchronizes)

UNION ALL

SELECT CheckID, Date

FROM Branch\_2.dbo.[Check]

WHERE Branch\_2.dbo.[Check].DateAdded > (select MAX(DateSynchron) from Storage.dbo.Synchronizes)

END

BEGIN

INSERT INTO Storage.dbo.Product (ProductID, ProductName)

SELECT ProductID, ProductName

FROM Branch\_1.dbo.Product

WHERE Branch\_1.dbo.Product.DateAdded > (select MAX(DateSynchron) from Storage.dbo.Synchronizes)

UNION ALL

SELECT ProductID, ProductName

FROM Branch\_2.dbo.Product

WHERE Branch\_2.dbo.Product.DateAdded > (select MAX(DateSynchron) from Storage.dbo.Synchronizes)

END

BEGIN

INSERT INTO Storage.dbo.TimePurchase (Year, Month, Day, DatePurchase)

SELECT DISTINCT DATEPART(year, Date), DATEPART(month, Date), DATEPART(day, Date), Date

FROM Branch\_1.dbo.[Check]

WHERE Branch\_1.dbo.[Check].DateAdded > (select MAX(DateSynchron) from Storage.dbo.Synchronizes)

UNION ALL

SELECT DISTINCT DATEPART(year, Date), DATEPART(month, Date), DATEPART(day, Date), Date

FROM Branch\_2.dbo.[Check]

WHERE Branch\_2.dbo.[Check].DateAdded > (select MAX(DateSynchron) from Storage.dbo.Synchronizes)

END

INSERT INTO Synchronizes (DateSynchron) VALUES (getdate())

INSERT INTO [Storage].[dbo].[Storage] (

CustomerID, BranchID, ProductID, OrderDataWarehouseID, SynchronizeID, TimePurchaseID, Count, PriceFix, PricePurchase

)

SELECT

Cust.CustomerID,

1 as Branch,

Prod.ProductID,

(SELECT OrderDataWarehouseID FROM [Storage].[dbo].[Order] where DatePurchase=Ord.Date),

(SELECT MAX(SynchronizeID) FROM [Storage].[dbo].[Synchronizes])-1 as SynchronizeID,

(SELECT TimePurchaseID FROM [Storage].[dbo].[TimePurchase] where [Storage].[dbo].[TimePurchase].DatePurchase=Ord.Date),

Spec.Count,

Prod.Price,

Spec.Price

FROM

[Storage].[dbo].[TimePurchase] as TimePurchase,

[Storage].[dbo].[Synchronizes] as Syn,

[Branch\_1].[dbo].[Customer] as Cust

INNER JOIN [Branch\_1].[dbo].[Check] as Ord ON Cust.CustomerID=Ord.CustomerID

INNER JOIN [Branch\_1].[dbo].[Specific] as Spec ON Spec.[CheckID]=Ord.CheckID

INNER JOIN [Branch\_1].[dbo].[Product] as Prod ON Prod.ProductID=Spec.ProductID

WHERE

Ord.DateAdded > (select max(DateSynchron) from Synchronizes where SynchronizeID < (select MAX(SynchronizeID) from Synchronizes))

AND

Ord.CheckID IN (SELECT CheckID FROM [Branch\_1].[dbo].[Check])

INSERT INTO [Storage].[dbo].[Storage] (

CustomerID, BranchID, ProductID, OrderDataWarehouseID, SynchronizeID, TimePurchaseID, Count, PriceFix, PricePurchase

)

SELECT

Cust.CustomerID,

2 as Branch,

Prod.ProductID,

(SELECT OrderDataWarehouseID FROM [Storage].[dbo].[Order] where DatePurchase=Ord.Date),

(SELECT MAX(SynchronizeID) FROM [Storage].[dbo].[Synchronizes]) as SynchronizeID,

(SELECT TimePurchaseID FROM [Storage].[dbo].[TimePurchase] where [Storage].[dbo].[TimePurchase].DatePurchase=Ord.Date),

Spec.Count,

Prod.Price,

Spec.Price

FROM

[Storage].[dbo].[TimePurchase] as TimePurchase,

[Storage].[dbo].[Synchronizes] as Syn,

[Branch\_2].[dbo].[Customer] as Cust

INNER JOIN [Branch\_2].[dbo].[Check] as Ord ON Cust.CustomerID=Ord.CustomerID

INNER JOIN [Branch\_2].[dbo].[Specific] as Spec ON Spec.CheckID=Ord.CheckID

INNER JOIN [Branch\_2].[dbo].[Product] as Prod ON Prod.ProductID=Spec.ProductID

WHERE

Ord.DateAdded > (select max(DateSynchron) from Synchronizes where SynchronizeID < (select MAX(SynchronizeID) from Synchronizes))

