Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования   
**Национальный исследовательский университет «МЭИ»**

**Типовой расчёт**по курсу«**Базы данных**»

Выполнил:

студент группы А-12-19

Винокуров Р.Н.

Вариант 6

Москва  
2021

1 – 3 Пункты были выполнены другим студентом:

1. **Выбрать актуальную тему: представить себя в качестве ИП. Вы наверняка столкнулись с проблемами в бизнесе т.к. у вас нет БД и клиент-серверного приложения.**

**Тема:** сеть салонов красоты.

1. **Описать бизнес процесс вашей компании (10-20 строк). Какую информацию нужно сохранить? Обработать? Удалить?**

**Бизнес-логика:**

Имеется информация о клиентах компании, оказываемых услугах и работающих в компании мастерах. Также БД хранит информацию об уже оказанных клиентам услугах, своих филиалах, и менеджерах данной компании.

Менеджеры должны иметь возможность создать заказ, изменить информацию о нем, а также иметь доступ к просмотру и редактированию остальной базы данных.

Клиенты имеют возможность занести информацию о себе в базу данных, просматривать список оказываемых сетью услуг, а также делать заказы.

Клиент может оформить заказ на оказание услуг. Но для этого сам клиент должен быть зарегистрирован, то есть занесен в базу данных компании. Мастер, который занимается данным видом услуг, берет заказ данного клиента. При этом заказ заносится в базу данных со статусом «запись». После оказания услуги ее статус изменяется с «запись» на «услуга оказана».

БД должна содержать следующую информацию:

о клиенте:

- полное ФИО;  
- дата рождения;  
- номер телефона;  
- адрес электронной почты;  
- даты и время всех посещений салона;  
- все процедуры, которые были предоставлены этому клиенту в определенный день;  
- мастер, работающий с клиентом по каждой из услуг в каждый из дней, когда клиент посещал салон;  
- цена каждой полученной услуги;  
- размер скидки; -логин для входа в аккаунт; -пароль для входа в аккаунт;

о мастере:

- список его компетенций;  
- список оконченных курсов;  
- стаж работы;  
- номер телефона;  
- дата заключения трудового договора с ним;  
- филиал, в котором он работает;  
- размер заработной платы;  
- дата выплаты аванса;  
- дата выплаты заработной платы.

о менеджере:

-его статус; -логин для входа в аккаунт; -пароль для входа в аккаунт;

1. **Кто будет работать с БД? Выделить как минимум 2 группы пользователей (admin или начальник, manager или обычный пользователь, какие атрибуты доступны для обработки той или иной группы?**

**Группы пользователей:**

Для данной БД должны быть предусмотрены 2 группы пользователей:

Клиент – имеет возможность добавить запись в базу данных клиентов, просматривать список оказываемых компанией услуг, а также записаться на оказание одной из услуг компании.

Менеджер – может редактировать все таблицы БД. При этом менеджеры со статусом <2 не имеют права добавлять и удалять менеджеров из базы данных, а также просматривать информацию об уже существующих менеджерах.

1. **Концептуальное проектирование. ER-диаграмма. Таблицы (не менее 3), индексы (обосновать), ключи (обосновать), связи, ограничения.**

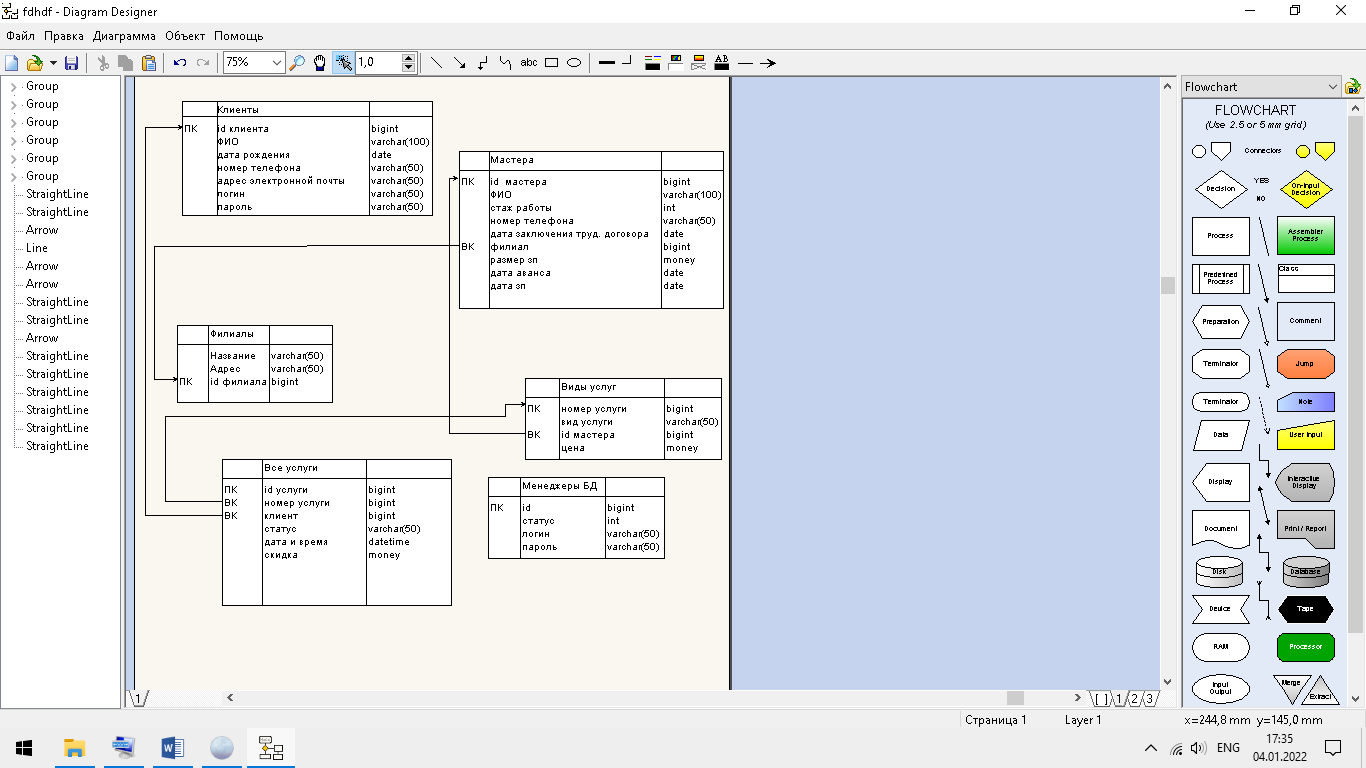


Рис.1 ER-диаграмма проектируемой БД

Все представленные таблицы содержат первичный ключ(ПК).

Рассмотрим каждую таблицу БД и обоснуем выбор ключей и индексов в ней.

Таблица **Клиенты**

В качестве первичного ключа данной таблицы выберем поле **id клиента**. Поле **ФИО** не подходит, так как содержит большое количество строк. Остальные поля могут дублироваться для разных пользователей, поэтому не могут являться первичным ключом.

Таблица **Мастера**

В качестве первичного ключа данной таблицы выберем поле **id мастера**. Поле **ФИО** не подходит, так как содержит большое количество строк. Остальные поля могут дублироваться для разных мастеров, поэтому не могут являться первичным ключом. Вторичный ключ данной таблицы – поле **филиал**, ссылающееся на поле **id филиала** из таблицы **Филиалы**.

Таблица **Филиалы**

В качестве первичного ключа данной таблицы выберем поле **id филиала**. Поля **адрес** и **название филиала** не подходят, так как содержат большое количество строк.

Таблица **Виды услуг**

В качестве первичного ключа данной таблицы выберем поле **номер услуги**. Поле **вид услуги** не подходит, так как содержит большое количество строк. Остальные поля могут дублироваться для разных видов услуг, поэтому не могут являться первичным ключом. Вторичный ключ данной таблицы – поле **id мастера**, ссылающееся на одноименное поле из таблицы **Мастера**.

Таблица **Все услуги**

В качестве первичного ключа данной таблицы выберем поле **id услуги**. Остальные поля могут дублироваться для разных видов услуг, поэтому не могут являться первичным ключом. Вторичные ключи данной таблицы – поле **номер услуги**, ссылающееся на одноименное поле таблицы **Виды услуг**, а также поле **клиент**, ссылающееся на поле **id клиента** из таблицы **Клиенты**.

Таблица **МенеджерыБД**

В качестве первичного ключа данной таблицы выберем поле **id**.Остальные поля могут дублироваться для разных менеджеров, поэтому не могут являться первичным ключом.

Так как при определении первичного ключа для таблицы автоматически создается кластеризованный индекс, в нашей БД будут иметься следующие кластеризованные индексы:

* Кластеризованный индекс для поля **id клиента** из таблицы **Клиенты**
* Кластеризованный индекс для поля **id мастера** из таблицы **Мастера**
* Кластеризованный индекс для поля **id филиала** из таблицы **Филиалы**
* Кластеризованный индекс для поля  **номер услуги** из таблицы **Филиалы**
* Кластеризованный индекс для поля  **id услуги** из таблицы **Все услуги**

Также добавим в нашу БД следующие некластеризованные индексы:

* Составной индекс на поля **вид услуги** и **цена** из таблицы **Виды услуг**
* Составной индекс на поля **id мастера, ФИО, филиал** из таблицы **Мастера**.

Ограничения:

Одному клиенту может оказываться множество услуг, но при этом одной оказанной услуге всегда соответствует один клиент.

Один мастер может оказывать множество услуг, но одна услуга может оказываться только одним мастером.

В одном филиале может работать множество мастеров, но один мастер может работать только в одном филиале.

