Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

**«Владимирский государственный университет имени Александра Григорьевича и Николая Григорьевича Столетовых»**

(ВлГУ)

Колледж инновационных технологий и предпринимательства

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ**

по дисциплине «Технология разработки и защиты баз данных»

Тема: «Отдел кадров организации: разработка и администрирование базы данных, разработка клиентского приложения»

Выполнила:

ст. гр. ПКсп-116

Толстова Ю.А.

Приняли:

Павлова О.Н.

Куприянов А.А.

Владимир 2019

**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 3](#_Toc24228948)

[1 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 4](#_Toc24228949)

[1.1 ER-ДИАГРАММА 4](#_Toc24228950)

[2 ПОДГОТОВКА БАЗЫ ДАННЫХ 5](#_Toc24228951)

[2.1 СЛОВАРЬ ДАННЫХ 5](#_Toc24228952)

[2.2 ПРОЦЕДУРЫ И ФУНКЦИИ 6](#_Toc24228953)

[3 РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ 8](#_Toc24228954)

[4 ИМПОРТ И ЭКСПОРТ ДАННЫХ 9](#_Toc24228955)

[4.1 ИМПОРТ ДАННЫХ ИЗ MS EXCEL 9](#_Toc24228956)

[4.2 ИМПОРТ ДАННЫХ С ПОМОЩЬЮ BULK INSERT 12](#_Toc24228957)

[4.3 КОМАНДНЫЙ ФАЙЛ 13](#_Toc24228958)

[5 УПРАВЛЕНИЕ ДОСТУПОМ 15](#_Toc24228959)

[6 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 16](#_Toc24228960)

[6.1 СТРУКТУРА ПРИЛОЖЕНИЯ 16](#_Toc24228961)

[6.2 ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС 16](#_Toc24228962)

[ЗАКЛЮЧЕНИЕ 19](#_Toc24228963)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 20](#_Toc24228964)

[ПРИЛОЖЕНИЕ А 21](#_Toc24228965)

[ПРИЛОЖЕНИЕ Б 22](#_Toc24228966)

[ПРИЛОЖЕНИЕ В 23](#_Toc24228967)

[ПРИЛОЖЕНИЕ Г 25](#_Toc24228968)

# **Введение**

В данной работе стоит задача создать программный модуль «Отдел кадров организации», который может быть использован отделами кадров. Так же, для хранения данных, которыми будет оперировать программа, необходимо разработать базу данных.

Организация может выполнять следующие действия:

* Просматривать информацию о сотрудниках;
* Добавлять и редактировать информацию о сотрудниках;
* Добавлять и редактировать всю информацию о сотрудниках.

В качестве основной информации о сотрудниках является: ФИО человека, возраст человека, отдел, должность.

# **1** **Проектирование базы данных**

## **ER-диаграмма**

База данных должна иметь следующий набор сущностей с, присущими им, атрибутами:

* Люди: ФИО, возраст, отдел, должность;
* Должность: Название должности;
* Отдел: Название отдела, область, город, улица. дом;

ER-диаграмма представлена на рисунке 1.1.

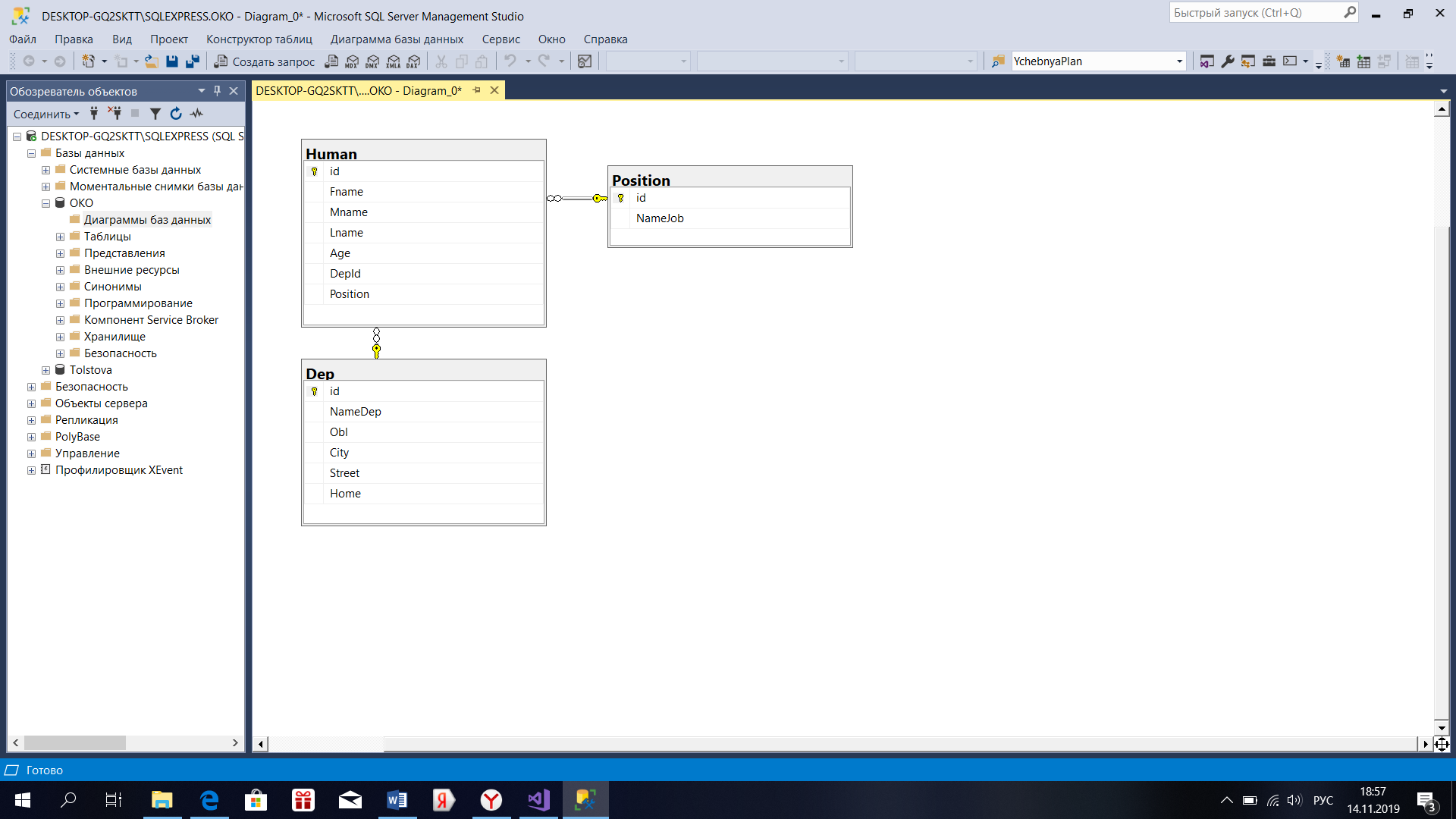


Рисунок 1.1 – ER-диаграмма

# **подготовка базы данных**

## **Словарь данных**

Словарь данных для создания базы данных представлен в таблице 1.

Таблица 1 – Словарь данных

|  |  |  |  |
| --- | --- | --- | --- |
| Human | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | Id | Да | Идентификационный номера человека |
|  | Fname | Да | Фамилия человека |
|  | Mname | Да | Имя человека |
|  | Lname | Да | Отчество человека |
|  | Age | Да | Возраст человека |
| Внешний | DepId | Да | Внешний ключ к таблице Dep |
| Внешний | Position | Да | Внешний ключ к таблице Position |
| Position | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | Id | Да | Идентификационный номер должности |
|  | NameJob | Да | Наименование должности |
| Dep |  | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | Id | Да | Идентификационный номер отдела |
|  | NameDep | Да | Название отдела |
|  | Obl | Да | Наименование области |
|  | City | Да | Наименование города |
|  | Street | Да | Наименование улицы |
|  | House | Да | Наименование дома |

Итоговая диаграмма базы данных, сгенерированная после создания всех таблиц представлена на рисунке 2.1.

