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 «Теория разработки и защиты баз данных»

Тема: «Аудиторный фонд многокорпусного здания: разработка и администрирование базы данных, разработка клиентского приложения»

Выполнил:

ст. гр. ПКсп-116

Молодцов Д.М

Приняли:

Павлова О.Н.

Куприянов А.А.

Владимир 2019

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc23889074)

[1 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ 4](#_Toc23889075)

[1.1 ER-ДИАГРАММА 4](#_Toc23889076)

[2 ПОДГОТОВКА БАЗЫ ДАННЫХ 6](#_Toc23889077)

[2.1 СЛОВАРЬ ДАННЫХ 6](#_Toc23889078)

[2.2 ПРОЦЕДУРЫ И ФУНКЦИИ 9](#_Toc23889079)

[3 РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ 11](#_Toc23889080)

[4 ИМПОРТ И ЭКСПОРТ ДАННЫХ 13](#_Toc23889081)

[4.1 ИМПОРТ ДАННЫХ ИЗ MS EXCEL 13](#_Toc23889082)

[4.2 ИМПОРТ ДАННЫХ С ПОМОЩЬЮ BULK INSERT 15](#_Toc23889083)

[4.3 КОМАНДНЫЙ ФАЙЛ 16](#_Toc23889084)

[5 УПРАВЛЕНИЕ ДОСТУПОМ 19](#_Toc23889085)

[6 РАЗРАБОТКА КЛИЕНТСКОГО ПРИЛОЖЕНИЯ 20](#_Toc23889086)

[6.1 СТРУКТУРА ПРИЛОЖЕНИЯ 20](#_Toc23889087)

[6.2 ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС 21](#_Toc23889088)

[ЗАКЛЮЧЕНИЕ 26](#_Toc23889089)

[ПРИЛОЖЕНИЕ А 2](#_Toc23889090)5

[ПРИЛОЖЕНИЕ Б 2](#_Toc23889091)6

[ПРИЛОЖЕНИЕ В 2](#_Toc23889092)7

[ПРИЛОЖЕНИЕ Г 3](#_Toc23889093)2

[ПРИЛОЖЕНИЕ Д 3](#_Toc23889094)5

# Введение

В данной работе стоит задача создать программный модуль «Аудиторный фонд многокорпусного здания», который может быть использован в различных многокорпусных учреждениях. Так же, для хранения данных, которыми будет оперировать программа, необходимо разработать базу данных.

Пользователь приложения может выполнять следующие действия:

* Просматривать информацию об аудитории.
* Добавлять и изменять аудиторию.
* Удалять аудиторию.

Для того чтобы добавить аудиторию, пользователь должен будет ввести: корпус, вместимость корпуса, аудиторию, вместимость аудитории, ФИО преподавателя ответственного за данную аудиторию, его личный номер, а также его дату рождения.

# Проектирование базы данных

## ER-диаграмма

База данных учета аудиторий в многокорпусном здании должна иметь следующий набор сущностей с, присущими им, атрибутами:

* Информация о корпусе: Номер корпуса и его вместимость.
* Информация о преподавателе: Фамилия, Имя, Отчество, номер телефона и дата рождения.
* Информация об аудитории: Номер аудитории и ее вместимость, так же, корпус где находится аудитория и ответственного за аудиторию.