1. **Получить физическую модель БД**

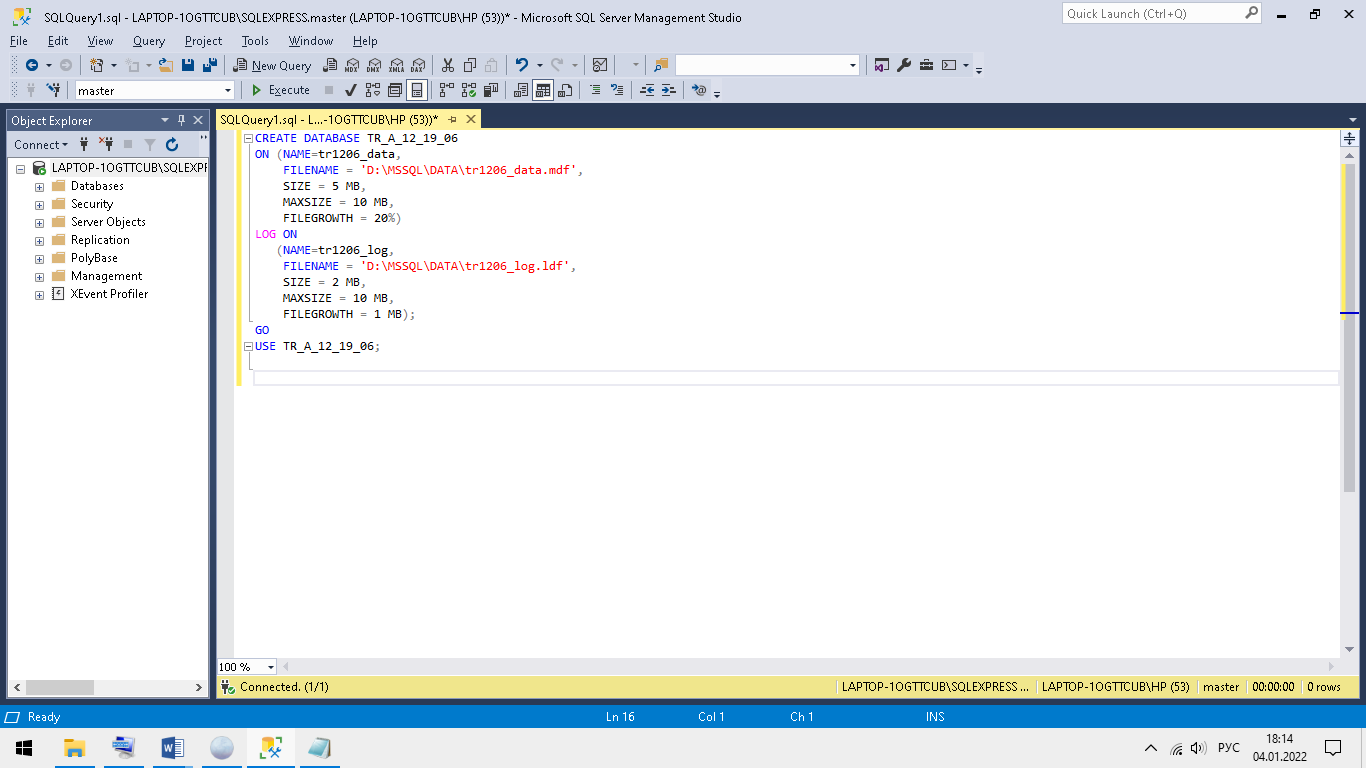


Рис.2 Создание физической модели БД

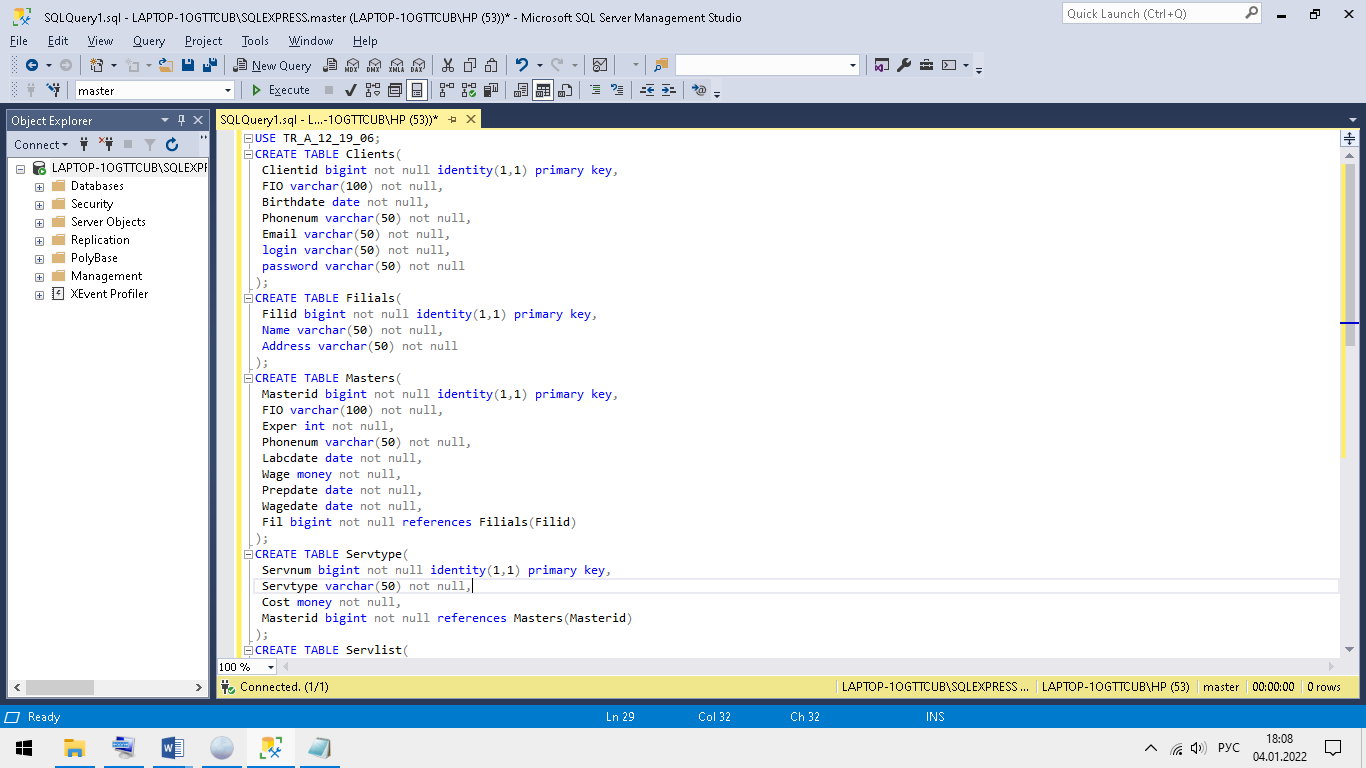


Рис. 3 Создание таблиц **Clients**, **Filials**, **Masters**

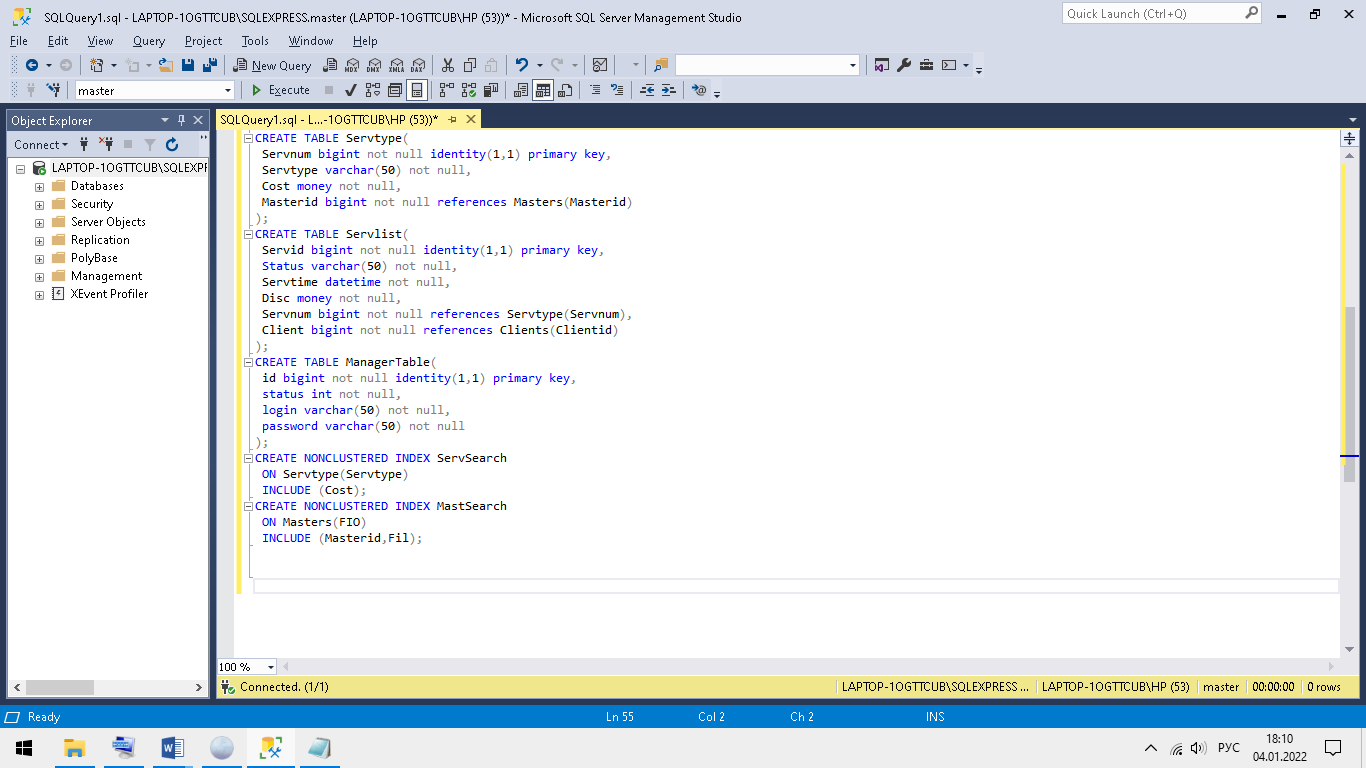


Рис.4 Создание таблиц **Servtype**, **Servlist**, **ManagerTable**, индексов **ServSearch**, **MastSearch**

1. **Создать 1 табличную функцию. 1 скалярную функцию. 1 представление**

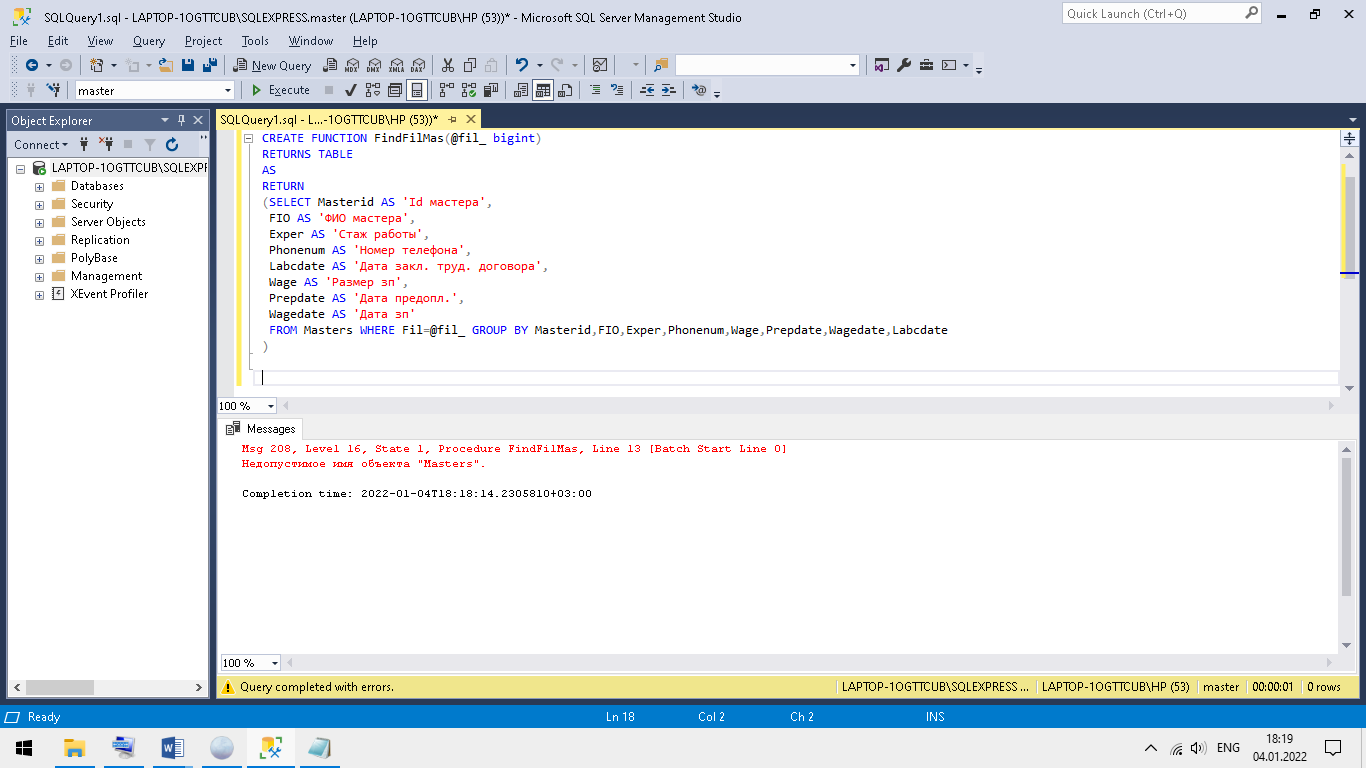


Рис.5 Создание табличной функции

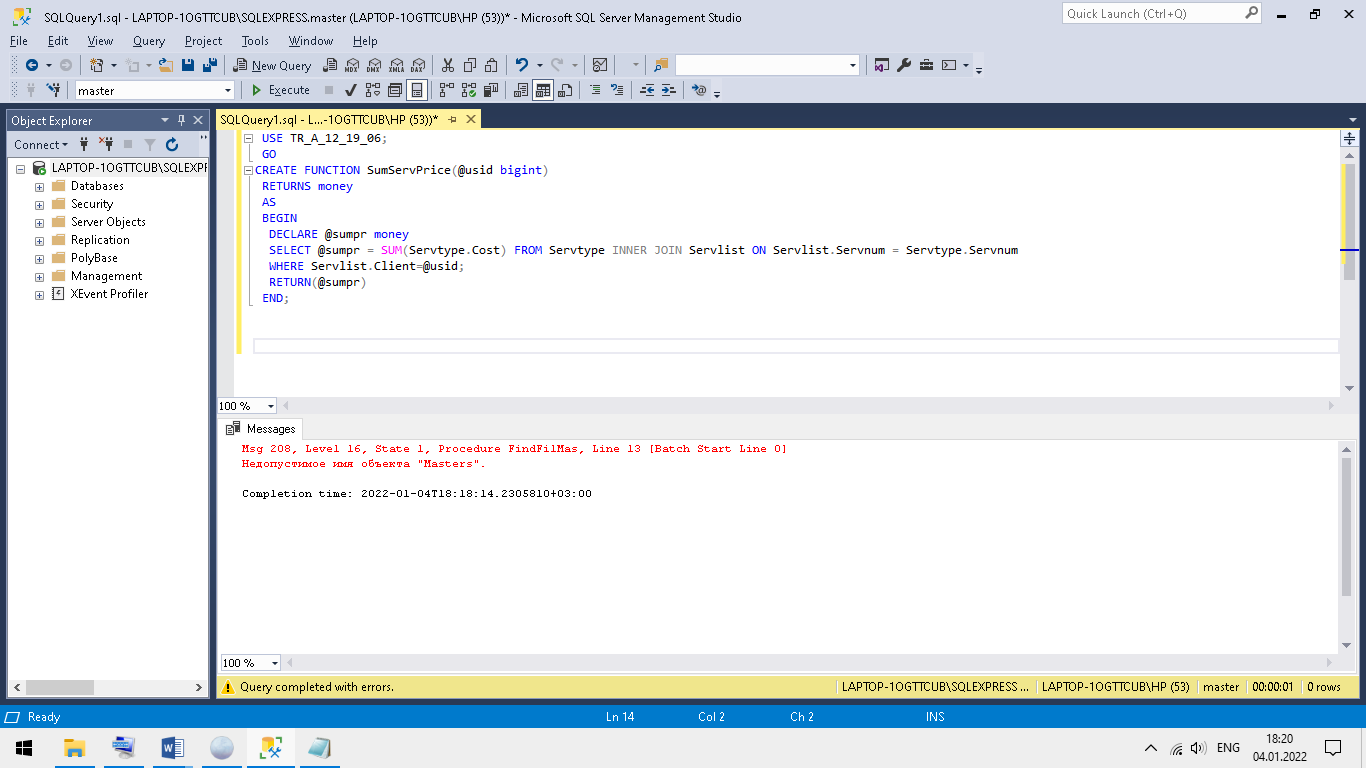


Рис.6 Создание скалярной функции

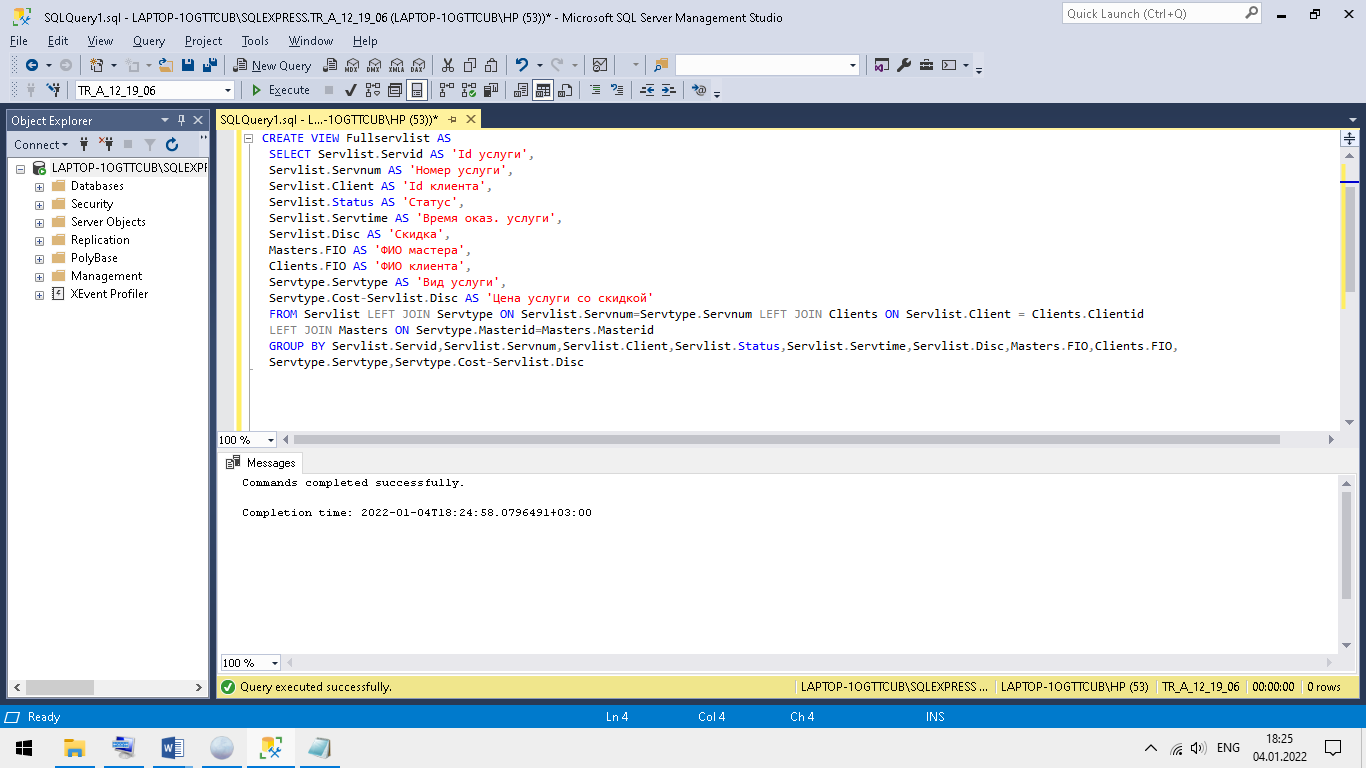


Рис.7 Создание представления

1. **Написать все необходимые процедуры для манипуляции с БД. Все действия INSERT UPDATE DELETE в виде процедур.**