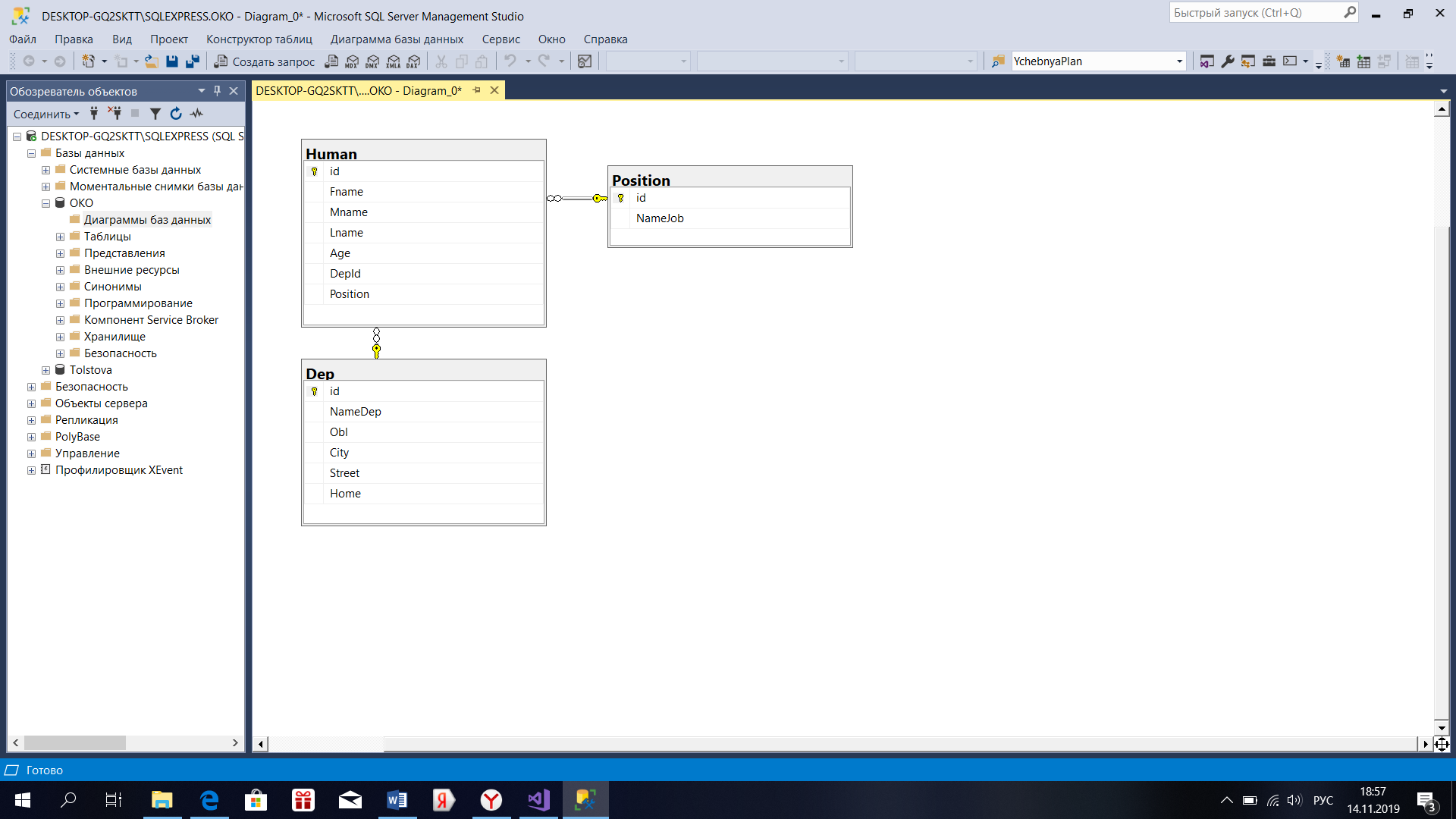


Рисунок 2.1 – Диаграмма базы данных

Код создания таблиц представлен в приложении А.

Код заполнения базы данных представлен в приложении Б.

## **Процедуры и функции**

В базе данных реализованы одна хранимая процедура и одна скалярная функция.

Хранимая процедура называется CoudOfRecors() и она предназначена для того, чтобы вывести количество записей.

Затем была разработана скалярная функция Stach(@date), которая ведет расчет года рождения.

На вход данная функция принимает параметр @date для определения даты.

Выходным параметром данной функции является числовое значение, которое будет определять год рождения работника.

Код создания хранимой процедуры и функции можно посмотреть в приложении В.

# **резервное копирование и восстановление**

Для резервного копирования с использованием командного файла изначально необходимо написать sql-скрипт, который будет выполнять резервное копирование базы данных и сохранит созданный скрипт в виде файла с расширением .sql.

Скрипт для резервного копирования представлен далее:

use master

go

backup database OKO

to disk='E:\CourseWork\Backups\course\_work.bak'

go

Затем, для запуска скрипта необходимо с помощью любого текстового редактора создать файл с расширением .bat или .cmd, внутри которого прописать следующий код:

sqlcmd -S DESKTOP-AS7DA8SD\SQLEXPRESS -i SQLQuery\_backup.sql

Теперь, при запуске файла будет создаваться резервная копия базы данных в указанной папке.

Аналогичным путем был создан sql-скрипт и командный файл для запуска восстановления базы данных из резервной копии.

Sql-скрипт для восстановления:

use master

go

restore database OKO

from disk = 'E:\CourseWork\Backups\course\_work.bak' with replace

go

Код командного файла:

sqlcmd -S DESKTOP-AS7DA86D\SQLEXPRESS -i SQLQuery\_restore.sql

# **ИМПОрт и экспорт данных**

## **Импорт данных из MS Excel**

Для импорта данных в таблицу базы данных необходимо создать источник, непосредственно из которого будет произведена операция.

Таким источником будет служить таблица, созданная в Microsoft Excel с расширением .xlsx. В данной таблице прописывается наименование столбца и данные, которые необходимо импортировать в таблицу базы данных. Данная таблица представлена на рисунке 4.1.