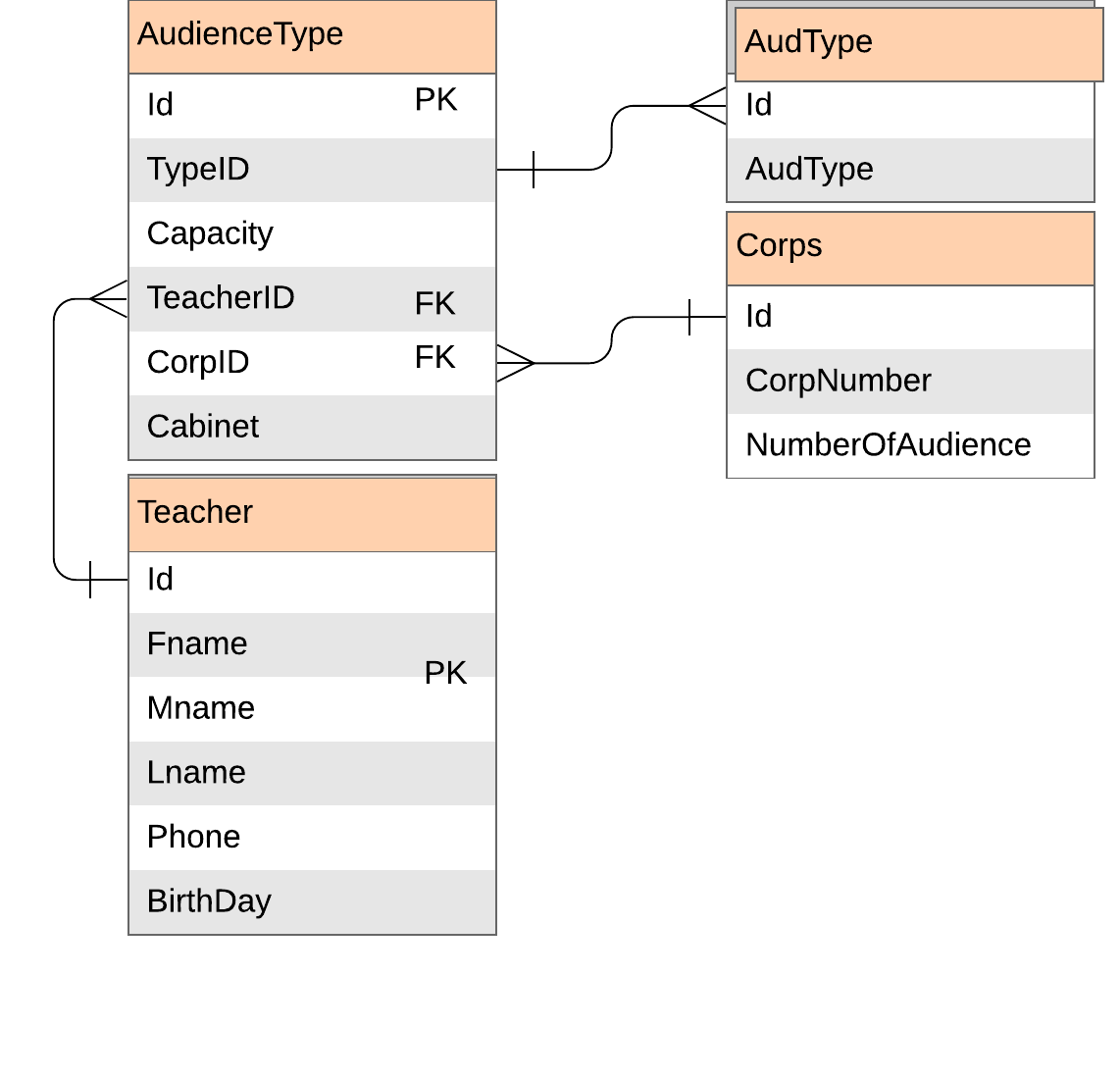


Рисунок 1.1 – ER-диаграмма

# 

# подготовка базы данных

## Словарь данных

|  |  |  |  |
| --- | --- | --- | --- |
| Corps | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | Id | Да | Идентификационный номер |
|  | CorpNumber | Да | Номер аудитории |
|  | NumberOfAudiences | Да | Количество аудиторий |

|  |  |  |  |
| --- | --- | --- | --- |
| Teacher | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | Id | Да | Идентификационный номер |
|  | Fname | Да | Имя преподавателя |
|  | Mname | Да | Фамилия преподавателя |
|  | Lname | Да | Отчество преподавателя |
|  | Phone | Да | Номер телефона преподавателя |
|  | Birthday | Да | Дата рождения преподавателя |

|  |  |  |  |
| --- | --- | --- | --- |
| AudienceType | | | |
| Ключ | Поле | Обязательное | Примечание |
| Первичный | ID | Да | Идентификационный номер |
|  | TypeOf | Да | Тип аудитории |
|  | Capacity | Да | Вместимость аудитории |
|  | Cabinet | Да | Номер кабинета |
| Внешний | TeacherID | Да | Внешний ключ преподавателя |
| Внешний | CorpID | Да | Внешний ключ корпуса |

Итоговая диаграмма классов, сгенерированная после создания всех таблиц, выглядит следующим образом:

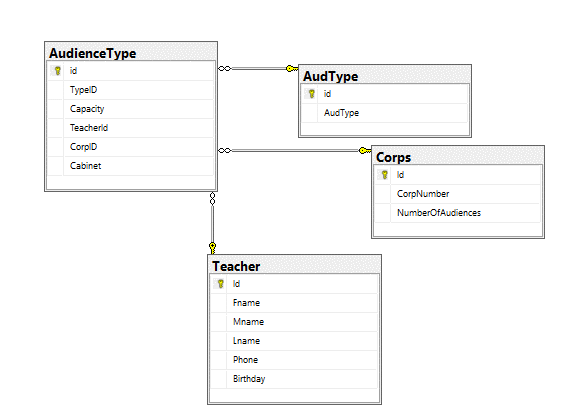


Рисунок 2.1 – Диаграмма базы данных

Код создания таблиц можно посмотреть в приложении А.

## Процедуры и функции

Всего в базе данных реализованы: две хранимых процедур и одна скалярная функция.

Процедура называется GetCountOfAllRecords и она предназначена для того, чтобы вывести сумму всех записей в таблице. Эта процедура помогает узнать, общее количество записей.

На выходе формируется выборка, в которую будут входить сумма количества всех полей.

Затем была разработана скалярная функция, которая ведет расчет количества забронированных столиков по определенной дате, она называется gethumancapacity.