**Таблица Клиенты:**

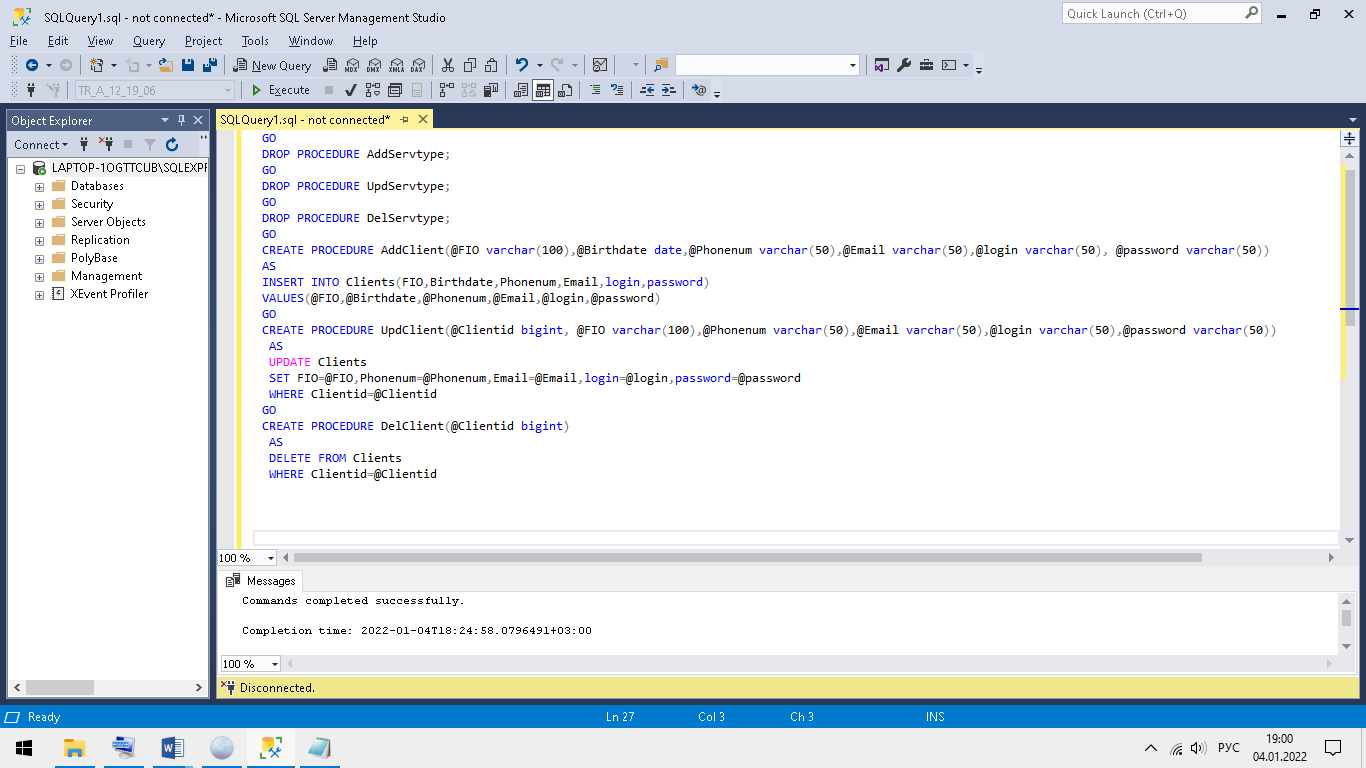


Рис.8 Создание процедур для таблицы Клиенты

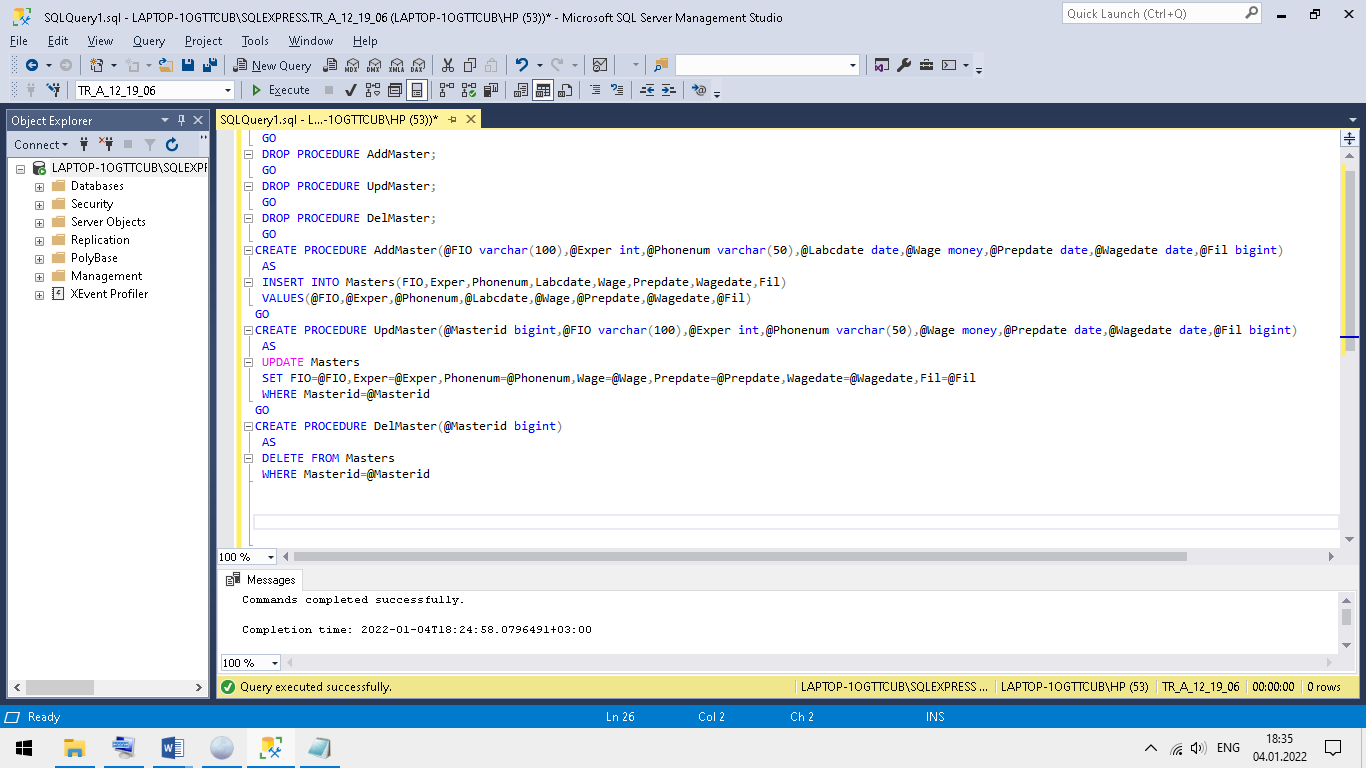
**Таблица Мастера:**  

Рис.9 Создание процедур для таблицы Мастера

**Таблица Список услуг:**

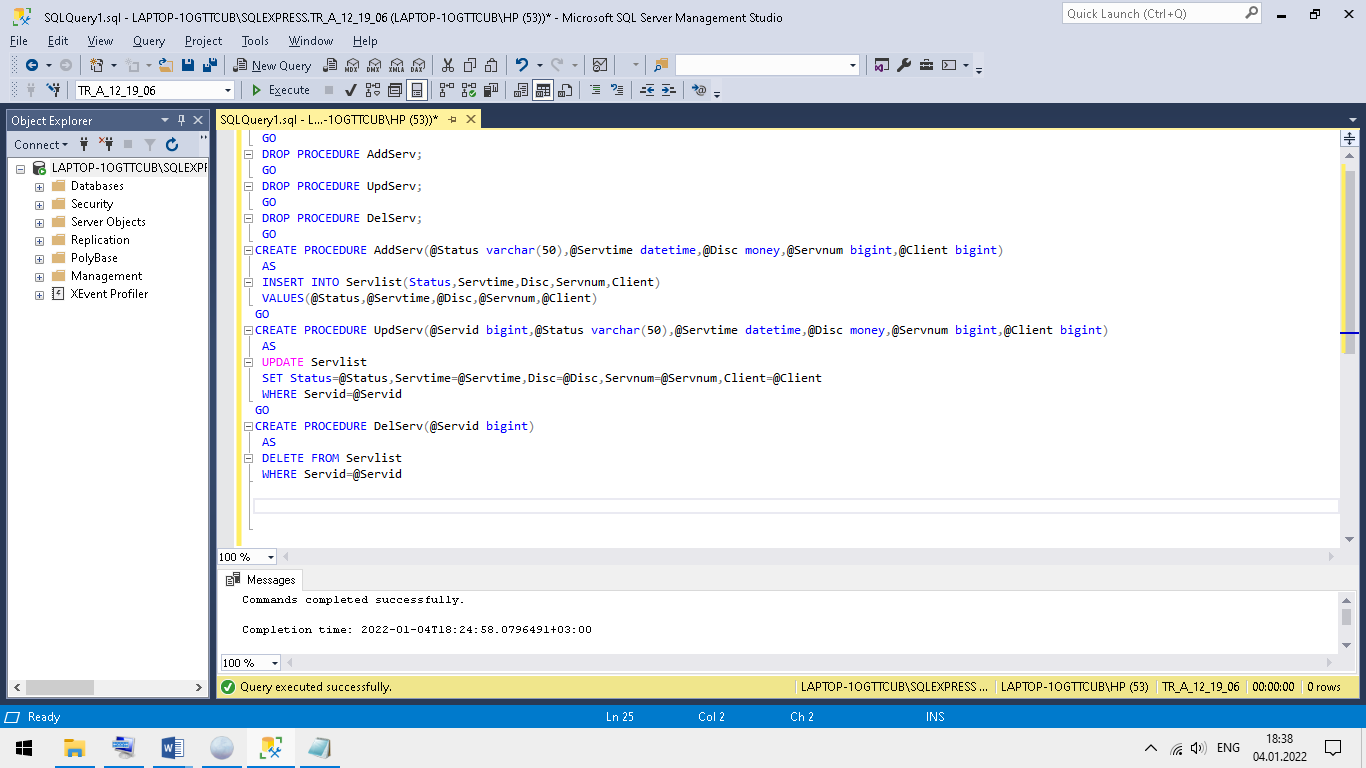


Рис.10 Создание процедур для таблицы Список услуг

**Таблица Виды услуг:**

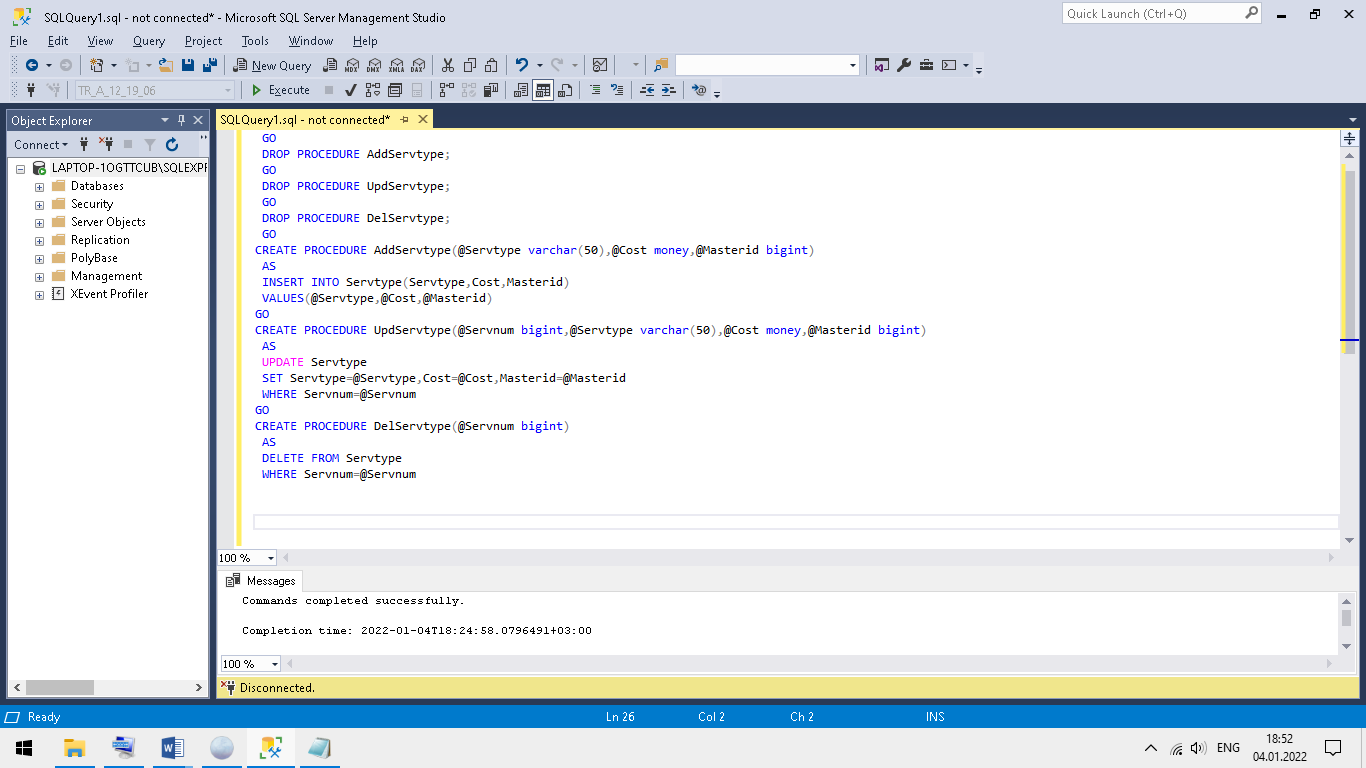


Рис.12 Создание процедур для таблицы Виды услуг

**Таблица Филиалы:**

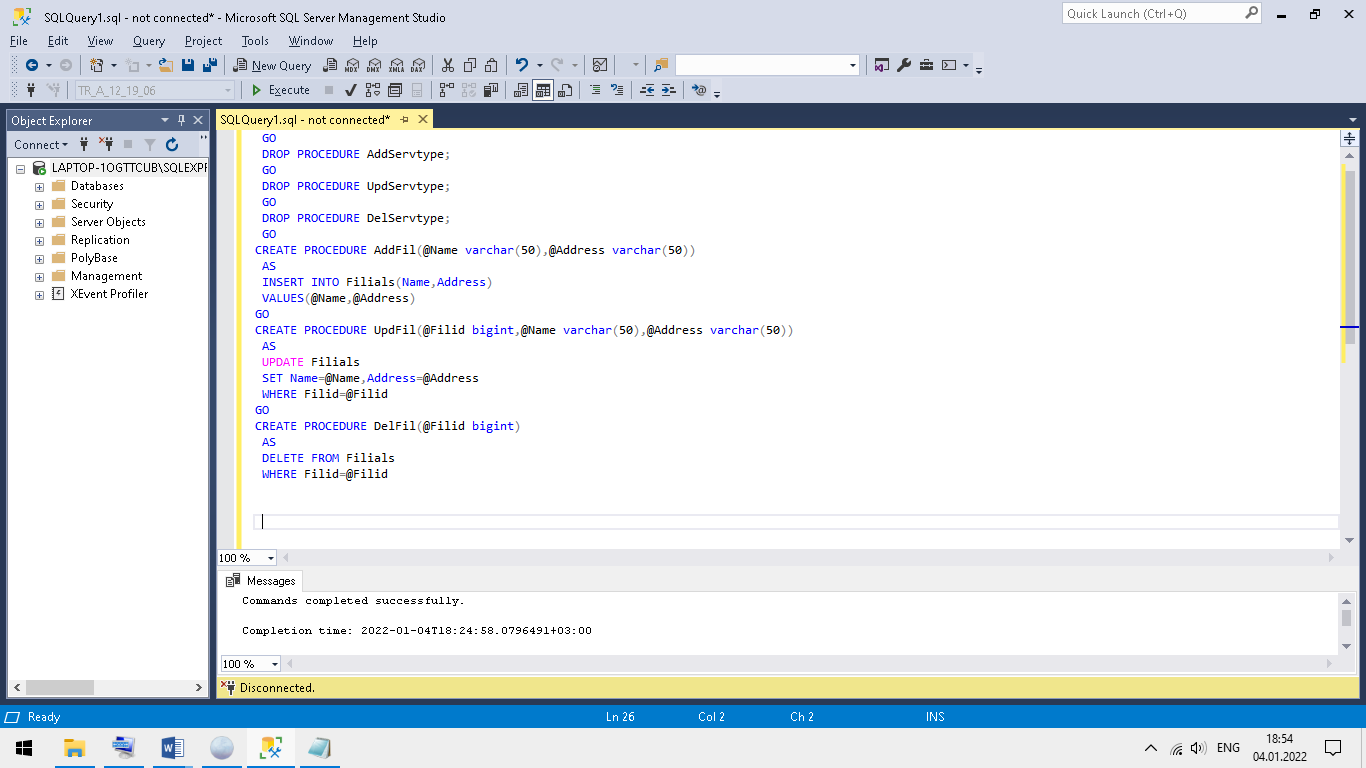


Рис.13 Создание процедур для таблицы Филиалы

**Таблица МенеджерыБД:**

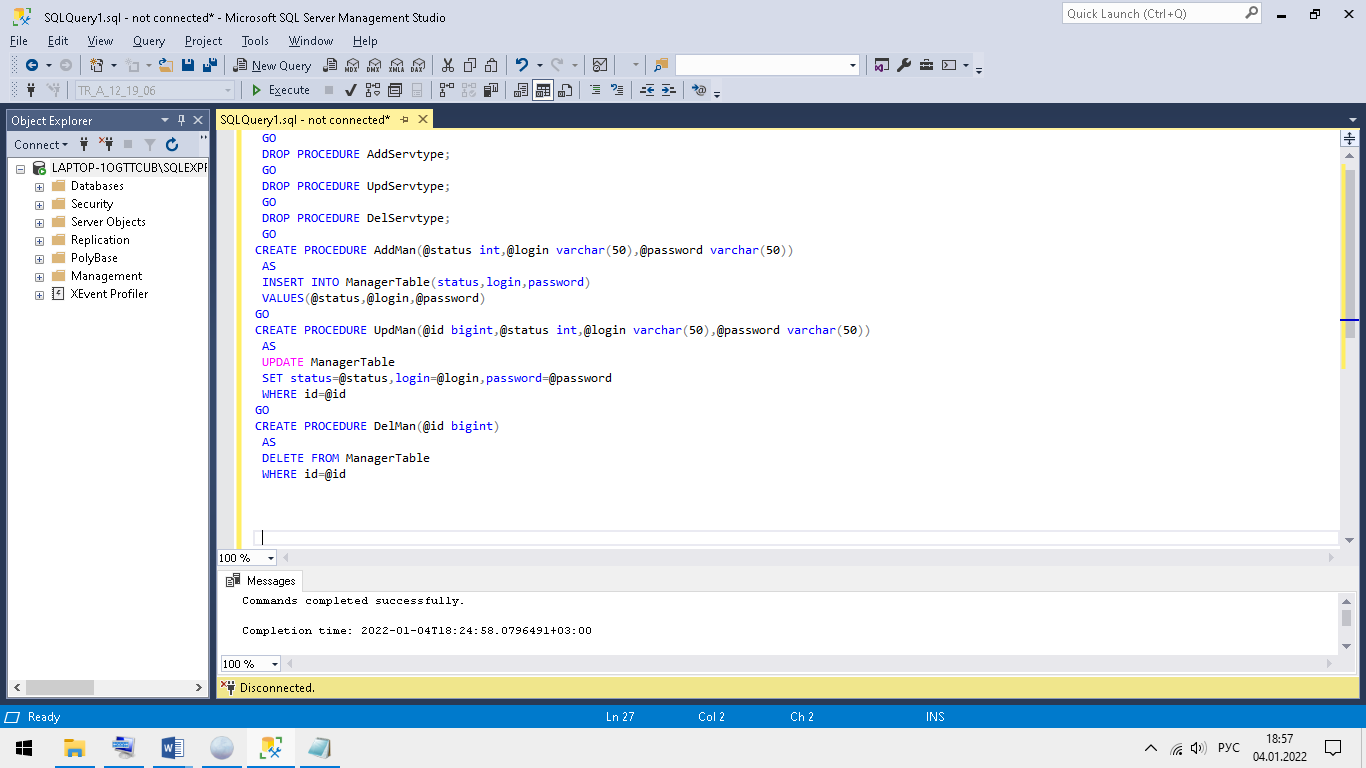


Рис.14 Создание процедур для таблицы МенеджерыБД