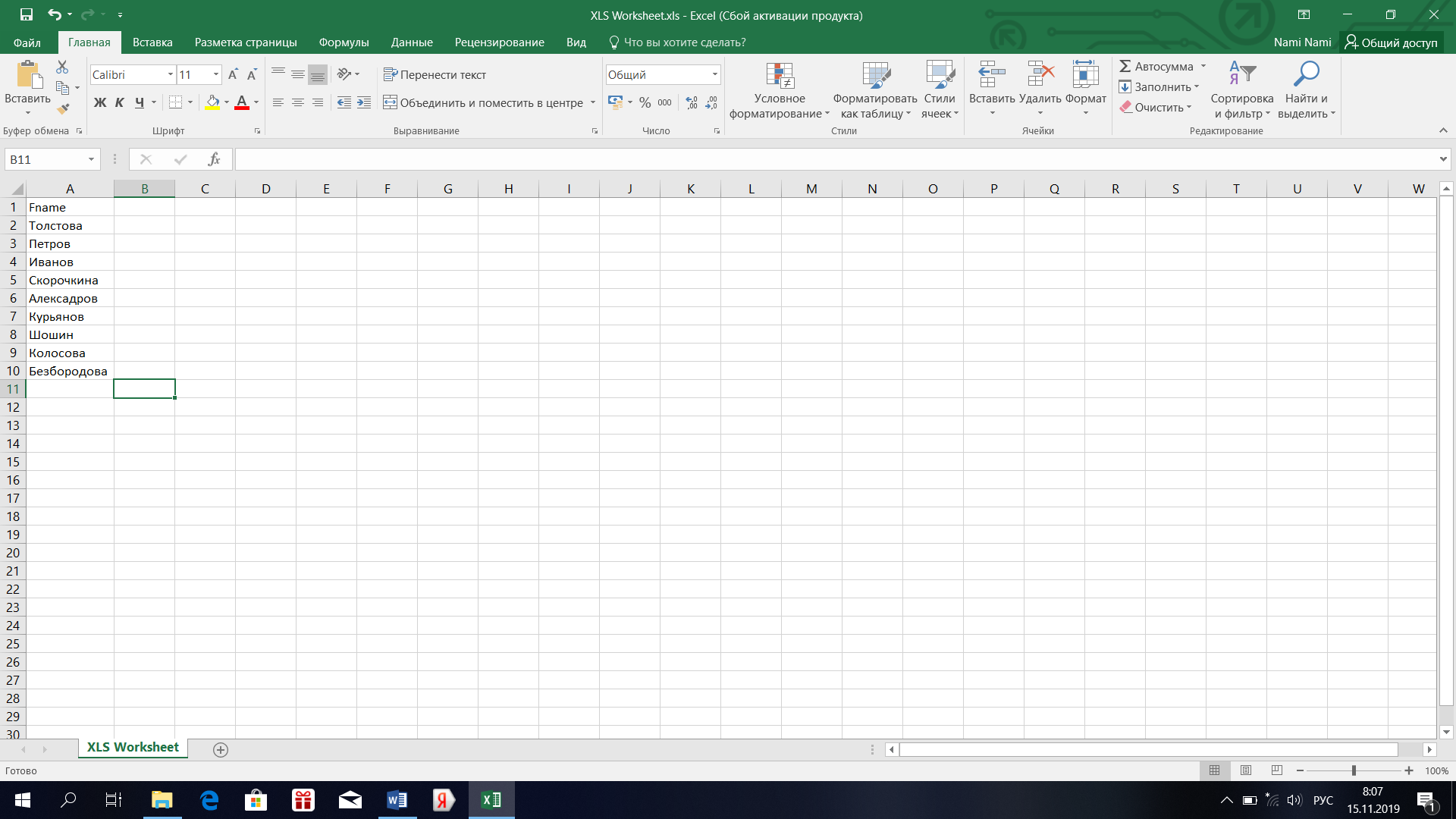


Рисунок 4.1 – Таблица для импорта

Для импорта данных из Excel-таблицы используется мастер импорта и экспорта SQL Server. На начальной странице настройки был выбран источник данных.

Выбор источника импорта данных показан на рисунке 4.2.

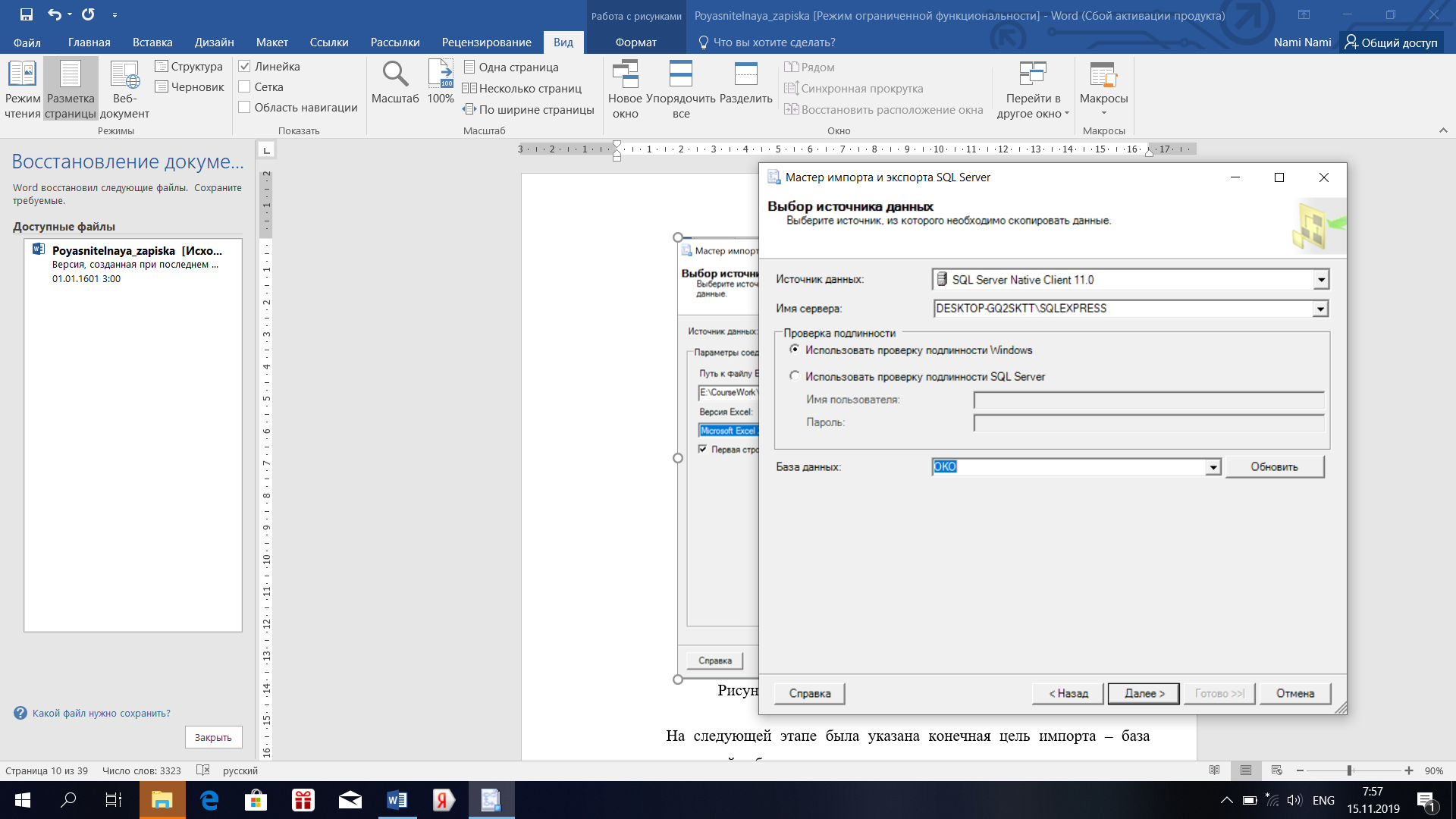


Рисунок 4.2 – Выбор источника импорта данных

На следующей этапе была указана конечная цель импорта – база данных курсовой работы.

Место назначения импорта данных представлено на рисунке 4.3.