На вход данная функция принимает: @corp для определения корпуса.

Выходным параметром данной функции является числовое значение, которое будет определять количество аудиторий в корпусе.

Код создания хранимых процедур и функций можно посмотреть в приложении В.

# 

# резервное копирование и восстановление

Для резервного копирования был реализован административный раздел, вход в раздел осуществляется через ввод пароля, уже в административной форме по средствам запроса выполняется резервное копирование, где saveFileDialog.FileName это путь и название файла которые выбрал пользователь.

Пример такого запроса представлен далее:

use master backup database MBAF to disk=\'{saveFileDialog.FileName}\'

Листинг формирования запроса из кода программы:

if (saveFileDialog.ShowDialog() == DialogResult.OK) { DBObject.context.Database.ExecuteSqlCommand(System.Data.Entity.TransactionalBehavior.DoNotEnsureTransaction, $"use master backup database MBAF to disk=\'{saveFileDialog.FileName}\'");

MessageBox.Show("Резервная копия сделана успешно!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);}

Затем, чтобы совершить данный запрос, необходимо с помощью кнопки в административном разделе выбрать куда будет сделано резервное копирование и запустить запрос.

Аналогичным путем создается запрос на восстановление базы данных.

Sql-запрос для восстановления:

use master restore database MBAF from disk=\'{openFileDialog.FileName}\' with replace

Листинг формирования запроса из кода программы:

if (openFileDialog.ShowDialog() == DialogResult.OK)

{ DBObject.context.Database.ExecuteSqlCommand(System.Data.Entity.TransactionalBehavior.DoNotEnsureTransaction, $"use master restore database MBAF from disk=\'{openFileDialog.FileName}\' with replace");

MessageBox.Show("База данных успешно восстановлена из резервной копии!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

# 

# ИМПОрт и экспорт данных

## Импорт данных из MS Excel

За мастер импорта/экспорта отвечает кнопка, которая запускает процесс DTSWizard.exe.

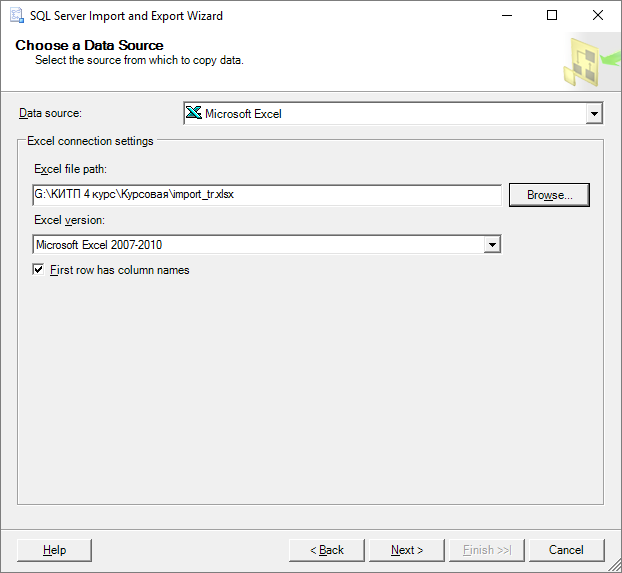


Рисунок 4.2 – Выбор источника импорта данных

На следующей странице нужно указать конечную цель импорта. В данном случае, это база данных курсовой работы.

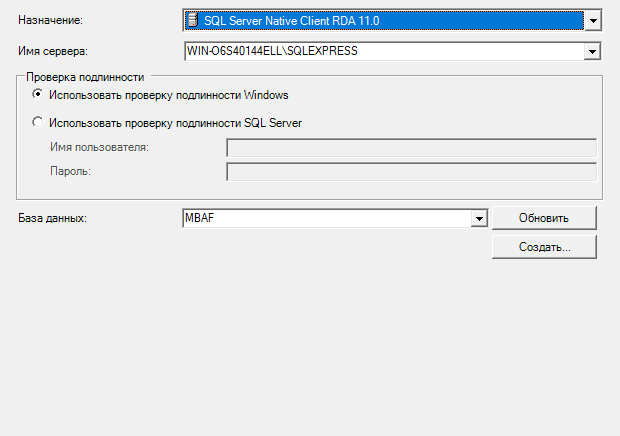


Рисунок 4.3 – Место назначения импорта данных

На следующей странице необходимо сопоставить страницы MS Excel и таблицы базы данных, для которых эти страницы предназначены.

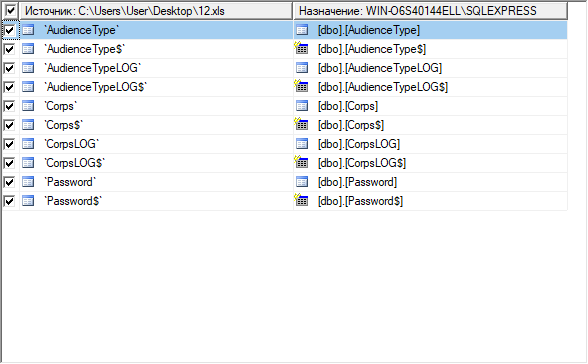


Рисунок 4.4 – Сопоставление страниц с таблицами

По завершению импорта данных, мастер импорта/экспорта уведомит пользователя о том, что импорт данных произошел успешно.