1. **Создать необходимые представления для визуализации данных в соответствии с пунктом 3.**

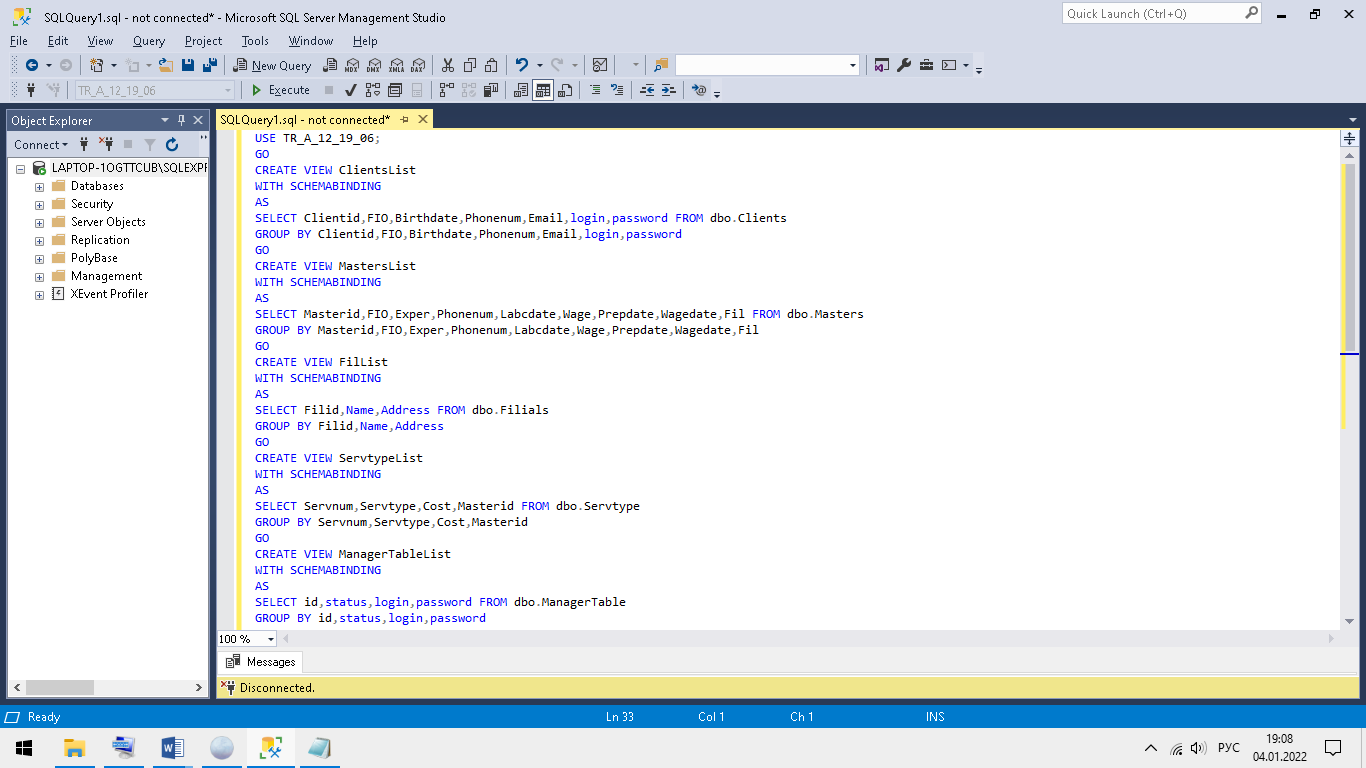


Рис.15 Создание представление для таблиц БД

1. **Создать собственную CLR функцию (MS SQL)**

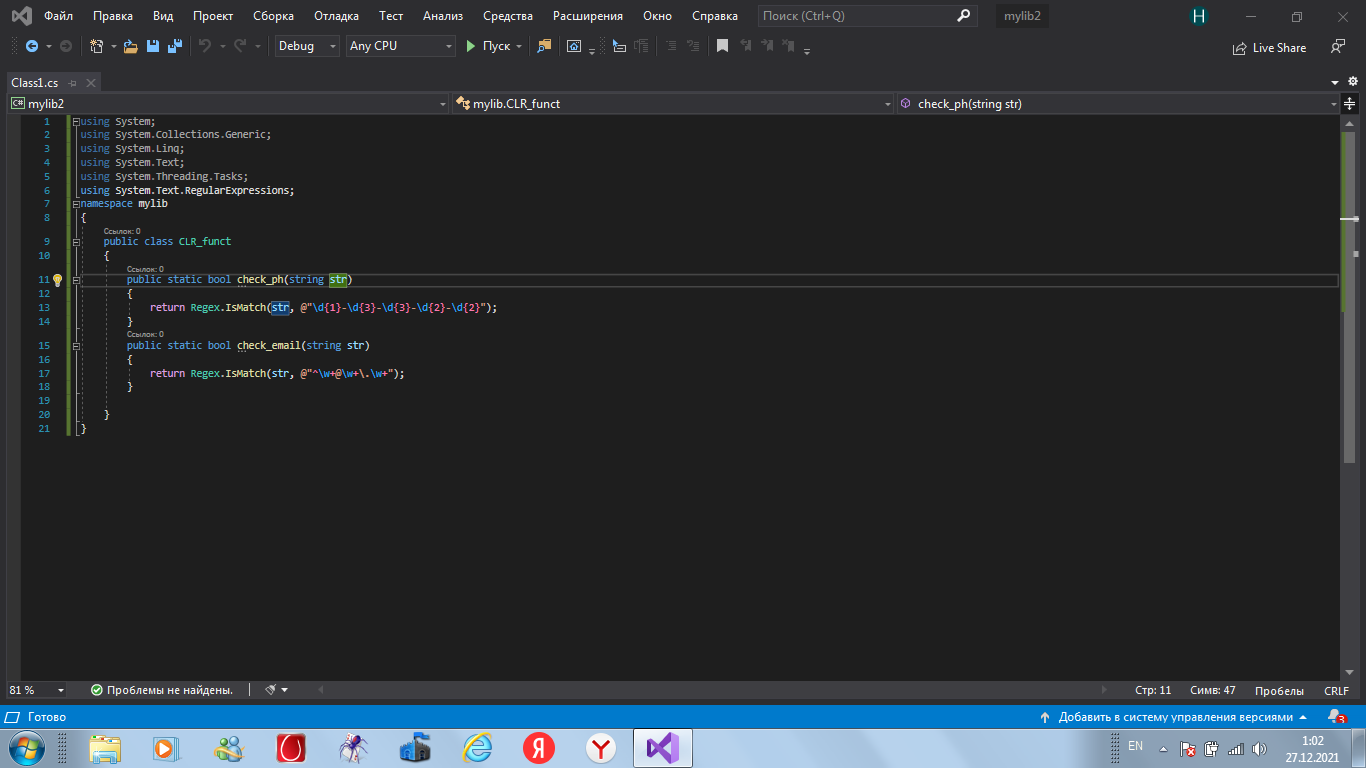


Рис.16 Создание библиотеки классов

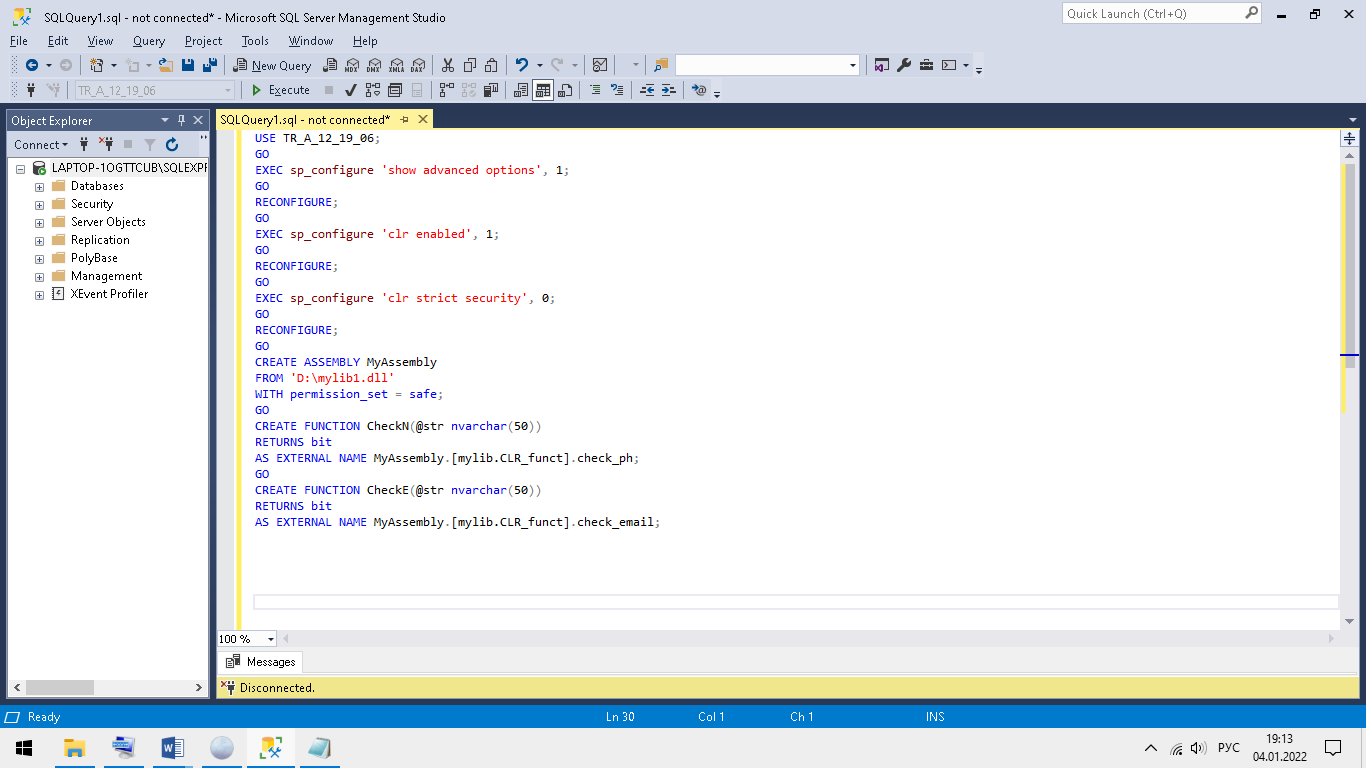


Рис.17 Подключение библиотеки классов

1. **Создать хотя бы 1 триггер на таблицу**

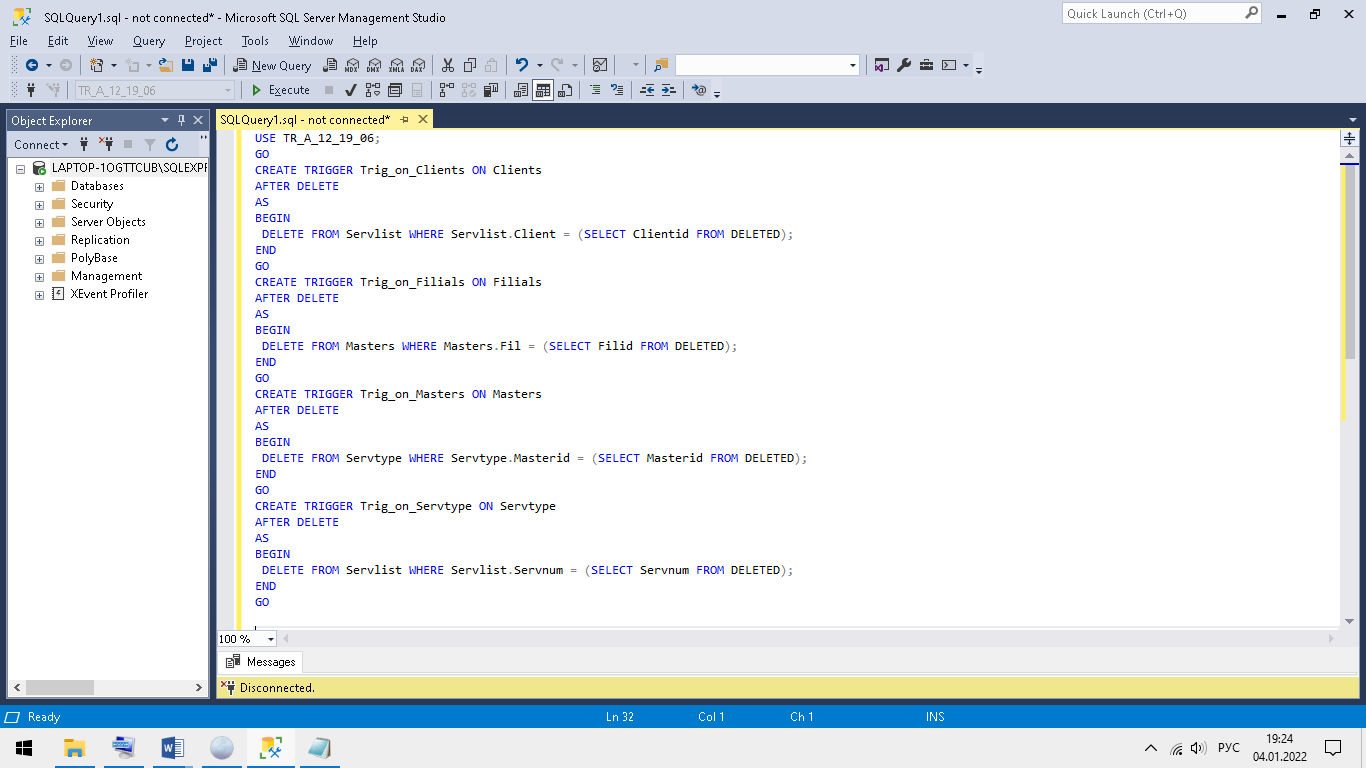


Рис.18 Создание триггеров для таблиц **Clients**,**Filials**,**Masters, Servtype**

1. **Создать клиент-северное приложение для работы с БД.**

Клиент-серверное приложение для работы с БД было написано при помощи технологии visual studio .NET Framework.