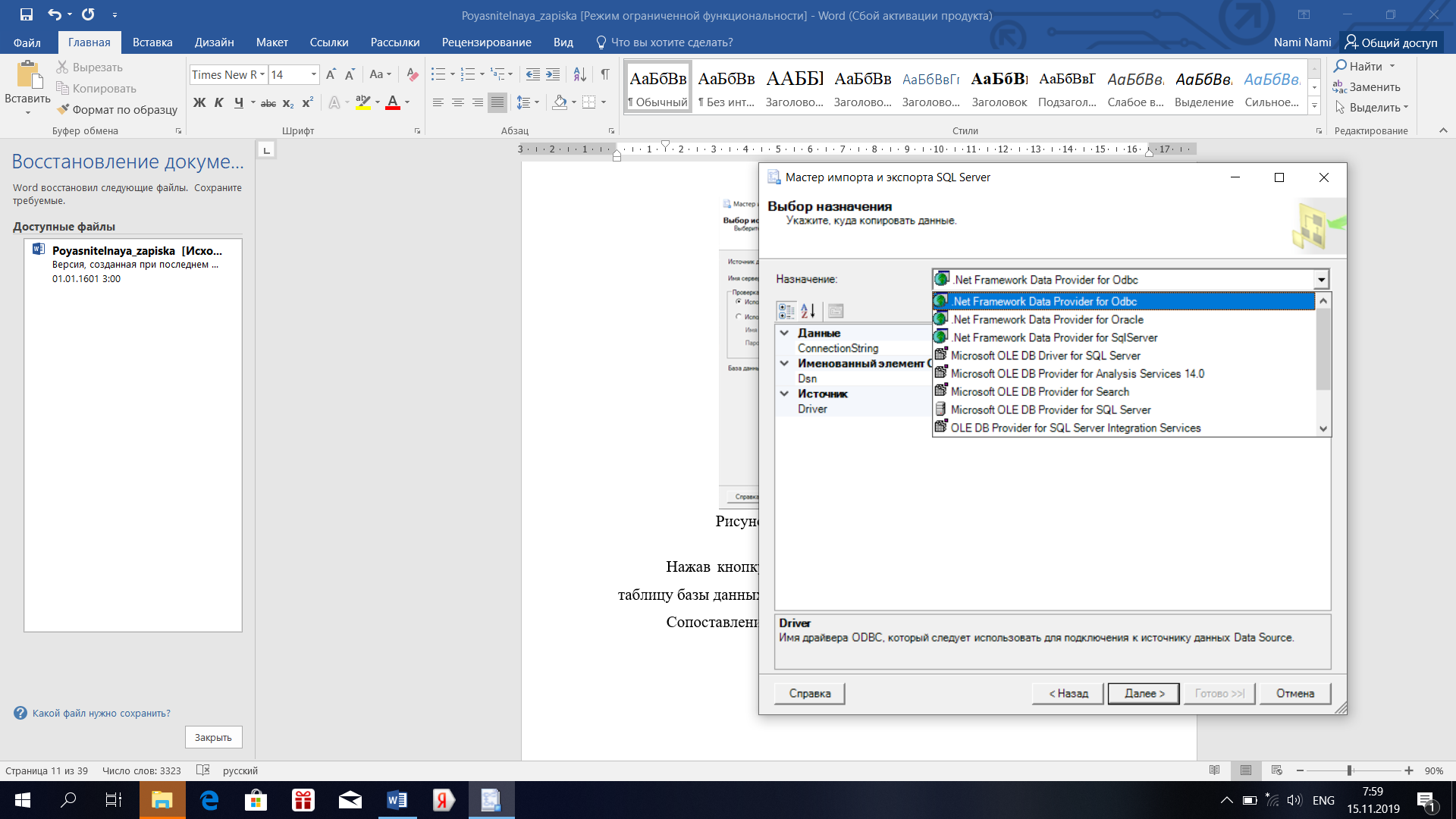


Рисунок 4.3 – Место назначения импорта данных

Нажав кнопку далее, нужно выбрать необходимый лист MS Excel и таблицу базы данных, для которых эти страницы предназначены.

Сопоставление страниц с таблицами показано на рисунке 4.4.

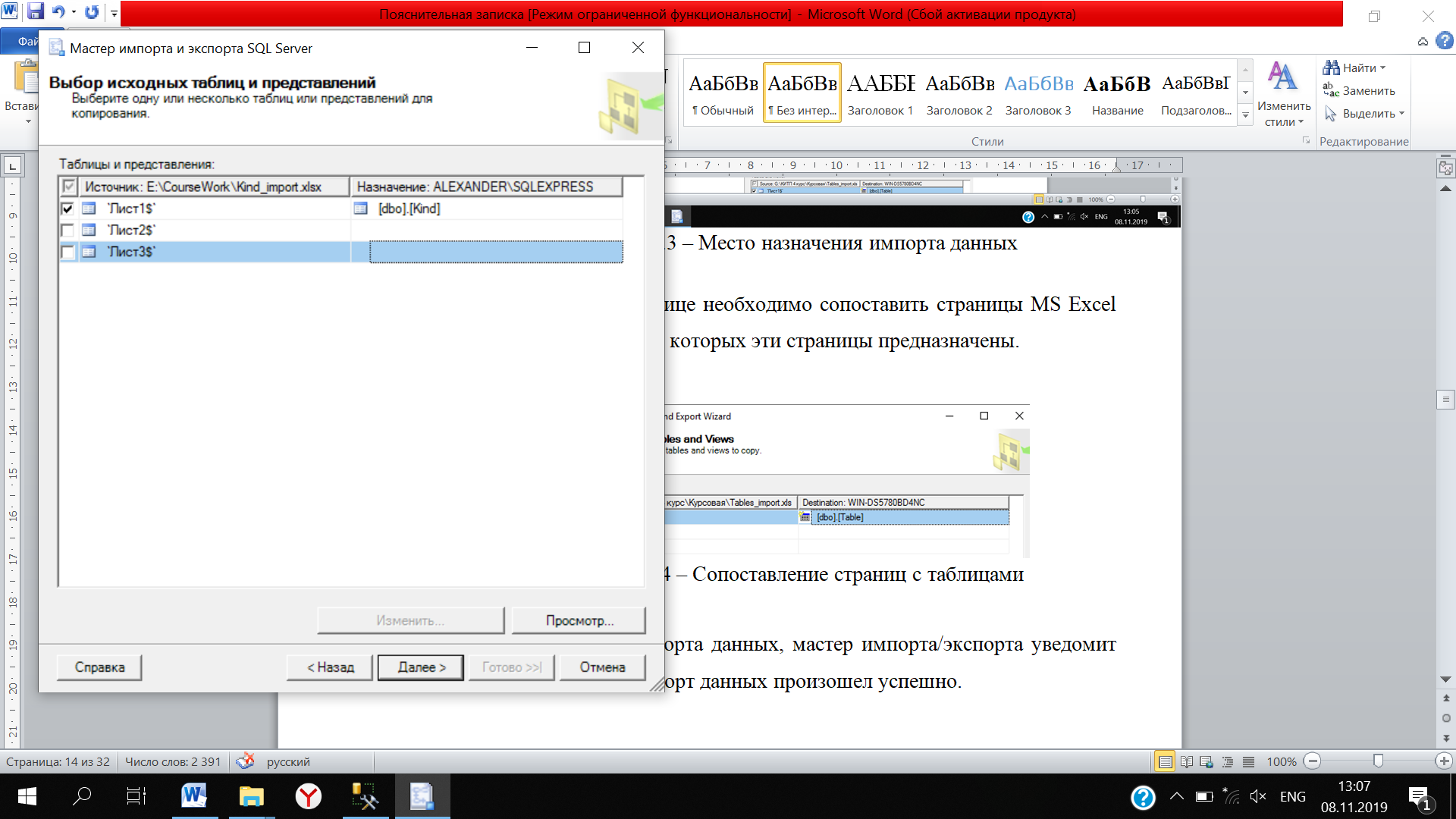


Рисунок 4.4 – Сопоставление страниц с таблицами

После успешного завершения импорта данных, мастер импорта/экспорта выдаст окно о завершении.

Завершение импорта данных представлено на рисунке 4.5.