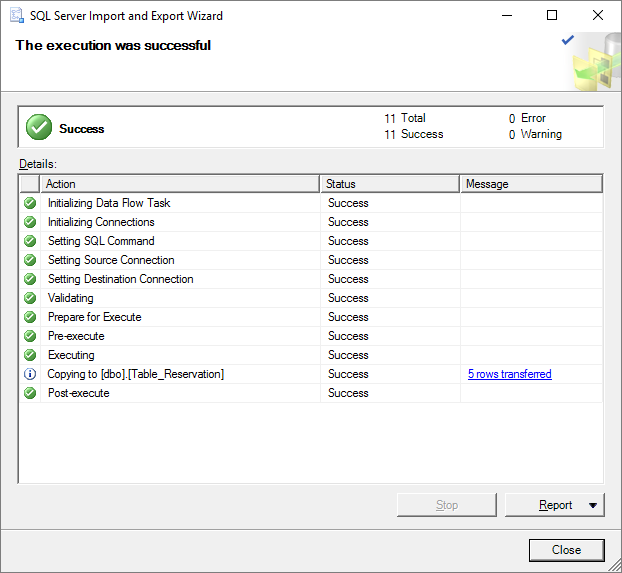


Рисунок 4.4 – Завершение импорта данных

## 

## Импорт данных с помощью BULK INSERT

Sql-скрипт, который импортирует данные с помощью конструкции BULK INSERT формируется следующим образом:

if (openFileDialog.ShowDialog() == DialogResult.OK)

{

DBObject.context.Database.ExecuteSqlCommand(System.Data.Entity.TransactionalBehavior.DoNotEnsureTransaction, $"bulk insert Reservation from \'{openFileDialog.FileName}\' with(keepidentity, fieldterminator=\';\', rowterminator=\'\\n\'");

MessageBox.Show("Данные успешно импортированы из файла!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

## Экспорт данных в csv

Согласно заданию, необходимо выполнить экспорт в csv файл.

Для того, чтобы произвести экспорт данных в указанный файл нужно указать для какой таблицы будет выполняться команда, а после указать файл, в который будет записываться результат.

Пример экспорта для таблицы “Teacher” выглядит следующим образом:

Если запустить данный файл, то в директории, где он лежит можно найти файл Reservation\_Export.csv со следующим содержимым:

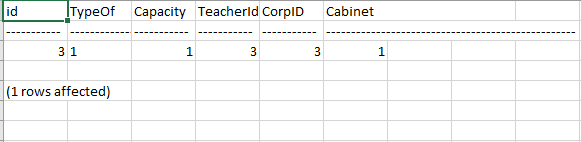


Рисунок 4.5 – Пример экспортированных данных

# 

# Управление доступом

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

* Root, который имеет неограниченные возможности по работе с данными в базе данных.
* User, основной задачей которого является лишь внесение данных в таблицы базы данных. Соответственно, для него ограничена возможность удаления данных из базы.

Для каждой из ролей были созданы два пользователя со своими логинами и паролями соответственно.

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

# 

# 6 разработка клиентского приложения

## Структура приложения

Структура приложения состоит из следующих слоев:

* слой пользователя;
* слой логики;
* слой данных.

Слой пользователя представляет собой формы с определенными контроллерами (таблицы, кнопки, поля ввода и т.д.), которые реализованы с помощью Windows Forms.

Слой логики отвечает за взаимодействия пользователя с базами данных. На этом уровне обрабатываются данные, которые предоставляются пользователю или базе данных. На этом уровне происходит проверка входных данных, чтобы все корректно вносилось в базу данных.

Слой данных представляет собой модель ранее созданной базы данных, который полностью копирует модель, реализованную в SSMS. В приложении этот слой реализован с помощью Entity Framework.

## Пользовательский интерфейс

При запуске приложения перед пользователем появляется окно, которое содержит:

* строку поиска;
* таблицу со всеми аудиториями;
* Кнопку вызова форму администратора;
* ToolStrip для работы с таблицей;