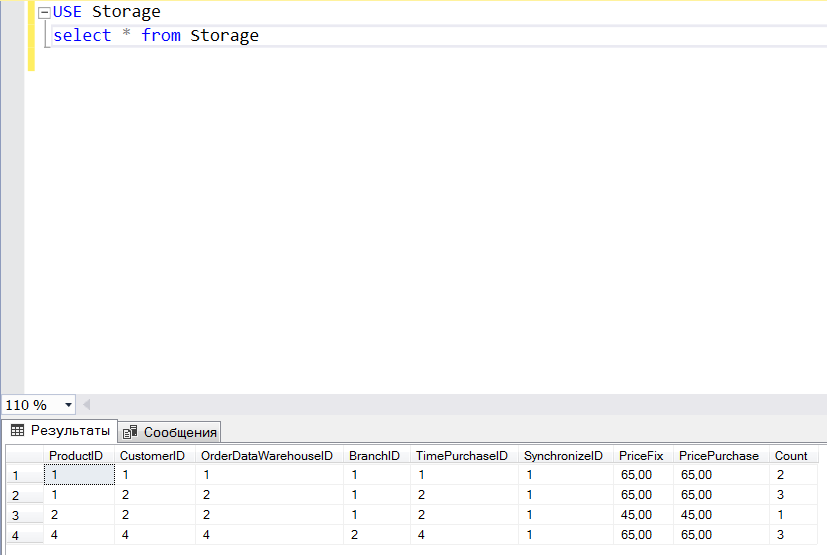
AND Ord.CheckID IN (SELECT CheckID FROM [Branch\_2].[dbo].[Check])

END

GO

EXECUTE InsertFromBranche

Все данные переносятся верно.



*Шаг 5. Приложение визуализации статистики продаж.*

Пользователь выбирает период, за который хочет увидеть статистику. Программа показывает диаграмму, на которой отображается количество совершенных покупок по датам, входящим в выбранный временной период.

Программа включает 3 класса: Connection – подключение к базе, Service – обработка запроса, Form – класс интерфейса приложения.

Код программы:

public class Connection

{

public SqlDataReader CollectData(string Date\_1, string Date\_2)

{

// Подключение к базе

*//Создаю строку запроса для подключения*

string connectString = "Data Source=Nastya;Initial Catalog=Storage;" + "Integrated Security=true;";

*//Создаю экземпляр подключения*

SqlConnection con;

con = new SqlConnection(connectString);

*//Запускаю подключение*

con.Open();

// Создание запроса

SqlCommand command;

SqlDataReader dataReader;

String sql = "";

*//Полученные из базы даты и время покупок перевожу в формат только дат, где время покупки чч:мм:сс не играет роли*

string date\_1 = "'" + Date\_1 + " 00:00:01 AM'";

string date\_2 = "'" + Date\_2 + " 11:59:59 PM'";

*//Запрос данных в базе. Возвращает разбитые на число, месяц и год даты и соответствующие им количества чеков*

sql = "select count(OrderDataWarehouseID) as Order, DATEPART(day, DatePurchase), DATEPART(month, DatePurchase), DATEPART(year, DatePurchase) from Order where DatePurchase > " + date\_1 + " and DatePurchase < " + date\_2 + " group by DATEPART(day, DatePurchase), DATEPART(month, DatePurchase), DATEPART(year, DatePurchase) order by DATEPART(year, DatePurchase), DATEPART(month, DatePurchase), DATEPART(day, DatePurchase)";

*//Создаю экземпляр SqlCommand, считываю данные*

command = new SqlCommand(sql, con);

dataReader = command.ExecuteReader();

return dataReader;

}

}

public class Service

{

//*Принимает начало и конец периода, за который нужно предоставить отчет*

public string[][] Data(string Date1, string Date2)

{

//*Создаю экземпляр класса подключения*

Connection dbc = new Connection();

SqlDataReader dataReader = dbc.CollectData(Date1,Date2);

String Output = "";

String Date\_S = "";