**Главное окно приложения (появляется при его открытии):**

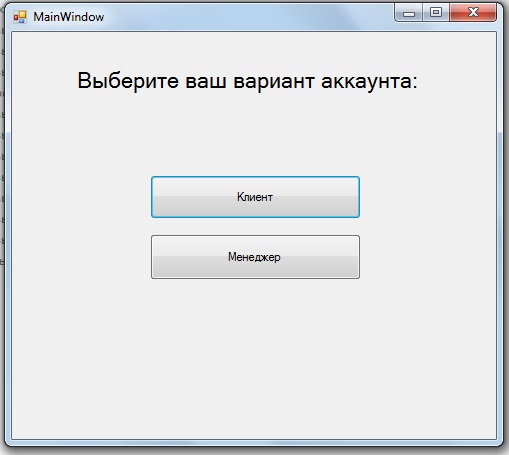
****

Рис.19 Основное окно клиент-серверного приложения

**Окно для входа клиента (нажатие «клиент» в предыдущем окне):**

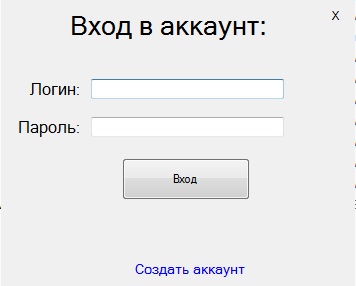


Рис. 20 Окно входа клиента

**Ошибка входа клиента:**

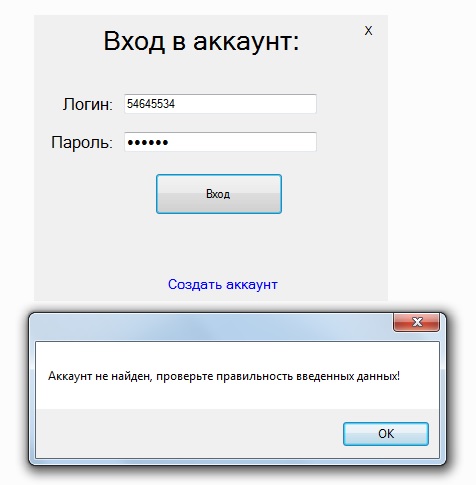


Рис. 21 Ошибка входа клиента

Подобную ошибку можно вызвать и при входе в аккаунт менеджера.

**Окно регистрации клиента (нажатие «создать аккаунт» в окне входа клиента):**

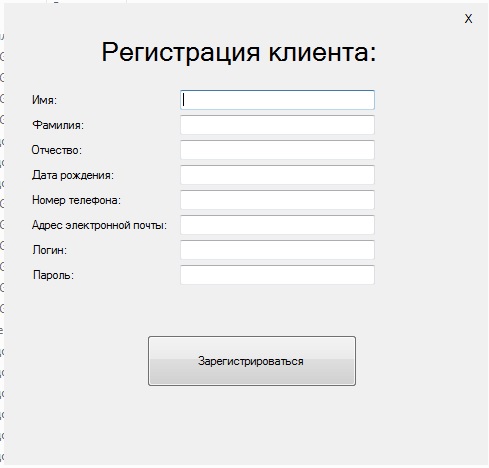


Рис. 22 Окно регистрации клиента

**Рабочее окно клиента (нажатие «вход» в окне входа клиента):**

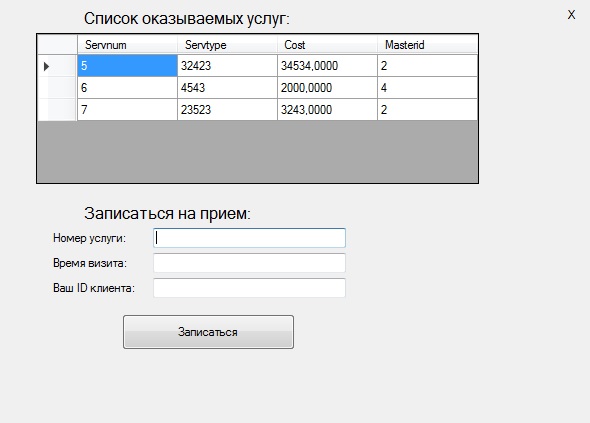
****

Рис. 23 Рабочее окно клиента

**Окно входа менеджера (нажатие «менеджер» в главном окне):**

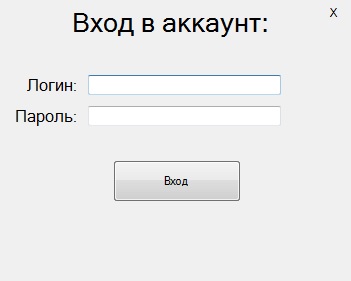


Рис. 24 Окно входа менеджера

**Рабочее окно менеджера (нажатие «вход» в предыдущем окне):**

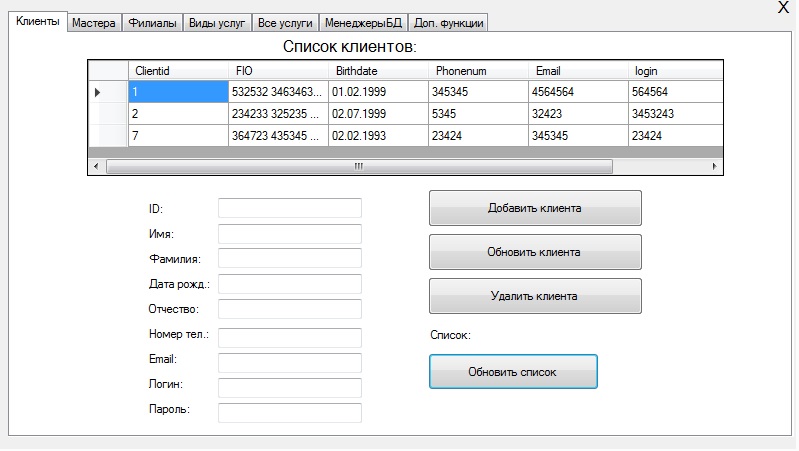


Рис. 25 Рабочее окно менеджера

Остальные вкладки данного окна (кроме «Доп. Функции») имеют похожий интерфейс: представление данных одной из таблиц БД и функционал для запуска функций, выполняющих редактирование данной таблицы.

**Ошибка при неполном заполнении необходимых полей:**

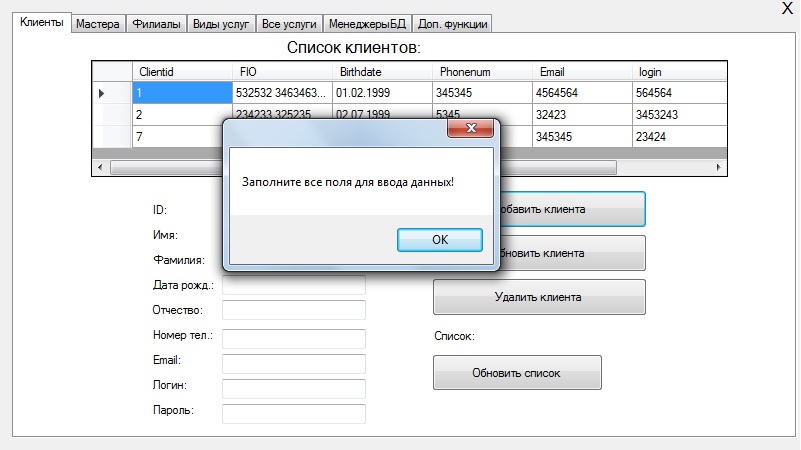


Рис. 26 Ошибка при неполном заполнении полей

**Ошибка при выполнении одной из функций редактирования БД:**

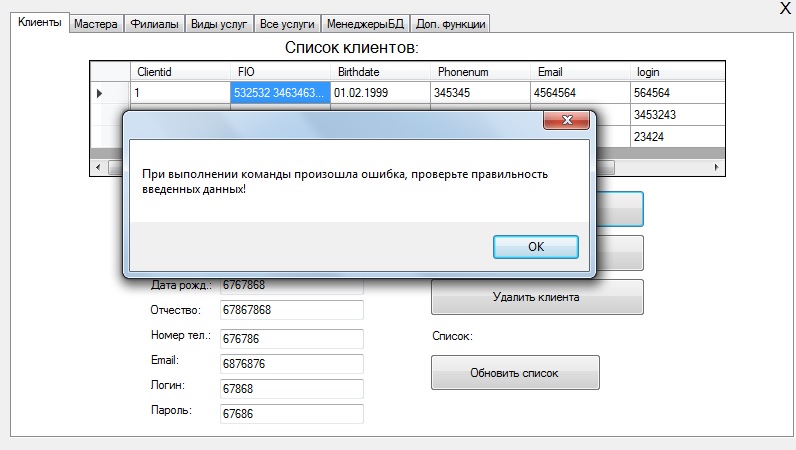


Рис. 27 Ошибка редактирования БД

**Вкладка для дополнительных функций работы с БД:**

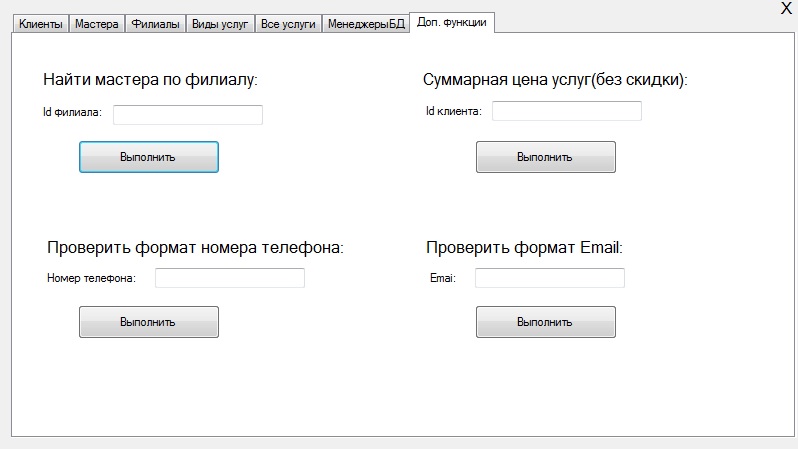


Рис. 28 Дополнительные функции для работы с БД

1. **Листинг запроса MS SQL для создания БД**

CREATE DATABASE TR\_A\_12\_19\_06

ON (NAME=tr1206\_data,

FILENAME = 'D:\MSSQL\DATA\tr1206\_data.mdf',

SIZE = 5 MB,

MAXSIZE = 10 MB,

FILEGROWTH = 20%)

LOG ON

(NAME=tr1206\_log,

FILENAME = 'D:\MSSQL\DATA\tr1206\_log.ldf',

SIZE = 2 MB,

MAXSIZE = 10 MB,

FILEGROWTH = 1 MB);

GO

USE TR\_A\_12\_19\_06;

CREATE TABLE Clients(

Clientid bigint not null identity(1,1) primary key,

FIO varchar(100) not null,

Birthdate date not null,

Phonenum varchar(50) not null,

Email varchar(50) not null,

login varchar(50) not null,

password varchar(50) not null

);

CREATE TABLE Filials(

Filid bigint not null identity(1,1) primary key,

Name varchar(50) not null,

Address varchar(50) not null

);

CREATE TABLE Masters(

Masterid bigint not null identity(1,1) primary key,

FIO varchar(100) not null,

Exper int not null,

Phonenum varchar(50) not null,

Labcdate date not null,

Wage money not null,

Prepdate date not null,

Wagedate date not null,

Fil bigint not null references Filials(Filid)

);

CREATE TABLE Servtype(

Servnum bigint not null identity(1,1) primary key,

Servtype varchar(50) not null,

Cost money not null,

Masterid bigint not null references Masters(Masterid)

);

CREATE TABLE Servlist(

Servid bigint not null identity(1,1) primary key,

Status varchar(50) not null,

Servtime datetime not null,

Disc money not null,

Servnum bigint not null references Servtype(Servnum),

Client bigint not null references Clients(Clientid)

);

CREATE TABLE ManagerTable(

id bigint not null identity(1,1) primary key,

status int not null,

login varchar(50) not null,

password varchar(50) not null

);

CREATE NONCLUSTERED INDEX ServSearch

ON Servtype(Servtype)

INCLUDE (Cost);

CREATE NONCLUSTERED INDEX MastSearch

ON Masters(FIO)

INCLUDE (Masterid,Fil);

GO

CREATE FUNCTION FindFilMas(@fil\_ bigint)

RETURNS TABLE

AS

RETURN

(SELECT Masterid AS 'Id мастера',

FIO AS 'ФИО мастера',

Exper AS 'Стаж работы',

Phonenum AS 'Номер телефона',

Labcdate AS 'Дата закл. труд. договора',

Wage AS 'Размер зп',

Prepdate AS 'Дата предопл.',

Wagedate AS 'Дата зп'

FROM Masters WHERE Fil=@fil\_ GROUP BY Masterid,FIO,Exper,Phonenum,Wage,Prepdate,Wagedate,Labcdate

)

GO

CREATE FUNCTION SumServPrice(@usid bigint)

RETURNS money

AS

BEGIN

DECLARE @sumpr money

SELECT @sumpr = SUM(Servtype.Cost) FROM Servtype INNER JOIN Servlist ON Servlist.Servnum = Servtype.Servnum

WHERE Servlist.Client=@usid;

RETURN(@sumpr)

END;

GO

CREATE VIEW Fullservlist AS

SELECT Servlist.Servid AS 'Id услуги',

Servlist.Servnum AS 'Номер услуги',

Servlist.Client AS 'Id клиента',

Servlist.Status AS 'Статус',

Servlist.Servtime AS 'Время оказ. услуги',

Servlist.Disc AS 'Скидка',

Masters.FIO AS 'ФИО мастера',

Clients.FIO AS 'ФИО клиента',

Servtype.Servtype AS 'Вид услуги',

Servtype.Cost-Servlist.Disc AS 'Цена услуги со скидкой'