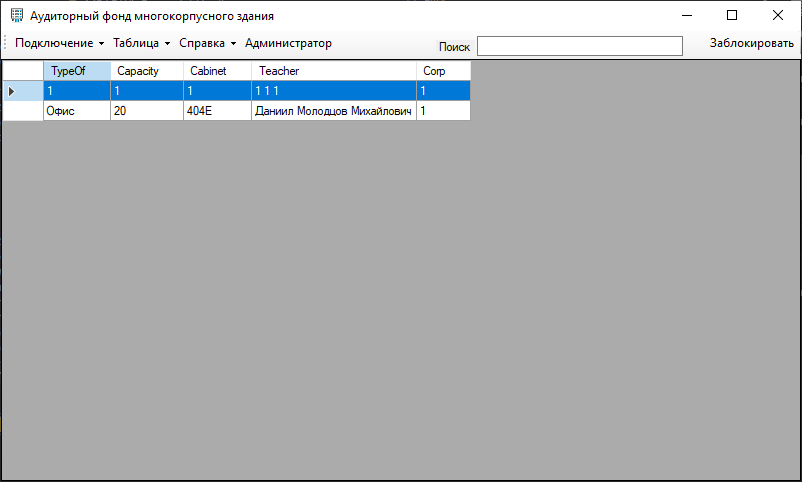


Рисунок 6.1 – Главная форма

Для того чтобы вывести информацию об аудитории, необходимо ввести в поля поиска любую информацию по аудитории. Для примера выведем информацию по аудиториям, которые находятся в корпусе 1. Перед нами откроется следующая форма:

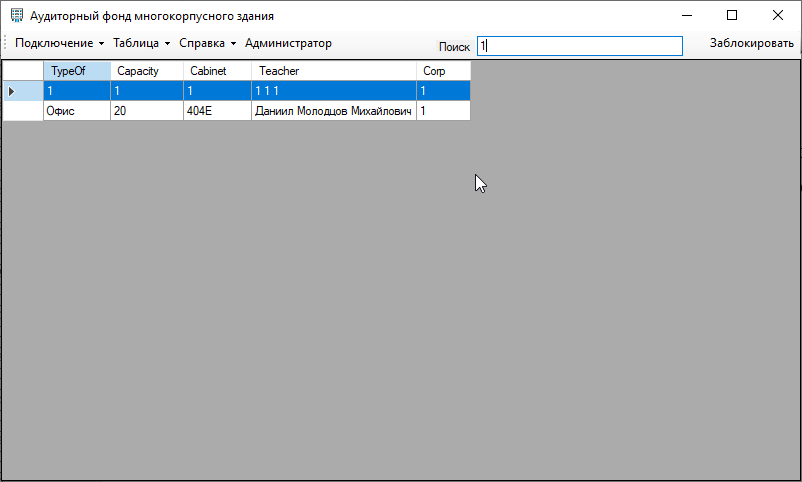


Рисунок 6.2 – Информация об аудиториях

Для того чтобы узнать количество аудиторий, нужно зайти на административную форму и нажать на соответствующую кнопку.

Получим следующее окно:

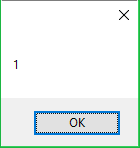


Рисунок 6.3 – Количество Записей

Двойной клик по записи в таблице с аудиториями приведет к тому, что откроется форма подробной информации по аудитории со всеми полями и таблице, в которой будет отображаться список всей информации для этой аудитории.

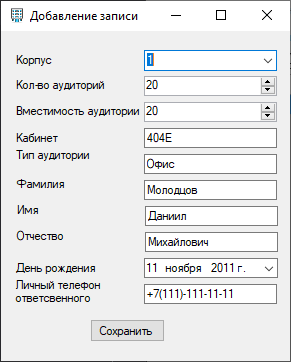


Рисунок 6.4 – Подробная информация по аудитории

На данной форме можно менять характеристики аудитории, а также добавить новую аудиторию для него нажатием кнопки «Сохранить».

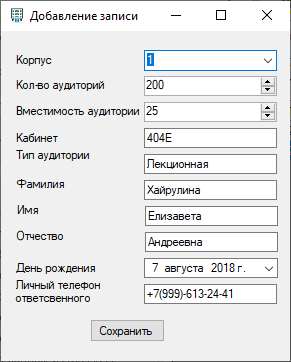


Рисунок 6.4 – Добавление аудитории

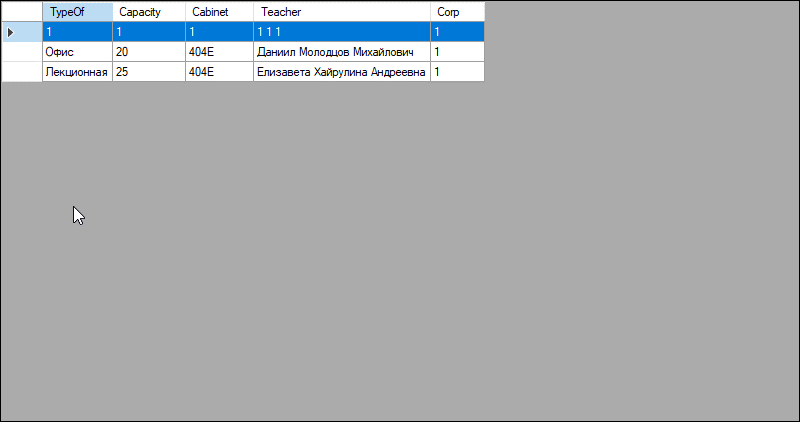


Рисунок 6.5 – Запись добавлена в таблицу

На главной форме можно удалить запись об аудитории. Для этого надо кликнуть левой кнопкой мыши, а затем в ToolStrip выбрать “удалить аудиторию”.