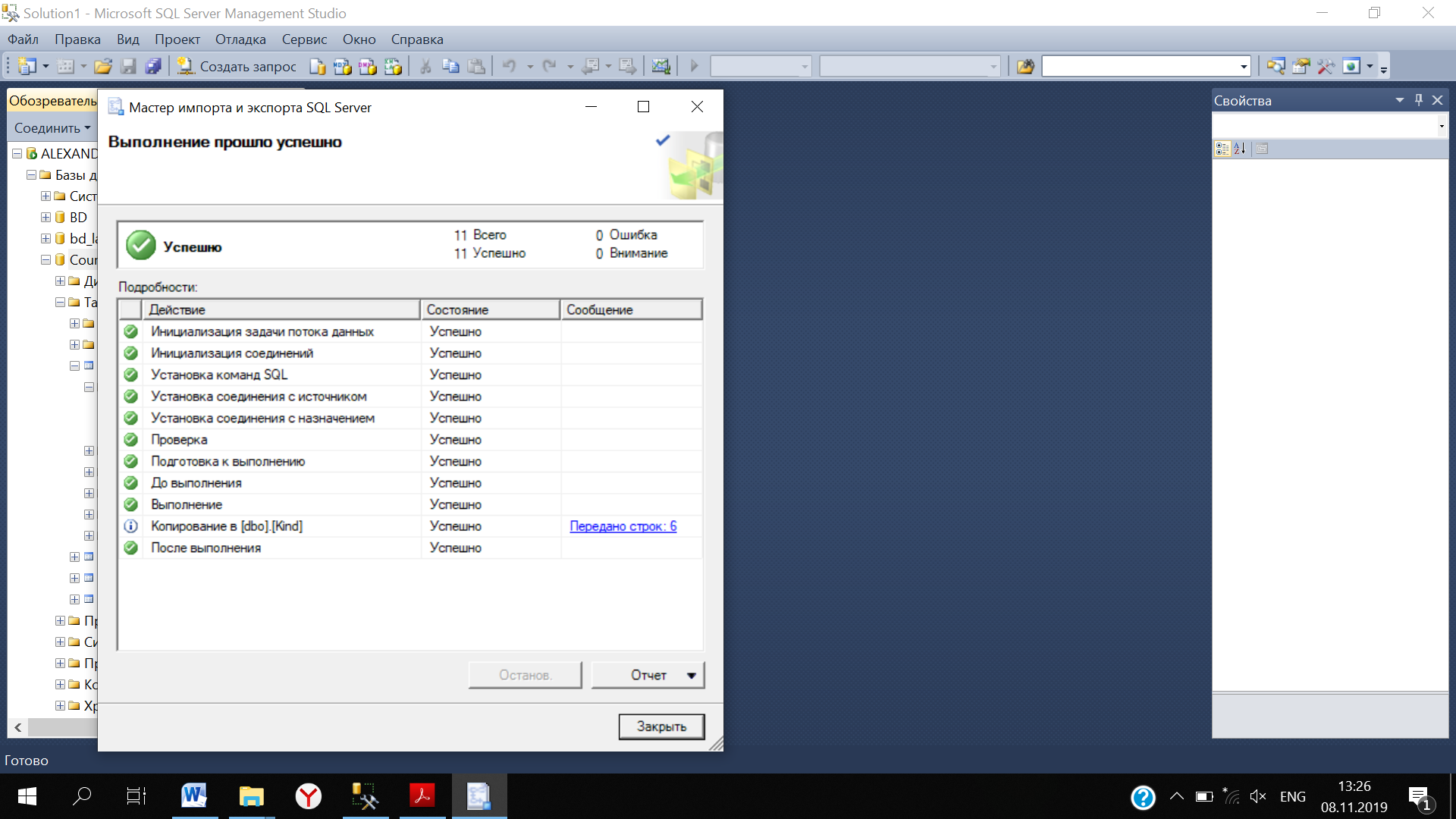


Рисунок 4.5 – Завершение импорта данных

## **Импорт данных с помощью BULK INSERT**

Sql-скрипт, который импортирует данные с помощью конструкции BULK INSERT показан ниже:

bulk insert OKO

from 'E:\CourseWork\Kind\_import.csv'

with(keepidentity,fieldterminator=';', rowterminator='\n');

## **Командный файл**

По заданию нужно создать командный файл, который будет выполнять экспорт данных в указанный файл. Для этого был написан sql-скрипт и создан командный файл.

Экспорта для таблицы «Kind» выглядит следующим образом:

sqlcmd –S DESKTOP-AS75DWB\SQLEXPRESS -d Course\_work -Q "select \* from Kind" –o

"Kind\_Export.csv" -s";" -w 700

При запуске данного файла, в папке, где он сохранился можно найти файл OKO\_Export.csv.

Экспортированные данные показаны на рисунке 4.6.

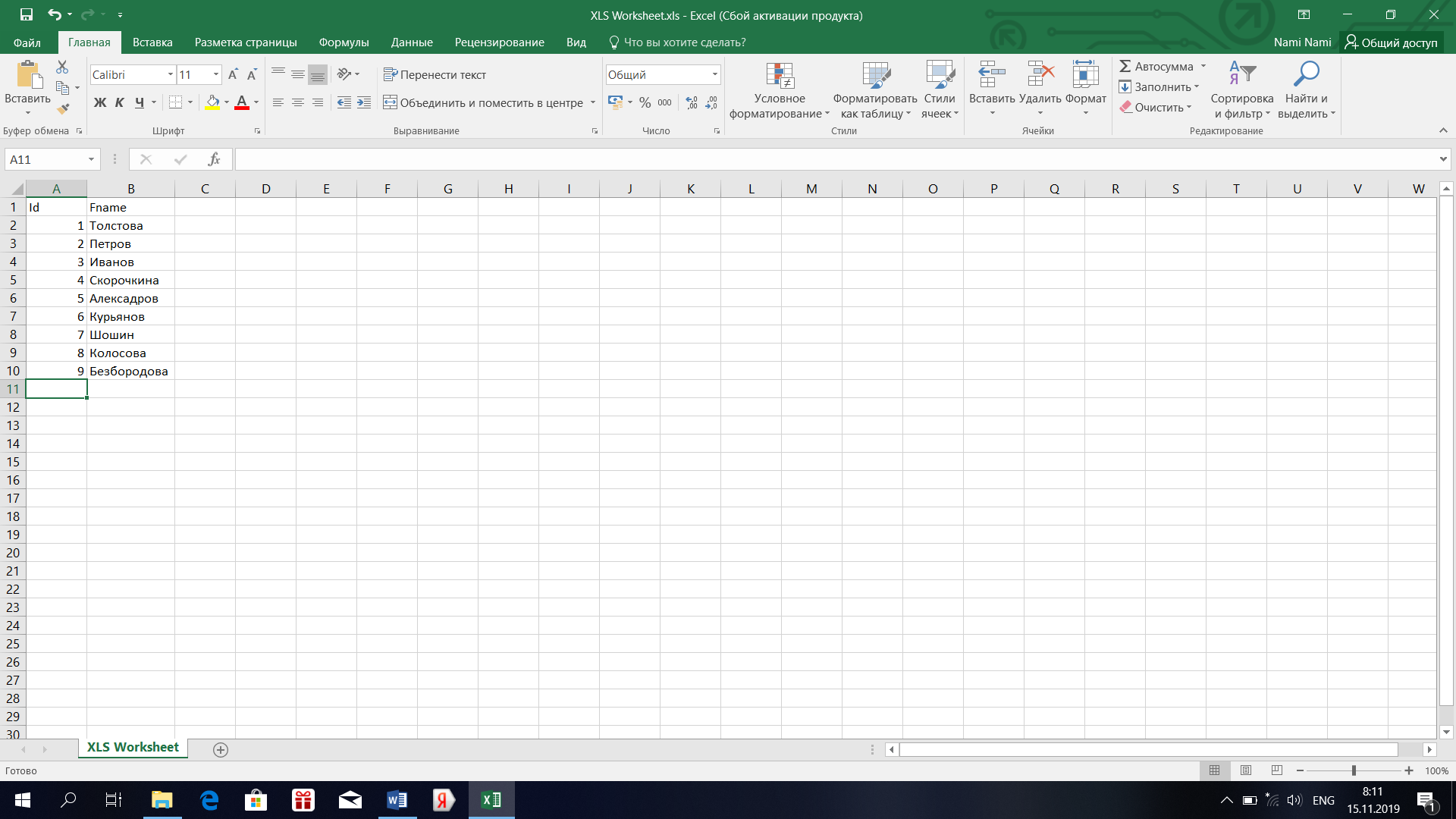


Рисунок 4.6 – Экспортированные данные

Аналогично был создан командный файл, который при вызове из командной строки будет выводить на экран необходимые данные. Текст данного исполняемого файла представлен далее:

sqlcmd -S DESKTOP-AS75DWB\SQLEXPRESS

-d Course\_work

-Q "select

h.Fname,

h.Mname,

h.Lname,

h.Age,

h.DepId,

h.Position,

from

Human h

join Position p on p.Id=h.Position"

При вызове его в командной строке на экране будет показан результат, представленный на рисунке 4.7.