FROM Servlist LEFT JOIN Servtype ON Servlist.Servnum=Servtype.Servnum LEFT JOIN Clients ON Servlist.Client = Clients.Clientid

LEFT JOIN Masters ON Servtype.Masterid=Masters.Masterid

GROUP BY Servlist.Servid,Servlist.Servnum,Servlist.Client,Servlist.Status,Servlist.Servtime,Servlist.Disc,Masters.FIO,Clients.FIO,

Servtype.Servtype,Servtype.Cost-Servlist.Disc

GO

CREATE PROCEDURE AddClient(@FIO varchar(100),@Birthdate date,@Phonenum varchar(50),@Email varchar(50),@login varchar(50), @password varchar(50))

AS

INSERT INTO Clients(FIO,Birthdate,Phonenum,Email,login,password)

VALUES(@FIO,@Birthdate,@Phonenum,@Email,@login,@password)

GO

CREATE PROCEDURE UpdClient(@Clientid bigint, @FIO varchar(100),@Phonenum varchar(50),@Email varchar(50),@login varchar(50),@password varchar(50))

AS

UPDATE Clients

SET FIO=@FIO,Phonenum=@Phonenum,Email=@Email,login=@login,password=@password

WHERE Clientid=@Clientid

GO

CREATE PROCEDURE DelClient(@Clientid bigint)

AS

DELETE FROM Clients

WHERE Clientid=@Clientid

GO

CREATE PROCEDURE AddMaster(@FIO varchar(100),@Exper int,@Phonenum varchar(50),@Labcdate date,@Wage money,@Prepdate date,@Wagedate date,@Fil bigint)

AS

INSERT INTO Masters(FIO,Exper,Phonenum,Labcdate,Wage,Prepdate,Wagedate,Fil)

VALUES(@FIO,@Exper,@Phonenum,@Labcdate,@Wage,@Prepdate,@Wagedate,@Fil)

GO

CREATE PROCEDURE UpdMaster(@Masterid bigint,@FIO varchar(100),@Exper int,@Phonenum varchar(50),@Wage money,@Prepdate date,@Wagedate date,@Fil bigint)

AS

UPDATE Masters

SET FIO=@FIO,Exper=@Exper,Phonenum=@Phonenum,Wage=@Wage,Prepdate=@Prepdate,Wagedate=@Wagedate,Fil=@Fil

WHERE Masterid=@Masterid

GO

CREATE PROCEDURE DelMaster(@Masterid bigint)

AS

DELETE FROM Masters

WHERE Masterid=@Masterid

GO

CREATE PROCEDURE AddServ(@Status varchar(50),@Servtime datetime,@Disc money,@Servnum bigint,@Client bigint)

AS

INSERT INTO Servlist(Status,Servtime,Disc,Servnum,Client)

VALUES(@Status,@Servtime,@Disc,@Servnum,@Client)

GO

CREATE PROCEDURE UpdServ(@Servid bigint,@Status varchar(50),@Servtime datetime,@Disc money,@Servnum bigint,@Client bigint)

AS

UPDATE Servlist

SET Status=@Status,Servtime=@Servtime,Disc=@Disc,Servnum=@Servnum,Client=@Client

WHERE Servid=@Servid

GO

CREATE PROCEDURE DelServ(@Servid bigint)

AS

DELETE FROM Servlist

WHERE Servid=@Servid

GO

CREATE PROCEDURE AddServtype(@Servtype varchar(50),@Cost money,@Masterid bigint)

AS

INSERT INTO Servtype(Servtype,Cost,Masterid)

VALUES(@Servtype,@Cost,@Masterid)

GO

CREATE PROCEDURE UpdServtype(@Servnum bigint,@Servtype varchar(50),@Cost money,@Masterid bigint)

AS

UPDATE Servtype

SET Servtype=@Servtype,Cost=@Cost,Masterid=@Masterid

WHERE Servnum=@Servnum

GO

CREATE PROCEDURE DelServtype(@Servnum bigint)

AS

DELETE FROM Servtype

WHERE Servnum=@Servnum

GO

CREATE PROCEDURE AddFil(@Name varchar(50),@Address varchar(50))

AS

INSERT INTO Filials(Name,Address)

VALUES(@Name,@Address)

GO

CREATE PROCEDURE UpdFil(@Filid bigint,@Name varchar(50),@Address varchar(50))

AS

UPDATE Filials

SET Name=@Name,Address=@Address

WHERE Filid=@Filid

GO

CREATE PROCEDURE DelFil(@Filid bigint)

AS

DELETE FROM Filials

WHERE Filid=@Filid

GO

CREATE PROCEDURE AddMan(@status int,@login varchar(50),@password varchar(50))

AS

INSERT INTO ManagerTable(status,login,password)

VALUES(@status,@login,@password)

GO

CREATE PROCEDURE UpdMan(@id bigint,@status int,@login varchar(50),@password varchar(50))

AS

UPDATE ManagerTable

SET status=@status,login=@login,password=@password

WHERE id=@id

GO

CREATE PROCEDURE DelMan(@id bigint)

AS

DELETE FROM ManagerTable

WHERE id=@id

GO

CREATE VIEW ClientsList

WITH SCHEMABINDING

AS

SELECT Clientid,FIO,Birthdate,Phonenum,Email,login,password FROM dbo.Clients

GROUP BY Clientid,FIO,Birthdate,Phonenum,Email,login,password

GO

CREATE VIEW MastersList

WITH SCHEMABINDING

AS

SELECT Masterid,FIO,Exper,Phonenum,Labcdate,Wage,Prepdate,Wagedate,Fil FROM dbo.Masters

GROUP BY Masterid,FIO,Exper,Phonenum,Labcdate,Wage,Prepdate,Wagedate,Fil

GO

CREATE VIEW FilList

WITH SCHEMABINDING

AS

SELECT Filid,Name,Address FROM dbo.Filials

GROUP BY Filid,Name,Address

GO

CREATE VIEW ServtypeList

WITH SCHEMABINDING

AS

SELECT Servnum,Servtype,Cost,Masterid FROM dbo.Servtype

GROUP BY Servnum,Servtype,Cost,Masterid

GO

CREATE VIEW ManagerTableList

WITH SCHEMABINDING

AS

SELECT id,status,login,password FROM dbo.ManagerTable

GROUP BY id,status,login,password

GO

EXEC sp\_configure 'show advanced options', 1;

GO

RECONFIGURE;

GO

EXEC sp\_configure 'clr enabled', 1;

GO

RECONFIGURE;

GO

EXEC sp\_configure 'clr strict security', 0;

GO

RECONFIGURE;

GO

CREATE ASSEMBLY MyAssembly

FROM 'D:\mylib1.dll'

WITH permission\_set = safe;

GO

CREATE FUNCTION CheckN(@str nvarchar(50))

RETURNS bit

AS EXTERNAL NAME MyAssembly.[mylib.CLR\_funct].check\_ph;

GO

CREATE FUNCTION CheckE(@str nvarchar(50))

RETURNS bit

AS EXTERNAL NAME MyAssembly.[mylib.CLR\_funct].check\_email;

GO

CREATE TRIGGER Trig\_on\_Clients ON Clients

AFTER DELETE

AS

BEGIN

DELETE FROM Servlist WHERE Servlist.Client = (SELECT Clientid FROM DELETED);

END

GO

CREATE TRIGGER Trig\_on\_Filials ON Filials

AFTER DELETE

AS

BEGIN

DELETE FROM Masters WHERE Masters.Fil = (SELECT Filid FROM DELETED);

END

GO

CREATE TRIGGER Trig\_on\_Masters ON Masters

AFTER DELETE

AS

BEGIN

DELETE FROM Servtype WHERE Servtype.Masterid = (SELECT Masterid FROM DELETED);

END

GO

CREATE TRIGGER Trig\_on\_Servtype ON Servtype

AFTER DELETE

AS

BEGIN

DELETE FROM Servlist WHERE Servlist.Servnum = (SELECT Servnum FROM DELETED);

END

1. **Листинг основной программы**

**App.config**

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<configSections>

</configSections>

<connectionStrings>

<add name="TRBDApp.Properties.Settings.MyConn" connectionString="Data Source=ОЛЬГА-ПК\SQLEXPRESS;Initial Catalog=TR\_A\_12\_19\_06;Integrated Security=True"

providerName="System.Data.SqlClient" />

</connectionStrings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.7.2" />

</startup>

</configuration>

**DBUtils.cs**

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConnectSQLServer

{

class DBUtils

{

public static SqlConnection GetDBConnection()

{

return DBSQLServerUtils.GetDBConnection();

}

}

}

**DBSQLServerUtils.cs**

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ConnectSQLServer

{

class DBSQLServerUtils

{

public static SqlConnection GetDBConnection()

{

string connString = ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;

SqlConnection conn = new SqlConnection(connString);

return conn;

}

}

}

**MainWindows.cs**

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;

namespace TRBDApp

{

public partial class MainWindow : Form

{

public MainWindow()

{

InitializeComponent();

}

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

{

this.Hide();

ClientEntry NewEntry = new ClientEntry();

NewEntry.Show();

}

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

{

this.Hide();

ManagerEntry newManagerEntry = new ManagerEntry();

newManagerEntry.Show();

}

}

}

**ClientEntry.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TRBDApp

{

public partial class ClientEntry : Form

{

public ClientEntry()

{

InitializeComponent();

}

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

{

this.Close();

ClientReg NewClientReg = new ClientReg();

NewClientReg.Show();

}

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

{

string sql = "SELECT login,password FROM Clients";

SqlConnection conn = DBUtils.GetDBConnection();

SqlCommand cmd = new SqlCommand();

conn.Open();

cmd.Connection = conn;

cmd.CommandText = sql;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (UserLogin.Text == "" || UserPassword.Text == "")

MessageBox.Show("Заполните все поля!");

else

{

if (reader.HasRows)

{

while (reader.Read())

{

int loginind = reader.GetOrdinal("login");

string loginstr = reader.GetString(loginind);

int passwordind = reader.GetOrdinal("password");

string passwordstr = reader.GetString(passwordind);

if (loginstr == UserLogin.Text && passwordstr == UserPassword.Text)

{

//---смена окна+выход из функции

this.Hide();

ClientWindow newClientWindow = new ClientWindow();

newClientWindow.Show();

conn.Close();

conn.Dispose();

conn = null;

return;

}

}

}

MessageBox.Show("Аккаунт не найден, проверьте правильность введенных данных!");

}

}

}

Point lastPoint;

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

{

Application.Exit();

}

private void ClientEntry\_MouseMove(object sender, MouseEventArgs e)

{

if(e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void ClientEntry\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

}

}

**ClientReg.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TRBDApp

{

public partial class ClientReg : Form

{

public ClientReg()

{

InitializeComponent();

}

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

{

bool a = ((ClientName.Text == "") || (ClientSurname.Text == "") || (ClientPatronymic.Text == "") || (ClientBirthdate.Text == "") || (ClientPhonenum.Text == "") || (ClientEmail.Text == "") || (ClientLogin.Text == "") || (ClientPassword.Text == ""));

string login = ClientLogin.Text;

string password = ClientPassword.Text;

SqlConnection conn = DBUtils.GetDBConnection();

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

try

{

int Clientid = 0;

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql = "INSERT INTO Clients (FIO,Birthdate,Phonenum,Email,login,password) " + " VALUES (@FIO,@Birthdate,@Phonenum,@Email,@login,@password)";

cmd.CommandText = sql;

SqlParameter FIOParam = new SqlParameter("@FIO", SqlDbType.VarChar);

FIOParam.Value = ClientName.Text + " " + ClientSurname.Text + " " + ClientPatronymic.Text;

cmd.Parameters.Add(FIOParam);

SqlParameter BirthdateParam = new SqlParameter("@Birthdate", SqlDbType.VarChar);

BirthdateParam.Value = ClientBirthdate.Text;

cmd.Parameters.Add(BirthdateParam);

SqlParameter PhonenumParam = new SqlParameter("@Phonenum", SqlDbType.VarChar);

PhonenumParam.Value = ClientPhonenum.Text;

cmd.Parameters.Add(PhonenumParam);

SqlParameter EmailParam = new SqlParameter("@Email", SqlDbType.VarChar);

EmailParam.Value = ClientEmail.Text;

cmd.Parameters.Add(EmailParam);

SqlParameter loginParam = new SqlParameter("@login", SqlDbType.VarChar);

loginParam.Value = ClientLogin.Text;

cmd.Parameters.Add(loginParam);

SqlParameter passwordParam = new SqlParameter("@password", SqlDbType.VarChar);

passwordParam.Value = ClientPassword.Text;

cmd.Parameters.Add(passwordParam);

cmd.ExecuteNonQuery();

string sql1 = "SELECT Clientid FROM Clients WHERE FIO=@FIO AND Birthdate=@Birthdate AND Phonenum=@Phonenum AND Email=@Email AND login=@login AND password=@password";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if(reader.HasRows)

{

while (reader.Read())

{

int Clientidind = reader.GetOrdinal("Clientid");

Clientid = (int)reader.GetInt64(Clientidind);

}

}

}

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Вы успешно зарегистрированы!\n" + "Ваш ID: " + Convert.ToString(Clientid));

}

catch

{

MessageBox.Show("При выполнении команды произошла ошибка!");

}

}

}

Point lastPoint;

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

{

Application.Exit();

}

private void ClientReg\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void ClientReg\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

}

}

**ClientWindow.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TRBDApp

{

public partial class ClientWindow : Form

{

public ClientWindow()

{

InitializeComponent();

}

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

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "lastVersDataSet.ServtypeList". При необходимости она может быть перемещена или удалена.

this.servtypeListTableAdapter1.Fill(this.lastVersDataSet.ServtypeList);

// TODO: данная строка кода позволяет загрузить данные в таблицу "tR\_A\_12\_19\_06DataSet1.ServtypeList". При необходимости она может быть перемещена или удалена.

this.servtypeListTableAdapter.Fill(this.tR\_A\_12\_19\_06DataSet1.ServtypeList);

// TODO: данная строка кода позволяет загрузить данные в таблицу "tR\_A\_12\_19\_06DataSet1.Servtype". При необходимости она может быть перемещена или удалена.

this.servtypeTableAdapter.Fill(this.tR\_A\_12\_19\_06DataSet1.Servtype);

}

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

{

this.Validate();

this.servtypeBindingSource.EndEdit();

this.tableAdapterManager.UpdateAll(this.tR\_A\_12\_19\_06DataSet1);

}

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

{

SqlConnection conn = DBUtils.GetDBConnection();

bool b = ((Servnum.Text == "") || (VisitTime.Text == "") || (ClientID.Text == ""));

if (b)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql = "INSERT INTO Servlist (Status,Servtime,Disc,Servnum,Client) "+"VALUES (@Status,@Servtime,@Disc,@Servnum,@Client)";

cmd.CommandText = sql;

SqlParameter StatusParam = new SqlParameter("@Status", SqlDbType.VarChar);

StatusParam.Value = "Запись";

cmd.Parameters.Add(StatusParam);

SqlParameter ServtimeParam = new SqlParameter("@Servtime", SqlDbType.DateTime);

ServtimeParam.Value = Convert.ToDateTime(VisitTime.Text);

cmd.Parameters.Add(ServtimeParam);

SqlParameter DiscParam = new SqlParameter("@Disc", SqlDbType.Int);

DiscParam.Value = 0;

cmd.Parameters.Add(DiscParam);

SqlParameter ServnumParam = new SqlParameter("@Servnum", SqlDbType.Int);