Для примера, удалим ранее добавленную запись.

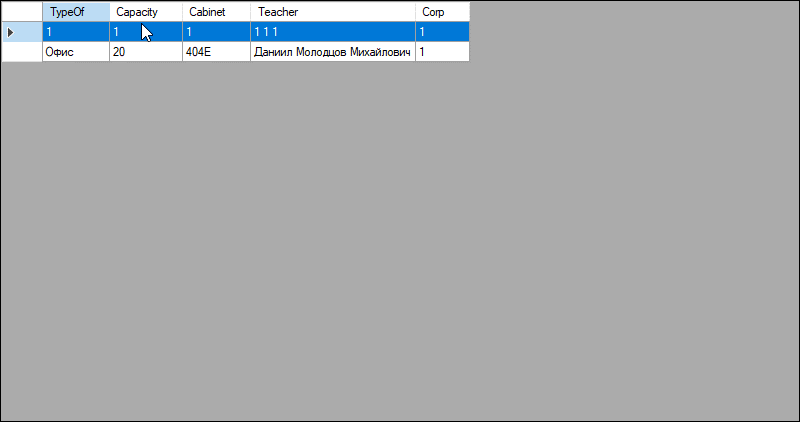


Рисунок 6.6 – Запись удалена

Так же с помощью показанных выше форм можно изменять информацию об аудитории (изменить описание, дату, телефон).

# заключение

Результатом выполнения данной работы стала база данных, позволяющая организовать работу системы учёта аудиторий. Разработанная модель 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.

Программа разработана таким образом, чтобы избежать возможного появления ошибок при работе с сервером базы данных. Там, где они могут встретиться, это предусмотрено и обрабатывается должным образом, а пользователь будет об этом уведомлен.

Работа была выполнена в среде программирования Visual Studio 2019 и Microsoft SQL Server Management Studio 18.2.

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

# приложение а

Создание БД

create table Teacher

(

Id int primary key identity(1,1),

Fname text not null default 'None',

Mname text not null default 'None',

Lname text not null default 'None',

Phone text not null default 'None',

Birthday Datetime

)

go

create table AudienceType

(

Id int primary key identity(1,1),

TypeOf text not null default 'None',

Capacity int not null default 0,

ResponsibleID int foreign key references Teacher(Id)

)

go

create table corps

(

Id int primary key identity(1,1),

CorpNumber text not null default '0A',

NumberOfAudiences int default 0,

)

go

# приложение Б

Заполнение таблиц

insert into AudienceType(TypeOf, Cabinet,Capacity,TeacherId,CorpID)

values

('Лабаратория',1,1,1,201),

('Лабаратория',2,1,1,202),

('Лабаратория',3,1,1,203),

('Лабаратория',4,1,1,204),

('Лабаратория',5,1,1,205),

('Лабаратория',6,1,1,206),

('Лабаратория',7,1,1,207),

('Лабаратория',8,1,1,208),

('Лабаратория',9,1,1,209),

('Лабаратория',10,1,1,2010);

insert into Corps(CorpNumber,NumberOfAudiences)

values

('1A',200),

('2A',200),

('3A',200),

('4A',200),

('5A',200),

('6A',200),

('7A',200),

('8A',200),

('9A',200),

('10A',200);

insert into Teacher(Fname,Mname,Lname,Phone,Birthday)

values

('Августин','Петров','Михайлович','+79996132441',CONVERT(datetime,'02.01.2001')),

('Августина','Петрова','Михайловна','+79996132441',CONVERT(datetime,'02.01.2001')),

('Юлий','Гай','Цезарь','+79996132441',CONVERT(datetime,'02.01.2001')),

('Исаак','Ньютон','СЭР','+79996132441',CONVERT(datetime,'02.01.2001')),

('Будда ','Шакьямуни','Буддович','+79996132441',CONVERT(datetime,'02.01.2001')),

('Цай ','Лунь','Чай','+79996132441',CONVERT(datetime,'02.01.2001')),

('Галилей','Галилео','Галилеович','+79996132441',CONVERT(datetime,'02.01.2001')),

('Тестер1','Тестер1','тестер1','+79996132441',CONVERT(datetime,'02.01.2001')),

('Тестер2','Тестер2','Тестер2','+79996132441',CONVERT(datetime,'02.01.2001')),

('Тестер3','Тестер3','Тестер3','+79996132441',CONVERT(datetime,'02.01.2001')),

('Августин','Петров','Михайлович','+79996132441',CONVERT(datetime,'02.01.2001'));

# приложение В

Создание дополнительных объектов

--LOG TABLE -------------------------------------------

create table AudienceTypeLOG

(

Id int,

TypeOf text ,

Capacity int,

ResponsibleID int,

operation VARCHAR(20),

operation\_dt DATETIME ,

operation\_user VARCHAR(255)