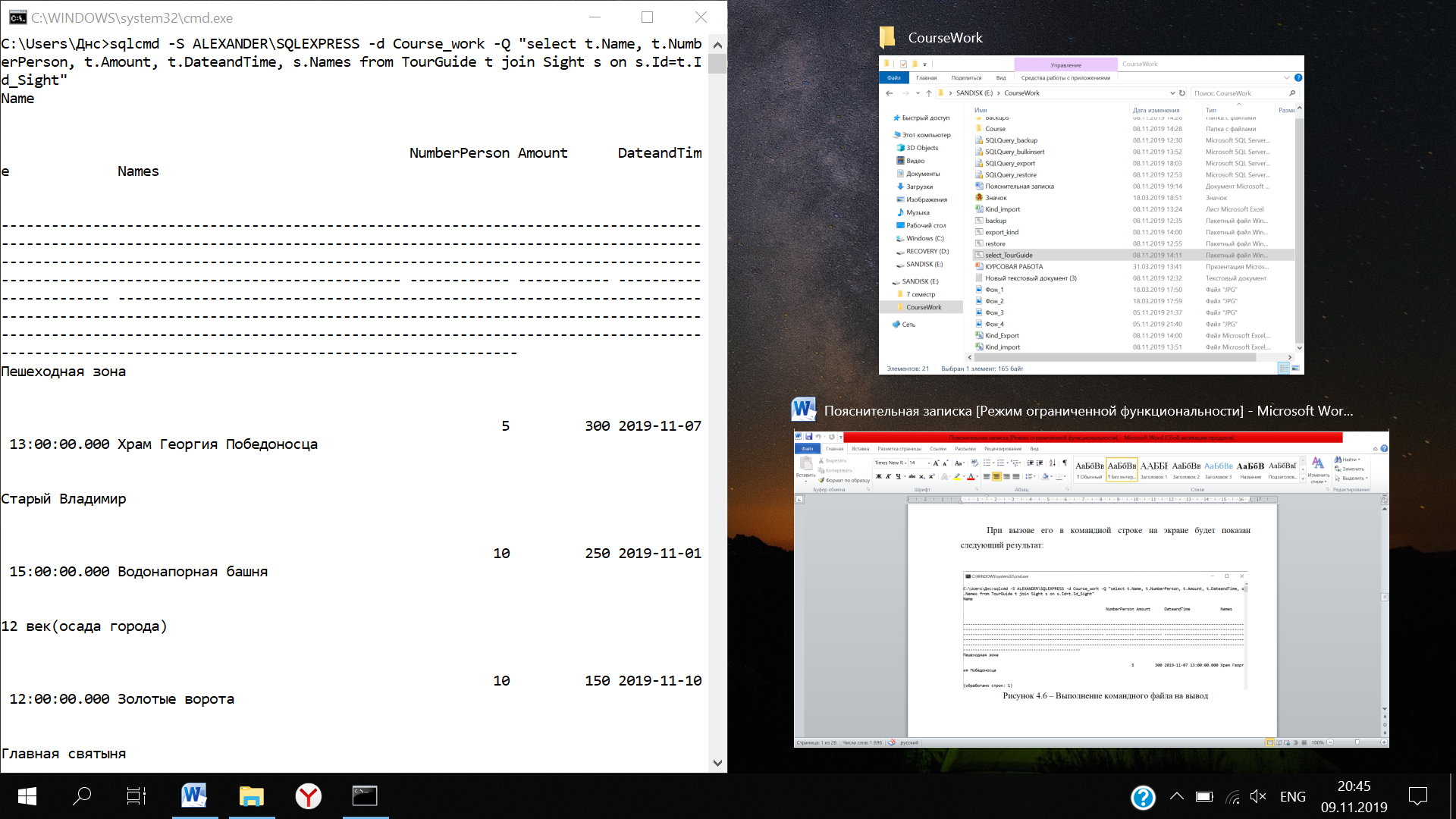


Рисунок 4.7 – Выполнение командного файла на вывод

# **Управление доступом**

Для управления доступом в базе данных с помощью sql-скрипта были созданы две роли:

* Director, который неограничен в работе с данными в базе данных;
* Jober, который может лишь просматривать данные в таблицах базы данных. Соответственно, для него ограничена возможность добавления, удаления и редактирование данных.

Sql-скрипт, выполняющий создание ролей и пользователей представлен в приложении В.

# **разработка клиентского приложения**

## **Структура приложения**

Мое приложение состоит из:

* главной формы , которая отвечает за отображение данных из базы данных
* форма редактирования и добавления, которая отвечает за добавление и редактирования данных.

## **Пользовательский интерфейс**

При запуске приложения перед пользователем появится окно главной формы, которое содержит:

* фоновое изображение;
* таблицу со всеми сотрудниками;
* кнопка для подсчета года рождения;
* кнопка для подсчета количество записей в таблице.

Главная форма показана на рисунке 6.1.

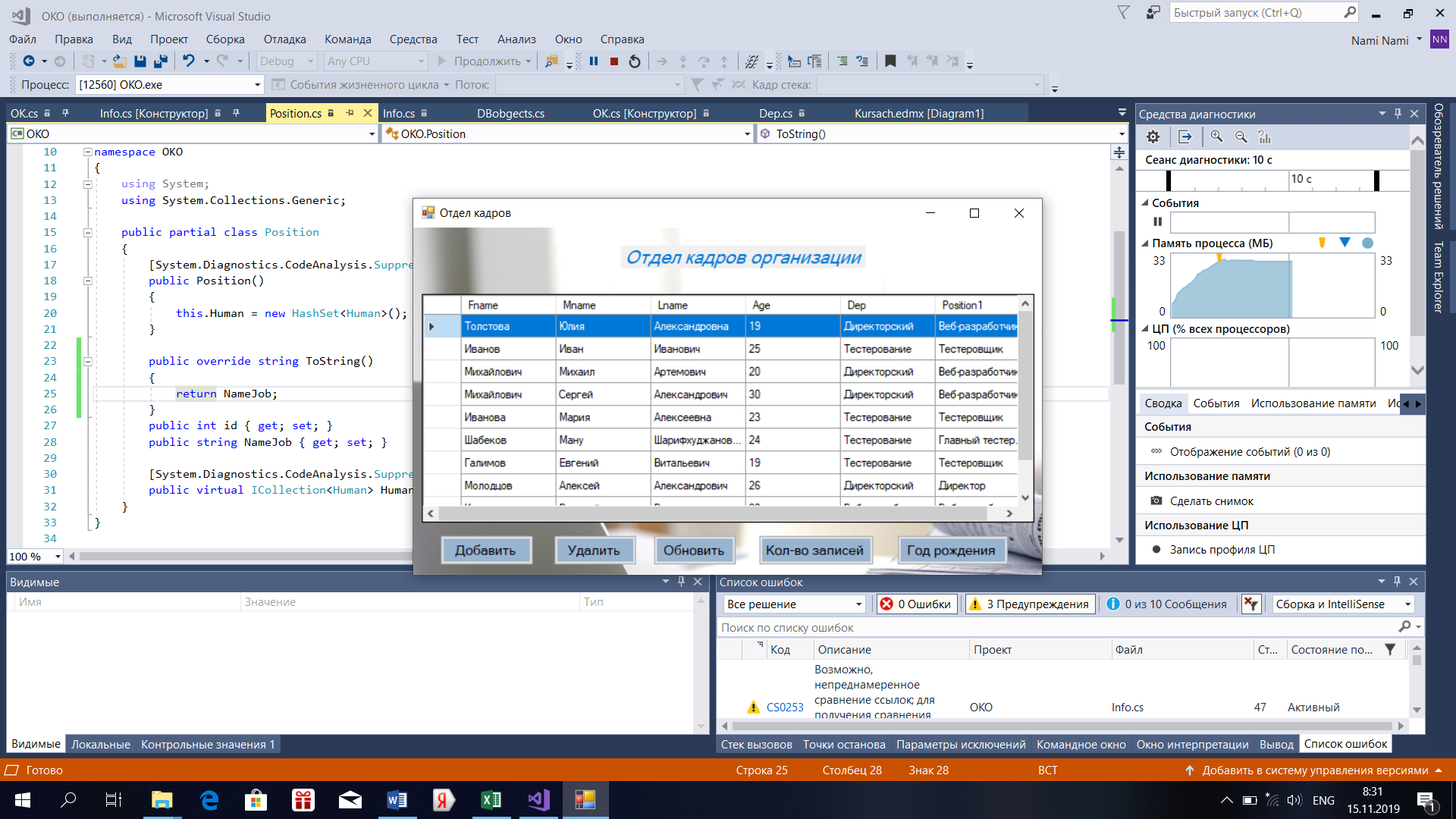


Рисунок 6.1 – Главная форма

Для того чтобы добавить человека необходимо нажать на кнопку «Добавить». Перед нами откроется форма, которая содержит:

* фоновое изображение;
* TextBox для заполнения/редактирования ФИО;
* TextBox для заполнения/редактирования возраста;
* ComboBox для заполнения/редактирования отдела, где работает сотрудник;
* ComboBox для заполнения/редактирования должности сотрудника;
* Button для сохранения работника.

Данная форма представлена на рисунке 6.2.