ServnumParam.Value = Convert.ToInt32(Servnum.Text);

cmd.Parameters.Add(ServnumParam);

SqlParameter ClientParam = new SqlParameter("@Client", SqlDbType.Int);

ClientParam.Value = Convert.ToInt32(ClientID.Text);

cmd.Parameters.Add(ClientParam);

cmd.ExecuteNonQuery();

string sql1 = "SELECT Name,Address FROM Filials LEFT JOIN Masters ON Filials.Filid = Masters.Fil LEFT JOIN Servtype ON Servtype.Masterid = Masters.Masterid WHERE Servnum=@Servn";

SqlParameter ServnParam = new SqlParameter("@Servn", SqlDbType.BigInt);

ServnParam.Value = Convert.ToInt32(Servnum.Text);

cmd.Parameters.Add(ServnParam);

cmd.CommandText = sql1;

string outputstr = "";

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Nameind = reader.GetOrdinal("Name");

string Name = reader.GetString(Nameind);

int Addressind = reader.GetOrdinal("Address");

string Address = reader.GetString(Addressind);

outputstr += Name + " " + Address;

}

}

}

MessageBox.Show("Регистрация прошла успешно!\nИнформация о филиале: "+outputstr);

}

catch

{

MessageBox.Show("При выполнении команды произошла ошибка!");

}

finally

{

conn.Close();

conn.Dispose();

conn = null;

}

}

}

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

{

Application.Exit();

}

Point lastPoint;

private void ClientWindow\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void ClientWindow\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

}

}

**ManagerEntry.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TRBDApp

{

public partial class ManagerEntry : Form

{

public ManagerEntry()

{

InitializeComponent();

}

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

{

string sql = "SELECT status,login,password FROM ManagerTable";

SqlConnection conn = DBUtils.GetDBConnection();

SqlCommand cmd = new SqlCommand();

conn.Open();

cmd.Connection = conn;

cmd.CommandText = sql;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (ManagerLogin.Text == "" || ManagerPassword.Text == "")

MessageBox.Show("Заполните все поля!");

else

{

if (reader.HasRows)

{

while (reader.Read())

{

int statusind = reader.GetOrdinal("status");

int loginind = reader.GetOrdinal("login");

int passwordind = reader.GetOrdinal("password");

CurrentUser.CurrentUserStatus = reader.GetInt32(statusind);

string loginstr = reader.GetString(loginind);

string passwordstr = reader.GetString(passwordind);

if (loginstr == ManagerLogin.Text && passwordstr == ManagerPassword.Text)

{

this.Hide();

ManagerWindow newManagerWindow = new ManagerWindow();

newManagerWindow.Show();

conn.Close();

conn.Dispose();

conn = null;

return;

}

}

}

MessageBox.Show("Аккаунт не найден, проверьте правильность введенных данных!");

}

}

}

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

{

Application.Exit();

}

Point lastPoint;

private void ManagerEntry\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void ManagerEntry\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

}

static class CurrentUser

{

public static int CurrentUserStatus = 0;

}

}

**ManagerWindow.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.Common;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace TRBDApp

{

public partial class ManagerWindow : Form

{

public ManagerWindow()

{

InitializeComponent();

}

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

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "lastVersDataSet.ManagerTableList". При необходимости она может быть перемещена или удалена.

this.managerTableListTableAdapter2.Fill(this.lastVersDataSet.ManagerTableList);

// TODO: данная строка кода позволяет загрузить данные в таблицу "lastVersDataSet.Fullservlist". При необходимости она может быть перемещена или удалена.

this.fullservlistTableAdapter.Fill(this.lastVersDataSet.Fullservlist);

// TODO: данная строка кода позволяет загрузить данные в таблицу "lastVersDataSet.ServtypeList". При необходимости она может быть перемещена или удалена.

this.servtypeListTableAdapter1.Fill(this.lastVersDataSet.ServtypeList);

// TODO: данная строка кода позволяет загрузить данные в таблицу "lastVersDataSet.FilList". При необходимости она может быть перемещена или удалена.

this.filListTableAdapter1.Fill(this.lastVersDataSet.FilList);

// TODO: данная строка кода позволяет загрузить данные в таблицу "lastVersDataSet.MastersList". При необходимости она может быть перемещена или удалена.

this.mastersListTableAdapter1.Fill(this.lastVersDataSet.MastersList);

// TODO: данная строка кода позволяет загрузить данные в таблицу "lastVersDataSet.ClientsList". При необходимости она может быть перемещена или удалена.

this.clientsListTableAdapter2.Fill(this.lastVersDataSet.ClientsList);

}

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

{

bool a = ClientName.Text == "" || ClientSurname.Text == "" || ClientPatronymic.Text == "" || ClientBirthdate.Text == "" || ClientPhonenum.Text == "" || ClientEmail.Text == "" || ClientLogin.Text == "" || ClientPassword.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql = "EXECUTE AddClient @FIO,@Birthdate,@Phonenum,@Email,@login,@password";

cmd.CommandText = sql;

SqlParameter FIOParam = new SqlParameter("@FIO", SqlDbType.VarChar);

FIOParam.Value = ClientName.Text + " " + ClientSurname.Text + " " + ClientPatronymic.Text;

cmd.Parameters.Add(FIOParam);

SqlParameter BirthdateParam = new SqlParameter("@Birthdate", SqlDbType.VarChar);

BirthdateParam.Value = ClientBirthdate.Text;

cmd.Parameters.Add(BirthdateParam);

SqlParameter PhonenumParam = new SqlParameter("@Phonenum", SqlDbType.VarChar);

PhonenumParam.Value = ClientPhonenum.Text;

cmd.Parameters.Add(PhonenumParam);

SqlParameter EmailParam = new SqlParameter("@Email", SqlDbType.VarChar);

EmailParam.Value = ClientEmail.Text;

cmd.Parameters.Add(EmailParam);

SqlParameter loginParam = new SqlParameter("@login", SqlDbType.VarChar);

loginParam.Value = ClientLogin.Text;

cmd.Parameters.Add(loginParam);

SqlParameter passwordParam = new SqlParameter("@password", SqlDbType.VarChar);

passwordParam.Value = ClientPassword.Text;

cmd.Parameters.Add(passwordParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Клиент успешно добавлен!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Clientid;

int flag = 0;

bool a = ClientID.Text=="" || ClientName.Text == "" || ClientSurname.Text == "" || ClientPatronymic.Text == "" || ClientBirthdate.Text == "" || ClientPhonenum.Text == "" || ClientEmail.Text == "" || ClientLogin.Text == "" || ClientPassword.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Clientid FROM Clients";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Clientidind = reader.GetOrdinal("Clientid");

Clientid = (int)reader.GetInt64(Clientidind);

if (Clientid == Convert.ToInt64(ClientID.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE UpdClient @ID,@FIO,@Phonenum,@Email,@login,@password";

cmd.CommandText = sql;

SqlParameter IDParam = new SqlParameter("@ID", SqlDbType.BigInt);

IDParam.Value = Convert.ToInt64(ClientID.Text);

cmd.Parameters.Add(IDParam);

SqlParameter FIOParam = new SqlParameter("@FIO", SqlDbType.VarChar);

FIOParam.Value = ClientName.Text + " " + ClientSurname.Text + " " + ClientPatronymic.Text;

cmd.Parameters.Add(FIOParam);

SqlParameter PhonenumParam = new SqlParameter("@Phonenum", SqlDbType.VarChar);

PhonenumParam.Value = ClientPhonenum.Text;

cmd.Parameters.Add(PhonenumParam);

SqlParameter EmailParam = new SqlParameter("@Email", SqlDbType.VarChar);

EmailParam.Value = ClientEmail.Text;

cmd.Parameters.Add(EmailParam);

SqlParameter loginParam = new SqlParameter("@login", SqlDbType.VarChar);

loginParam.Value = ClientLogin.Text;

cmd.Parameters.Add(loginParam);

SqlParameter passwordParam = new SqlParameter("@password", SqlDbType.VarChar);

passwordParam.Value = ClientPassword.Text;

cmd.Parameters.Add(passwordParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Данные клиента успешно обновлены!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Clientid;

int flag = 0;

SqlConnection conn = DBUtils.GetDBConnection();

bool a = ClientID.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Clientid FROM Clients";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Clientidind = reader.GetOrdinal("Clientid");

Clientid = (int)reader.GetInt64(Clientidind);

if (Clientid == Convert.ToInt64(ClientID.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE DelClient @ID";

cmd.CommandText = sql;

SqlParameter IDParam = new SqlParameter("@ID", SqlDbType.BigInt);

IDParam.Value = Convert.ToInt64(ClientID.Text);

cmd.Parameters.Add(IDParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Клиент удален!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

bool a = MasterName.Text == "" || MasterSurname.Text == "" || MasterPatronymic.Text == "" || MasterExp.Text == "" || MasterPhonenum.Text == "" || MasterLabcdate.Text == "" || MasterWage.Text == "" || MasterPrepdate.Text == "" || MasterWagedate.Text == "" || MasterFil.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql = "EXECUTE AddMaster @FIO,@Exper,@Phonenum,@Labcdate,@Wage,@Prepdate,@Wagedate,@Fil";

cmd.CommandText = sql;

SqlParameter FIOParam = new SqlParameter("@FIO", SqlDbType.VarChar);

FIOParam.Value = MasterName.Text + " " + MasterSurname.Text + " " + MasterPatronymic.Text;

cmd.Parameters.Add(FIOParam);

SqlParameter ExperParam = new SqlParameter("@Exper", SqlDbType.Int);

ExperParam.Value = Convert.ToInt32(MasterExp.Text);

cmd.Parameters.Add(ExperParam);

SqlParameter PhonenumParam = new SqlParameter("@Phonenum", SqlDbType.VarChar);

PhonenumParam.Value = MasterPhonenum.Text;

cmd.Parameters.Add(PhonenumParam);

SqlParameter LabcdateParam = new SqlParameter("@Labcdate", SqlDbType.VarChar);

LabcdateParam.Value = MasterLabcdate.Text;

cmd.Parameters.Add(LabcdateParam);

SqlParameter WageParam = new SqlParameter("@Wage", SqlDbType.Money);

WageParam.Value = Convert.ToInt32(MasterWage.Text);

cmd.Parameters.Add(WageParam);

SqlParameter PrepdateParam = new SqlParameter("@Prepdate", SqlDbType.VarChar);

PrepdateParam.Value = MasterPrepdate.Text;

cmd.Parameters.Add(PrepdateParam);

SqlParameter WagedateParam = new SqlParameter("@Wagedate", SqlDbType.VarChar);

WagedateParam.Value = MasterWagedate.Text;

cmd.Parameters.Add(WagedateParam);

SqlParameter FilParam = new SqlParameter("@Fil", SqlDbType.BigInt);

FilParam.Value = Convert.ToInt64(MasterFil.Text);

cmd.Parameters.Add(FilParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Мастер успешно добавлен!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Masterid;

int flag = 0;

bool a = MasterId.Text=="" || MasterName.Text == "" || MasterSurname.Text == "" || MasterPatronymic.Text == "" || MasterExp.Text == "" || MasterPhonenum.Text == "" || MasterLabcdate.Text == "" || MasterWage.Text == "" || MasterPrepdate.Text == "" || MasterWagedate.Text == "" || MasterFil.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Masterid FROM Masters";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Masterind = reader.GetOrdinal("Masterid");

Masterid = (int)reader.GetInt64(Masterind);

if (Masterid == Convert.ToInt64(MasterId.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE UpdMaster @Masterid,@FIO,@Exper,@Phonenum,@Wage,@Prepdate,@Wagedate,@Fil";

cmd.CommandText = sql;

SqlParameter MasteridParam = new SqlParameter("@Masterid", SqlDbType.BigInt);

MasteridParam.Value = Convert.ToInt64(MasterId.Text);

cmd.Parameters.Add(MasteridParam);

SqlParameter FIOParam = new SqlParameter("@FIO", SqlDbType.VarChar);

FIOParam.Value = MasterName.Text + " " + MasterSurname.Text + " " + MasterPatronymic.Text;

cmd.Parameters.Add(FIOParam);

SqlParameter ExperParam = new SqlParameter("@Exper", SqlDbType.Int);

ExperParam.Value = Convert.ToInt32(MasterExp.Text);

cmd.Parameters.Add(ExperParam);

SqlParameter PhonenumParam = new SqlParameter("@Phonenum", SqlDbType.VarChar);

PhonenumParam.Value = MasterPhonenum.Text;

cmd.Parameters.Add(PhonenumParam);

SqlParameter WageParam = new SqlParameter("@Wage", SqlDbType.Money);

WageParam.Value = Convert.ToInt32(MasterWage.Text);

cmd.Parameters.Add(WageParam);

SqlParameter PrepdateParam = new SqlParameter("@Prepdate", SqlDbType.VarChar);

PrepdateParam.Value = MasterPrepdate.Text;

cmd.Parameters.Add(PrepdateParam);

SqlParameter WagedateParam = new SqlParameter("@Wagedate", SqlDbType.VarChar);

WagedateParam.Value = MasterWagedate.Text;

cmd.Parameters.Add(WagedateParam);

SqlParameter FilParam = new SqlParameter("@Fil", SqlDbType.BigInt);

FilParam.Value = Convert.ToInt64(MasterFil.Text);

cmd.Parameters.Add(FilParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Данные успешно обновлены!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Masterid;

int flag = 0;

SqlConnection conn = DBUtils.GetDBConnection();

bool a = MasterId.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Masterid FROM Masters";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Masterind = reader.GetOrdinal("Masterid");

Masterid= (int)reader.GetInt64(Masterind);

if (Masterid == Convert.ToInt64(MasterId.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE DelMaster @ID";

cmd.CommandText = sql;

SqlParameter IDParam = new SqlParameter("@ID", SqlDbType.BigInt);

IDParam.Value = Convert.ToInt64(MasterId.Text);

cmd.Parameters.Add(IDParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Мастер удален!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

bool a = FilName.Text == "" || FilAddress.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql = "EXECUTE AddFil @Name,@Address";

cmd.CommandText = sql;

SqlParameter NameParam = new SqlParameter("@Name", SqlDbType.VarChar);

NameParam.Value = FilName.Text;

cmd.Parameters.Add(NameParam);

SqlParameter AddressParam = new SqlParameter("@Address", SqlDbType.VarChar);

AddressParam.Value = FilAddress.Text;

cmd.Parameters.Add(AddressParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Филиал успешно добавлен!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Filialid;

int flag = 0;

bool a = Filid.Text=="" || FilName.Text == "" || FilAddress.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Filid FROM Filials";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Filialind = reader.GetOrdinal("Filid");

Filialid = (int)reader.GetInt64(Filialind);

if (Filialid == Convert.ToInt64(Filid.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE UpdFil @Filid,@Name,@Address";

cmd.CommandText = sql;

SqlParameter FilidParam = new SqlParameter("@Filid", SqlDbType.BigInt);

FilidParam.Value = Convert.ToInt64(Filid.Text);

cmd.Parameters.Add(FilidParam);

SqlParameter NameParam = new SqlParameter("@Name", SqlDbType.VarChar);

NameParam.Value = FilName.Text;

cmd.Parameters.Add(NameParam);

SqlParameter AddressParam = new SqlParameter("@Address", SqlDbType.VarChar);

AddressParam.Value = FilAddress.Text;

cmd.Parameters.Add(AddressParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Данные успешно обновлены!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Filialid;

int flag = 0;

SqlConnection conn = DBUtils.GetDBConnection();

bool a = Filid.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Filid FROM Filials";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Filialind = reader.GetOrdinal("Filid");

Filialid = (int)reader.GetInt64(Filialind);

if (Filialid == Convert.ToInt64(Filid.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE DelFil @ID";

cmd.CommandText = sql;