)

go

create table CorpsLOG

(

Id int ,

CorpNumber text,

NumberOfAudiences int,

operation VARCHAR(20),

operation\_dt DATETIME ,

operation\_user VARCHAR(255)

)

go

create table TeacherLOG

(

Id int ,

Fname text,

Mname text,

Lname text,

Phone text,

Birthday Datetime,

operation VARCHAR(20),

operation\_dt DATETIME ,

operation\_user VARCHAR(255)

)

go

---------------------TRIGERS--------------------------------------------

create trigger AudienceType\_u on AudienceType

after update

as

begin

INSERT INTO AudienceTypeLOG

--LOG old string

SELECT id,

TypeOf,

Capacity,

ResponsibleID,

'update\_old',

Getdate(),

Suser\_name()

from deleted

--LOG new string

INSERT INTO AudienceTypeLOG

SELECT id,

TypeOf,

Capacity,

ResponsibleID,

'update\_new',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER AudienceType\_i ON AudienceType

after INSERT

AS

BEGIN

INSERT INTO AudienceTypeLOG

SELECT id,

TypeOf,

Capacity,

ResponsibleID,

'insert',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER Corps\_d ON corps

after DELETE

AS

BEGIN

INSERT INTO TeacherLOG

SELECT id,

TypeOf,

Capacity,

ResponsibleID,

'delete',

Getdate(),

Suser\_name()

FROM deleted

END

go

---------------CORPS--------------------------------------

create trigger Corps\_u on corps

after update

as

begin

INSERT INTO TeacherLOG

--LOG old string

SELECT id,

CorpNumber,

NumberOfAudiences,

'update\_old',

Getdate(),

Suser\_name()

from deleted

--LOG new string

INSERT INTO TeacherLOG

SELECT id,

CorpNumber,

NumberOfAudiences,

'update\_new',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER Corps\_i ON corps

after INSERT

AS

BEGIN

INSERT INTO TeacherLOG

SELECT id,

CorpNumber,

NumberOfAudiences,

'insert',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER Corps\_d ON corps

after DELETE

AS

BEGIN

INSERT INTO TeacherLOG

SELECT id,

CorpNumber,

NumberOfAudiences,

'delete',

Getdate(),

Suser\_name()

FROM deleted

END

go

-----------TEACHER---------------------------

create trigger Teacher\_u on Teacher

after update

as

begin

INSERT INTO TeacherLOG

--LOG old string

SELECT id,

Fname,

Mname,

Lname,

Phone,

Birthday,

'update\_old',

Getdate(),

Suser\_name()

FROM deleted

--LOG new string

INSERT INTO TeacherLOG

SELECT id,

Fname,

Mname,

Lname,

Phone,

Birthday,

'update\_new',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER Teacher\_i ON Teacher

after INSERT

AS

BEGIN

INSERT INTO TeacherLOG

SELECT id,

Fname,

Mname,

Lname,

Phone,

Birthday,

'insert',

Getdate(),

Suser\_name()

FROM inserted

END

go

CREATE TRIGGER Teacher\_d ON Teacher

after DELETE

AS

BEGIN

INSERT INTO TeacherLOG

SELECT id,

Fname,

Mname,

Lname,

Phone,

Birthday,

'delete',

Getdate(),

Suser\_name()

FROM deleted

END

go

SET ansi\_padding OFF

go

Создание ролей и пользователей

use MBAF

go

create login userlogin

with password = '123';

create user user1

for login userlogin

create role root authorization userlogin

create login notrootlogin

with password = '123';

create user nonroot

for login notrootlogin

create role authorization notrootlogin

deny delete

to notrootlogin;

# приложение г

Код основной формы

using System;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Data.Entity;

using MBAF.EntityModel;

namespace MBAF

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

this.CenterToScreen();

DGVRefresh();

}

void DGVRefresh()

{

MainDataGridView.DataSource = DBObject.context.AudienceType.ToList();

MainDataGridView.Columns["id"].Visible = false;

MainDataGridView.Columns["Teacherid"].Visible = false;

MainDataGridView.Columns["Corpid"].Visible = false;

}

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

{

DGVRefresh();

}

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

{

MainDataGridView.DataSource = null;

}

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

{

if (MainDataGridView.DataSource != null)

{

DGVRefresh();

}

}

void ShowAddAudience(AudienceType audience,Teacher teacher,Corps corps)

{

WinForms.AddAudience add = null;

if (MainDataGridView.DataSource != null)

{

add = new WinForms.AddAudience(audience,teacher,corps);

add.ShowDialog();

add.Dispose();

DGVRefresh();

}

else

MessageBox.Show("Таблица не подключена! Пожалуйста подключите таблицу и попробуйте еще раз", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

void DeleteRow()

{

int? id = 0;

if (MainDataGridView.DataSource != null)

if (MainDataGridView.CurrentRow != null)