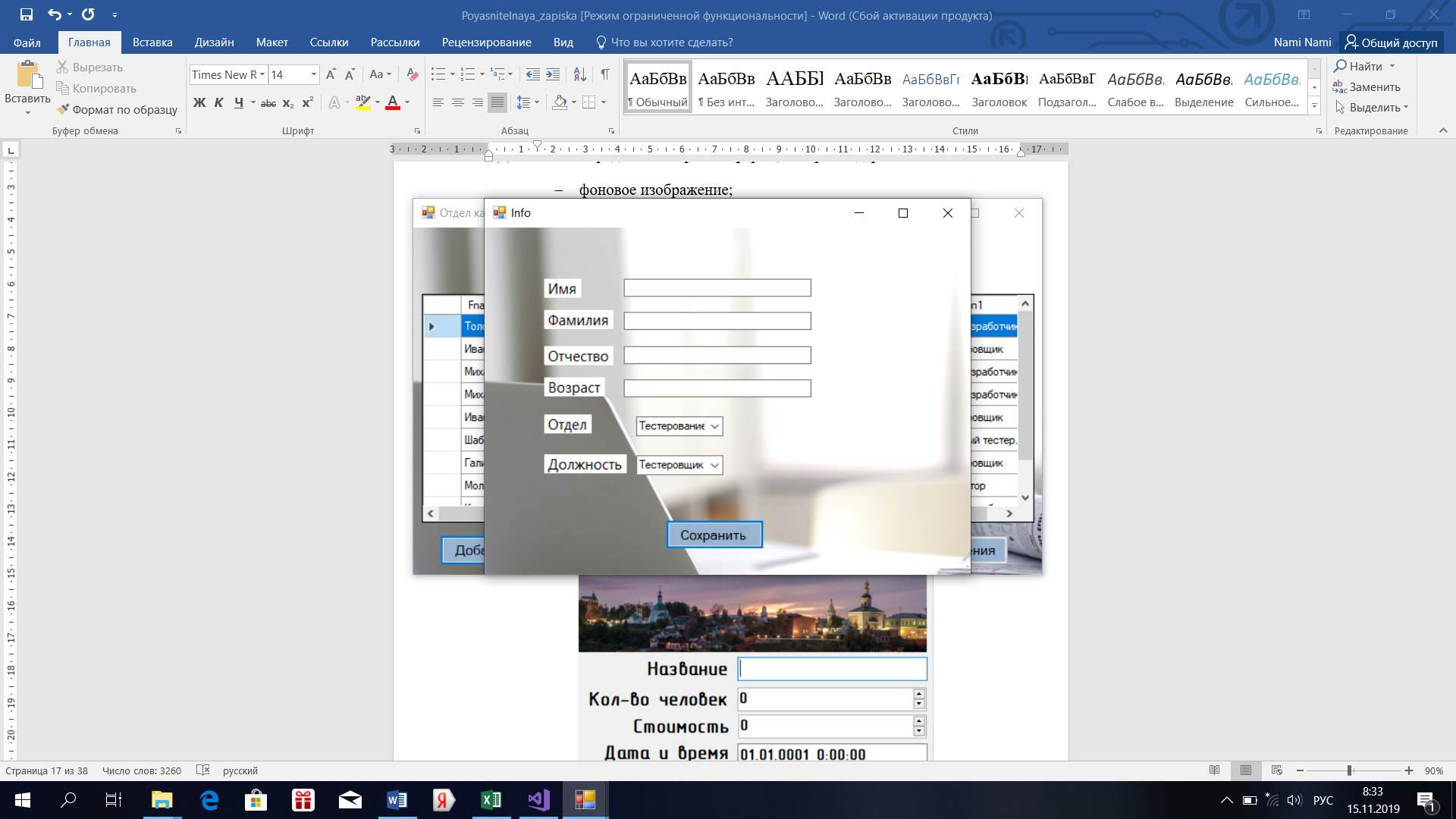


Рисунок 6.2 – Добавление человека

Для того чтобы отредактировать информацию о работнике необходимо два раза кликнуть по нужной строке и после этого откроется форма, представленная на рисунке 6.3.

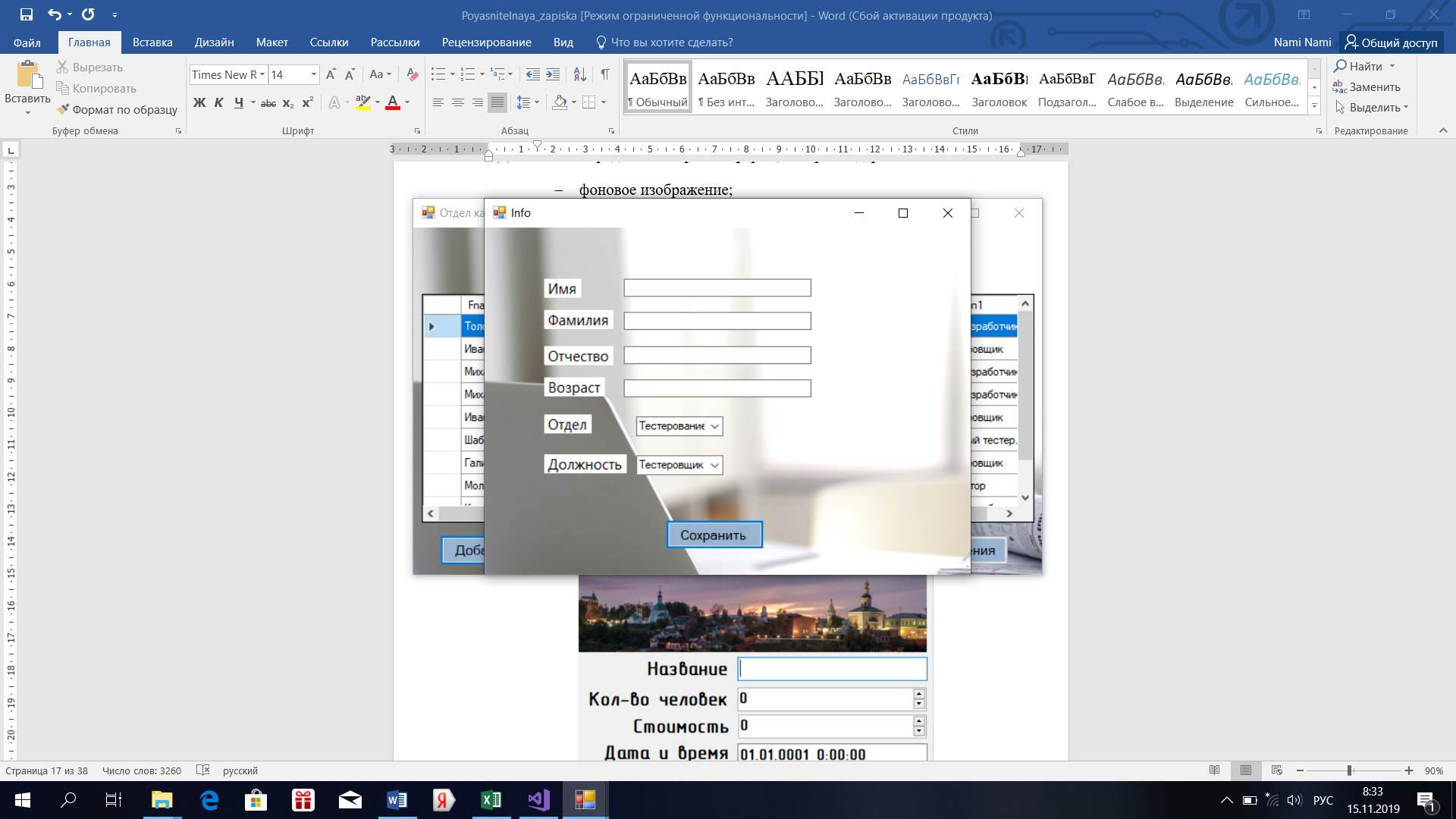


Рисунок 6.3 – Редактирование экскурсии

# **заключение**

Результатом выполнения данной работы стало приложение, позволяющее организовать работу системы отдела кадров. Разработанная программа удовлетворяет всем требованиям, предъявленным в задании, позволяет добавлять, удалять и редактировать записи списка. Для работы с базой данных было разработано приложение, которое грамотно реализует все функции, которые необходимы для системы, описанной в задании.

Работа была выполнена в среде программирования Visual Studio2015 и Microsoft SQL Server Management Studio.

Во время выполнения курсовой работы изучены и закреплены навыки создания sql-скриптов для проектирования базы данных.