SqlParameter IDParam = new SqlParameter("@ID", SqlDbType.BigInt);

IDParam.Value = Convert.ToInt64(Filid.Text);

cmd.Parameters.Add(IDParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Филиал удален!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

bool a = Servtype.Text == "" || Cost.Text == "" || Mastid.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql = "EXECUTE AddServtype @Servtype,@Cost,@Masterid";

cmd.CommandText = sql;

SqlParameter ServtypeParam = new SqlParameter("@Servtype", SqlDbType.VarChar);

ServtypeParam.Value = Servtype.Text;

cmd.Parameters.Add(ServtypeParam);

SqlParameter CostParam = new SqlParameter("@Cost", SqlDbType.Money);

CostParam.Value = Convert.ToInt32(Cost.Text);

cmd.Parameters.Add(CostParam);

SqlParameter MasteridParam = new SqlParameter("@Masterid", SqlDbType.BigInt);

MasteridParam.Value = Convert.ToInt64(Mastid.Text);

cmd.Parameters.Add(MasteridParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Вид услуг успешно добавлен!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Servnum;

int flag = 0;

bool a = ServtypeNum.Text == "" || Servtype.Text == "" || Cost.Text == "" || Mastid.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Servnum FROM Servtype";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Servnumind = reader.GetOrdinal("Servnum");

Servnum = (int)reader.GetInt64(Servnumind);

if (Servnum == Convert.ToInt64(ServtypeNum.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE UpdServtype @Servnum,@Servtype,@Cost,@Masterid";

cmd.CommandText = sql;

SqlParameter ServnumParam = new SqlParameter("@Servnum", SqlDbType.BigInt);

ServnumParam.Value = Convert.ToInt64(ServtypeNum.Text);

cmd.Parameters.Add(ServnumParam);

SqlParameter ServtypeParam = new SqlParameter("@Servtype", SqlDbType.VarChar);

ServtypeParam.Value = Servtype.Text;

cmd.Parameters.Add(ServtypeParam);

SqlParameter CostParam = new SqlParameter("@Cost", SqlDbType.Money);

CostParam.Value = Convert.ToInt32(Cost.Text);

cmd.Parameters.Add(CostParam);

SqlParameter MasteridParam = new SqlParameter("@Masterid", SqlDbType.BigInt);

MasteridParam.Value = Convert.ToInt64(Mastid.Text);

cmd.Parameters.Add(MasteridParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Данные успешно обновлены!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Servnum;

int flag = 0;

SqlConnection conn = DBUtils.GetDBConnection();

bool a = ServtypeNum.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Servnum FROM Servtype";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Servnumind = reader.GetOrdinal("Servnum");

Servnum = (int)reader.GetInt64(Servnumind);

if (Servnum == Convert.ToInt64(ServtypeNum.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE DelServtype @Servnum";

cmd.CommandText = sql;

SqlParameter ServnumParam = new SqlParameter("@Servnum", SqlDbType.BigInt);

ServnumParam.Value = Convert.ToInt64(ServtypeNum.Text);

cmd.Parameters.Add(ServnumParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Вид услуг удален!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

bool a = ServStatus.Text == "" || ServTime.Text == "" || ServDisc.Text == "" || ServNum.Text == "" || ServClientId.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql = "EXECUTE AddServ @Status,@Servtime,@Disc,@Servnum,@Client";

cmd.CommandText = sql;

SqlParameter StatusParam = new SqlParameter("@Status", SqlDbType.VarChar);

StatusParam.Value = ServStatus.Text;

cmd.Parameters.Add(StatusParam);

SqlParameter ServtimeParam = new SqlParameter("@Servtime", SqlDbType.DateTime);

ServtimeParam.Value = ServTime.Text;

cmd.Parameters.Add(ServtimeParam);

SqlParameter DiscParam = new SqlParameter("@Disc", SqlDbType.Money);

DiscParam.Value = Convert.ToInt32(ServDisc.Text);

cmd.Parameters.Add(DiscParam);

SqlParameter ServnumParam = new SqlParameter("@Servnum", SqlDbType.BigInt);

ServnumParam.Value = Convert.ToInt64(ServNum.Text);

cmd.Parameters.Add(ServnumParam);

SqlParameter ClientParam = new SqlParameter("@Client", SqlDbType.BigInt);

ClientParam.Value = Convert.ToInt64(ServClientId.Text);

cmd.Parameters.Add(ClientParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Вид услуг успешно добавлен!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Servnum1;

int flag = 0;

bool a = ServId.Text == "" || ServStatus.Text == "" || ServTime.Text == "" || ServDisc.Text == "" || ServNum.Text == "" || ServClientId.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Servid FROM Servlist";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Servidind = reader.GetOrdinal("Servid");

Servnum1 = (int)reader.GetInt64(Servidind);

if (Servnum1 == Convert.ToInt64(ServId.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE UpdServ @Servid,@Status,@Servtime,@Disc,@Servnum,@Client";

cmd.CommandText = sql;

SqlParameter ServidParam = new SqlParameter("@Servid", SqlDbType.BigInt);

ServidParam.Value = Convert.ToInt64(ServId.Text);

cmd.Parameters.Add(ServidParam);

SqlParameter StatusParam = new SqlParameter("@Status", SqlDbType.VarChar);

StatusParam.Value = ServStatus.Text;

cmd.Parameters.Add(StatusParam);

SqlParameter ServtimeParam = new SqlParameter("@Servtime", SqlDbType.DateTime);

ServtimeParam.Value = ServTime.Text;

cmd.Parameters.Add(ServtimeParam);

SqlParameter DiscParam = new SqlParameter("@Disc", SqlDbType.Money);

DiscParam.Value = Convert.ToInt32(ServDisc.Text);

cmd.Parameters.Add(DiscParam);

SqlParameter ServnumParam = new SqlParameter("@Servnum", SqlDbType.BigInt);

ServnumParam.Value = Convert.ToInt64(ServNum.Text);

cmd.Parameters.Add(ServnumParam);

SqlParameter ClientParam = new SqlParameter("@Client", SqlDbType.BigInt);

ClientParam.Value = Convert.ToInt64(ServClientId.Text);

cmd.Parameters.Add(ClientParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Данные успешно обновлены!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

int Servnum1;

int flag = 0;

SqlConnection conn = DBUtils.GetDBConnection();

bool a = ServId.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT Servid FROM Servlist";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int Servidind = reader.GetOrdinal("Servid");

Servnum1 = (int)reader.GetInt64(Servidind);

if (Servnum1 == Convert.ToInt64(ServId.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE DelServ @Servid";

cmd.CommandText = sql;

SqlParameter ServidParam = new SqlParameter("@Servid", SqlDbType.BigInt);

ServidParam.Value = Convert.ToInt64(ServId.Text);

cmd.Parameters.Add(ServidParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Услуга удалена!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

if (CurrentUser.CurrentUserStatus > 1)

{

bool a = ManagerStatus.Text == "" || ManagerLogin.Text == "" || ManagerPassword.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql = "EXECUTE AddMan @status,@login,@password";

cmd.CommandText = sql;

SqlParameter statusParam = new SqlParameter("@status", SqlDbType.Int);

statusParam.Value = Convert.ToInt32(ManagerStatus.Text);

cmd.Parameters.Add(statusParam);

SqlParameter loginParam = new SqlParameter("@login", SqlDbType.VarChar);

loginParam.Value = ManagerLogin.Text;

cmd.Parameters.Add(loginParam);

SqlParameter passwordParam = new SqlParameter("@password", SqlDbType.VarChar);

passwordParam.Value = ManagerPassword.Text;

cmd.Parameters.Add(passwordParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Менеджер успешно добавлен!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

else

MessageBox.Show("Недостаточно прав для вызова данной функции! Обратитесь к администраторам с status>=2");

}

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

{

if (CurrentUser.CurrentUserStatus > 1)

{

int Managerid;

int flag = 0;

bool a = ManagerId.Text == "" || ManagerStatus.Text == "" || ManagerLogin.Text == "" || ManagerPassword.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

SqlConnection conn = DBUtils.GetDBConnection();

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT id FROM ManagerTable";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int idind = reader.GetOrdinal("id");

Managerid = (int)reader.GetInt64(idind);

if (Managerid == Convert.ToInt64(ManagerId.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE UpdMan @id,@status,@login,@password";

cmd.CommandText = sql;

SqlParameter idParam = new SqlParameter("@id", SqlDbType.BigInt);

idParam.Value = Convert.ToInt64(ManagerId.Text);

cmd.Parameters.Add(idParam);

SqlParameter statusParam = new SqlParameter("@status", SqlDbType.Int);

statusParam.Value = Convert.ToInt32(ManagerStatus.Text);

cmd.Parameters.Add(statusParam);

SqlParameter loginParam = new SqlParameter("@login", SqlDbType.VarChar);

loginParam.Value = ManagerLogin.Text;

cmd.Parameters.Add(loginParam);

SqlParameter passwordParam = new SqlParameter("@password", SqlDbType.VarChar);

passwordParam.Value = ManagerPassword.Text;

cmd.Parameters.Add(passwordParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Данные успешно обновлены!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

else

MessageBox.Show("Недостаточно прав для вызова данной функции! Обратитесь к администраторам с status>=2");

}

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

{

if (CurrentUser.CurrentUserStatus > 1)

{

int Managerid;

int flag = 0;

SqlConnection conn = DBUtils.GetDBConnection();

bool a = ManagerId.Text == "";

if (a)

MessageBox.Show("Заполните все поля для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string sql1 = "SELECT id FROM ManagerTable";

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int idind = reader.GetOrdinal("id");

Managerid = (int)reader.GetInt64(idind);

if (Managerid == Convert.ToInt64(ManagerId.Text))

{

flag = 1;

break;

}

}

}

}

if (flag == 0)

throw new OverflowException();

string sql = "EXECUTE DelMan @id";

cmd.CommandText = sql;

SqlParameter idParam = new SqlParameter("@id", SqlDbType.BigInt);

idParam.Value = Convert.ToInt64(ManagerId.Text);

cmd.Parameters.Add(idParam);

cmd.ExecuteNonQuery();

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Менеджер удален!");

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

else

MessageBox.Show("Недостаточно прав для вызова данной функции! Обратитесь к администраторам с status>=2");

}

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

{

this.clientsListTableAdapter2.Fill(this.lastVersDataSet.ClientsList);

}

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

{

this.mastersListTableAdapter1.Fill(this.lastVersDataSet.MastersList);

}

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

{

this.filListTableAdapter1.Fill(this.lastVersDataSet.FilList);

}

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

{

this.servtypeListTableAdapter1.Fill(this.lastVersDataSet.ServtypeList);

}

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

{

this.fullservlistTableAdapter.Fill(this.lastVersDataSet.Fullservlist);

}

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

{

this.managerTableListTableAdapter2.Fill(this.lastVersDataSet.ManagerTableList);

}

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

{

SqlConnection conn = DBUtils.GetDBConnection();

bool a = FuncFilid.Text == "";

if (a)

MessageBox.Show("Заполните поле для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string outputstr = "";

string sql1 = "SELECT \* FROM FindFilMas(@ID)";

SqlParameter IDParam = new SqlParameter("@id", SqlDbType.BigInt);

IDParam.Value = Convert.ToInt64(FuncFilid.Text);

cmd.Parameters.Add(IDParam);

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int MasIdind = reader.GetOrdinal("Id мастера");

int MasId = (int)reader.GetInt64(MasIdind);

int FIOind = reader.GetOrdinal("ФИО мастера");

string FIO = reader.GetString(FIOind);

int ExpInd = reader.GetOrdinal("Стаж работы");

int Exp = reader.GetInt32(ExpInd);

int PhonenumInd = reader.GetOrdinal("Номер телефона");

string Phonenum = reader.GetString(PhonenumInd);

int LabcdateInd = reader.GetOrdinal("Дата закл. труд. договора");

string Labcdate = Convert.ToString(reader.GetDateTime(LabcdateInd));

int WageInd = reader.GetOrdinal("Размер зп");

int Wage = Convert.ToInt32(reader.GetValue(WageInd));

int PrepDateInd = reader.GetOrdinal("Дата предопл.");

string PrepDate = Convert.ToString(reader.GetDateTime(PrepDateInd));

int WageDateInd = reader.GetOrdinal("Дата зп");

string WageDate = Convert.ToString(reader.GetDateTime(WageDateInd));

outputstr += Convert.ToString(MasId) + " " + FIO + " " + Convert.ToString(Exp) + " " + Phonenum + " " + Labcdate + " " + Convert.ToString(Wage) + " " + PrepDate + " " + WageDate + "\n";

}

}

}

conn.Close();

conn.Dispose();

conn = null;

if (outputstr == "")

outputstr = "нет";

MessageBox.Show("Найденные записи:\n" + outputstr);

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

SqlConnection conn = DBUtils.GetDBConnection();

bool a = FuncClientId.Text == "";

if (a)

MessageBox.Show("Заполните поле для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string outputstr = "";

string sql1 = "SELECT dbo.SumServPrice(@id) AS 'SUMPR'";

SqlParameter IDParam = new SqlParameter("@id", SqlDbType.BigInt);

IDParam.Value = Convert.ToInt64(FuncClientId.Text);

cmd.Parameters.Add(IDParam);

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int SumprInd = reader.GetOrdinal("SUMPR");

int Sumpr = Convert.ToInt32(reader.GetValue(SumprInd));

outputstr += Convert.ToString(Sumpr);

}

}

}

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Суммарная цена:\n" + outputstr);

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

SqlConnection conn = DBUtils.GetDBConnection();

bool a = Phonenum.Text == "";

if (a)

MessageBox.Show("Заполните поле для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string outputstr = "";

string sql1 = "SELECT dbo.CheckN(@PhN) AS 'CheckN'";

SqlParameter PhNParam = new SqlParameter("@PhN", SqlDbType.NVarChar);

PhNParam.Value = Phonenum.Text;

cmd.Parameters.Add(PhNParam);

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int CheckNind = reader.GetOrdinal("CheckN");

bool CheckN = reader.GetBoolean(CheckNind);

if (!CheckN)

outputstr += "False";

else

outputstr += "True";

}

}

}

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Полученный результат:\n" + outputstr);

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

SqlConnection conn = DBUtils.GetDBConnection();

bool a = Email.Text == "";

if (a)

MessageBox.Show("Заполните поле для ввода данных!");

else

{

try

{

conn.Open();

SqlCommand cmd = conn.CreateCommand();

string outputstr = "";

string sql1 = "SELECT dbo.CheckE(@email) AS 'CheckE'";

SqlParameter emailParam = new SqlParameter("@email", SqlDbType.NVarChar);

emailParam.Value = Email.Text;

cmd.Parameters.Add(emailParam);

cmd.CommandText = sql1;

using (DbDataReader reader = cmd.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

int CheckEind = reader.GetOrdinal("CheckE");

bool CheckE = reader.GetBoolean(CheckEind);

if (!CheckE)

outputstr += "False";

else

outputstr += "True";

}

}

}

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("Полученный результат:\n" + outputstr);

}

catch

{

conn.Close();

conn.Dispose();

conn = null;

MessageBox.Show("При выполнении команды произошла ошибка, проверьте правильность введенных данных!");

}

}

}

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

{

Application.Exit();

}

Point lastPoint;

private void ManagerWindow\_MouseMove(object sender, MouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

this.Left += e.X - lastPoint.X;

this.Top += e.Y - lastPoint.Y;

}

}

private void ManagerWindow\_MouseDown(object sender, MouseEventArgs e)

{

lastPoint = new Point(e.X, e.Y);

}

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

{

if (CurrentUser.CurrentUserStatus < 2)

managerTableListDataGridView.Visible = false;

else

StatusOneUserWarning.Visible = false;

}

}

}