// Обработка запроса

*//Обработчик добавляет в строку Date\_S даты покупок из базы данных, в строку //Output – количество чеков в эти даты*

while (dataReader.Read())

{

Date\_S = Date\_S + dataReader.GetValue(1) + "." + dataReader.GetValue(2) +

"." + dataReader.GetValue(3) + "/";

Output = Output + dataReader.GetValue(0) + "/";

}

//*Создаю массивы из полученных строк с данными*

string[] Orders\_List = Output.Split('/');

string[] Date\_List = Date\_S.Split('/');

//*Объединяю массивы для передачи данных в форму*

string[][] Param = new string[2][];

Param[0] = Orders\_List;

Param[1] = Date\_List;

//*Передаю полученные данные в форму*

return Param;

}

}

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

public static string Date\_1;

public static string Date\_2;

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

{

//*Создаю экземпляр класса - обработчика*

Service serv = new Service();

//*Получение данных из базы*

string[][] data = serv.Data(Date\_1, Date\_2);

// Добавление данных в диаграмму

for (int i = 0; i < data[0].Length; i++)

{

for (int j = 0; j < data[1].Length; j++)

{

if (i == j)

{

this.chart1.Series["Order"].Points.AddXY(data[1][j], data[0][i]);

}

}

}

}

*//Для обозначения периода, за который нужно предоставить данные, добавила в форму календаря. Пользователь выбирает дату начала периода и дату окончания периода, данные поступают в обработчик*

public void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

Date\_1 = dateTimePicker1.Value.ToString("yyyyMMdd");

}

public void dateTimePicker2\_ValueChanged(object sender, EventArgs e)

{

Date\_2 = dateTimePicker2.Value.ToString("yyyyMMdd");

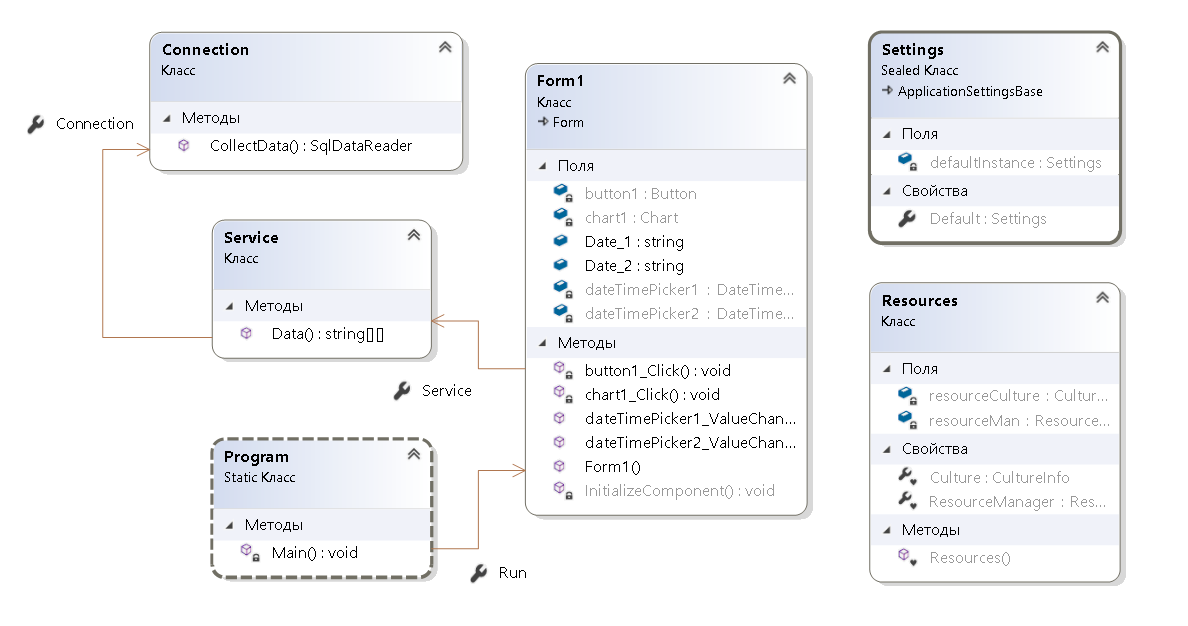
}

private void chart1\_Click(object sender, EventArgs e)

{

}

}



При запуске приложения открывается окно, содержащее диаграмму отображения отчета, календари для выбора периода отчета и кнопки для начала работы программы.