# **СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Астахова И.Ф. SQL в примерах и задачах; Учеб. пособие / И.Ф. Астахова, А.П. Толстобров, В.М. Мельников.— Мн.: Новое знание, 2002 — 176 с. — ISBN 985-475-004-3.
2. Описание основных приемов нормализации базы данных // Служба поддержки Майкрософт URL: https://support.microsoft.com/ru-ru/help/283878/description-of-the-database-normalization-basics (дата обращения: 10.11.2019).
3. Петкович Д. Microsoft® SQL Server™ 2012. Руководство для начинающих: Пер. с англ. — СПб.: БХВ-Петербург, 2013. — 816 с.: ил. ISBN 978-5-9775-0854-4
4. Практическое руководство. Выполнение хранимой процедуры в LightSwitch // MSDN – сеть разработчиков Microsoft URL: https://msdn.microsoft.com/ru-ru/library/jj635144.aspx (дата обращения: 10.11.2019).

# приложение а

Создание БД

CREATE TABLE [dbo].[Human] (

[id] INT IDENTITY (1, 1) NOT NULL,

[Fname] VARCHAR (150) NOT NULL,

[Mname] VARCHAR (150) NOT NULL,

[Lname] VARCHAR (150) NOT NULL,

[Age] SMALLINT NOT NULL,

[DepId] INT NULL,

[Position] INT NULL,

PRIMARY KEY CLUSTERED ([id] ASC),

FOREIGN KEY ([DepId]) REFERENCES [dbo].[Dep] ([id]),

FOREIGN KEY ([Position]) REFERENCES [dbo].[Position] ([id])

);

CREATE TABLE [dbo].[Dep] (

[id] INT IDENTITY (1, 1) NOT NULL,

[NameDep] VARCHAR (500) NOT NULL,

[Obl] VARCHAR (300) NOT NULL,

[City] VARCHAR (300) NOT NULL,

[Street] VARCHAR (300) NOT NULL,

[Home] VARCHAR (20) NOT NULL,

PRIMARY KEY CLUSTERED ([id] ASC)

);

CREATE TABLE [dbo].[Position] (

[id] INT IDENTITY (1, 1) NOT NULL,

[NameJob] VARCHAR (500) NOT NULL,

PRIMARY KEY CLUSTERED ([id] ASC)

);

# приложение Б

Заполнение таблиц

insert into Dep(NameDep,Obl,City,Street,Home)

value ('Директорский','Владимирская','Владимир','Горького','71'),

('Директорский','Владимирская','Владимир','Горького','71'),

('Юридический','Владимирская','Владимир','Садовая','63'),

('Директорский','Владимирская','Владимир','Гастелло','71'),

('Директорский','Владимирская','Владимир','Горького','96'),

('Директорский','Владимирская','Владимир','Садовая','70'),

('Директорский','Владимирская','Владимир','Горького','71'),

('Директорский','Владимирская','Владимир','Горького','71'),

('Директорский','Владимирская','Владимир','Гастелло','50'),

('Директорский','Владимирская','Владимир','Горького','71'),

insert into Human(Fname,Mname,Lname,Age,DepId,Position)

values ('Толстова','Юлия','Александровна', 19, 1,1),

('Молодцов','Даниил','Михайлович', 19, 1, 1),

('Безбородова','Александра','Александровна', 20, 1, 1),

('Колосова','Елена','Александровна', 19, 1, 1),

('Потапов','Алексей','Александрович', 23, 1, 1),

('Тhyryht','hgh','guiliu', 19, 1, 1),

('bdf','htrgtet','ghdjg', 19, 1, 1),

('hjglh','hjfk','hvhvl', 19, 1, 1),

('jbjkb;','jkbh;','hglhg', 19, 1, 1),

('h;jh','b;u','vlvhjkb', 19, 1, 1)

insert into Position(NameJob)

values ('Веб-разработчик'),

('Тестеровщик'),

('Веб-дизайнер'),

('Дизайнер интерфейса'),

('Веб-разработчик'),

('нгагщнг'),

('рпд1')

# приложение В

Создание процедур и функций

USE [OKO]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[CoutOfRecords] Script Date: 15.11.2019 9:06:47 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER PROCEDURE [dbo].[CoutOfRecords]

AS

BEGIN

select count(\*) from Human

END

USE [OKO]

GO

/\*\*\*\*\*\* Object: UserDefinedFunction [dbo].[Stach] Script Date: 15.11.2019 9:07:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

ALTER FUNCTION [dbo].[Stach]

(

@id int

)

RETURNS int

AS

BEGIN

declare @Resoult int

select @Resoult = YEAR(getdate()) - h.Age from Human h where h.id = @id

return @Resoult

END

Создание триггеров

GO

create trigger people on Human

after delete

as

print 'Успешно удалено'

Создание ролей и пользователей

use Course\_work

go

create login Director\_1

with password = '1111';

create login user\_2

with password = '2222';

create user director\_1

for login director\_1

create user user\_2

for login user\_2

create role Director authorization director\_1

create role USER authorization user\_2

deny delete to user\_2;

# приложение г

Код главной формы

namespace OKO

{

public partial class OK : Form

{

public OK()

{

InitializeComponent();

CenterToScreen();

Fill();

}

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

{

}

public void Fill()

{

dataGridView1.DataSource = null;

this.dataGridView1.DataSource = DBobgects.Entities.Human.ToList();

dataGridView1.Columns["Id"].Visible = false;

dataGridView1.Columns["DepId"].Visible = false;

dataGridView1.Columns["Position"].Visible = false;

}

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

{

Delete();

}

void Delete()

{

Human human = (Human)dataGridView1.Rows[dataGridView1.CurrentRow.Index].DataBoundItem;

if (DBobgects.Entities.Human.Where(c => c.id == human.id).Count() > 0)

{

DBobgects.Entities.Human.Remove(human);

DBobgects.Entities.SaveChanges();

}

}

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

{

Human human = new Human();

Info i = new Info(human);

i.ShowDialog();

this.Fill();

}

private void button5\_Click(object sender, EventArgs e)

{

DBobgects.Entities.SaveChanges();

Fill();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView1\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

Human human = (Human)this.dataGridView1.Rows[e.RowIndex].DataBoundItem;

Info i = new Info(human);

i.ShowDialog();

this.Fill();

}

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

{

MessageBox.Show(DBobgects.Entities.CoutOfRecords().FirstOrDefault().ToString());

}

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

{

try

{

MessageBox.Show(DBobgects.Entities.Database.SqlQuery<int>($"use OKO select dbo.Stach({dataGridView1.CurrentRow.Cells[0].Value.ToString()})").FirstOrDefault().ToString());

}

catch (Exception exc)

{

MessageBox.Show("Не выбрана строка");

}

}

}

}

Код формы редактирования и добавления

namespace OKO

{

public partial class Info : Form

{

Human client;

public Info(Human client)

{

this.client = client;

InitializeComponent();

CenterToScreen();

if (client.id != 0)

fill();

comboBox2.DataSource = DBobgects.Entities.Dep.Select(c => c.NameDep).ToList();

comboBox1.DataSource = DBobgects.Entities.Position.Select(c => c.NameJob).ToList();

}

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

{

}

void fill()

{

textBox2.Text = client.Fname.ToString();

textBox1.Text = client.Mname.ToString();

textBox3.Text = client.Lname.ToString();

textBox4.Text = client.Age.ToString();

comboBox1.Text = DBobgects.Entities.Position.Where(c => c.id == client. Position).Select(c => c.NameJob).ToString() ;

comboBox2.Text = DBobgects.Entities.Dep.Where(c => c.id == client.DepId).Select(c => c.NameDep).ToString();

}

void fillentity()

{

client.Fname = textBox2.Text;

client.Mname = textBox1.Text;

client.Lname = textBox3.Text;

client.Age = Convert.ToInt16(textBox4.Text);

client.DepId = DBobgects.Entities.Dep.Where(c=>c.NameDep == comboBox2.SelectedItem).Select(c=>c.id).FirstOrDefault();

client.Position = DBobgects.Entities.Position.Where(c => c.NameJob == comboBox1.SelectedItem).Select(c => c.id).FirstOrDefault();

}

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

{

fillentity();

if (!DBobgects.Entities.Human.ToArray().Contains(client))

{

DBobgects.Entities.Human.Add(client);

}

DBobgects.Entities.SaveChanges();

}

}

}