{

id = Convert.ToInt32(MainDataGridView.CurrentRow.Cells[0].Value);

if (id != null)

{

EntityModel.AudienceType audience = DBObject.context.AudienceType.Where(c => c.Id == id).FirstOrDefault();

DBObject.context.Corps.Remove(audience.Corp);

DBObject.context.Teachers.Remove(audience.Teacher);

DBObject.context.AudienceType.Remove(audience);

DBObject.context.SaveChanges();

DGVRefresh();

}

else

MessageBox.Show("Таблица не подключена! Пожалуйста подключите таблицу и попробуйте еще раз", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

else

MessageBox.Show("Запись не выбрана! Пожалуйста выберете запись и попробуйте еще раз", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

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

{

AudienceType audience=new AudienceType();

Teacher teacher = new Teacher();

Corps corps = new Corps();

ShowAddAudience(audience, teacher, corps);

}

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

{

DeleteRow();

}

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

{

WinForms.About about = new WinForms.About();

about.ShowDialog();

about.Dispose();

}

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

{

WinForms.Administrativ.AdminForm admin = new WinForms.Administrativ.AdminForm();

WinForms.Administrativ.UnlockForm unlock = new WinForms.Administrativ.UnlockForm();

unlock.ShowDialog();

unlock.Dispose();

admin.ShowDialog();

admin.Dispose();

}

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

{

WinForms.Administrativ.UnlockForm unlock = new WinForms.Administrativ.UnlockForm();

unlock.ShowDialog();

unlock.Dispose();

}

private void MainDataGridView\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

AudienceType audiencetype = (AudienceType)MainDataGridView.Rows[e.RowIndex].DataBoundItem;

Teacher teacher = audiencetype.Teacher;

Corps corps = audiencetype.Corp;

ShowAddAudience(audiencetype,teacher, corps);

}

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

{

MainDataGridView.DataSource = DBObject.context.AudienceType.Where(c => c.Teacher.Lname.Contains(searchTextBox.Text)|| c.Teacher.Fname.Contains(searchTextBox.Text) || c.Teacher.Mname.Contains(searchTextBox.Text) || c.TypeOf.Contains(searchTextBox.Text) || c.Corp.CorpNumber.ToString().Contains(searchTextBox.Text) || c.Capacity.ToString().Contains(searchTextBox.Text) ||c.Cabinet.Contains(searchTextBox.Text)).ToList();

}

}

}

# приложение д

Код формы редактирования и добавления аудиторий

using MBAF.EntityModel;

using System;

using System.Data;

using System.Linq;

using System.Windows.Forms;

namespace MBAF.WinForms

{

public partial class AddAudience : Form

{

private readonly AudienceType audience;

private readonly Teacher teacher;

private readonly Corps corps;

public AddAudience(AudienceType audtype,Teacher teacher,Corps corp)

{

this.audience = audtype;

this.teacher = teacher;

this.corps = corp;

InitializeComponent();

CorpsComboBox.DataSource = DBObject.context.Corps.Select(c => c.CorpNumber).ToList();

CorpsComboBox.SelectedIndex = -1;

if (DBObject.context.AudienceType.Where(c=>c.Id==audtype.Id).Count()>0)

Fill();

}

void Fill()

{

CorpsComboBox.DataSource = DBObject.context.Corps.Select(c => c.CorpNumber).ToList();

CorpsComboBox.SelectedItem = corps.CorpNumber.FirstOrDefault();

AuditorNumericUpDown.Value = corps.NumberOfAudiences;

AuditoryCapacityNumericUpDown.Value = audience.Capacity;

AudTypeTextBox.Text = audience.TypeOf;

CabinetTextBox.Text = audience.Cabinet;

MnameTextBox.Text = teacher.Mname;

FnameTextBox.Text = teacher.Fname;

LnameTextBox.Text = teacher.Lname

BirthDateTimePicker.Value = teacher.Birthday;

PhoneMaskedTextBox.Text = teacher.Phone;

}

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

{

try

{

teacher.Mname = MnameTextBox.Text;

teacher.Fname = FnameTextBox.Text;

teacher.Lname = LnameTextBox.Text;

teacher.Phone = PhoneMaskedTextBox.Text;

teacher.Birthday = BirthDateTimePicker.Value;

corps.CorpNumber = CorpsComboBox.Text;

corps.NumberOfAudiences = (int)AuditorNumericUpDown.Value;

audience.Cabinet = CabinetTextBox.Text;

audience.TypeOf = AudTypeTextBox.Text;

audience.Capacity = (int)AuditoryCapacityNumericUpDown.Value;

audience.Teacherid = teacher.Id;

audience.Corpid = corps.Id;

if (DBObject.context.AudienceType.Where(c => c.Cabinet == audience.Cabinet).Count() ==0 && DBObject.context.Teachers.Where(c => c.ToString() == teacher.ToString()).Count() == 0 && DBObject.context.Corps.Where(c => c.CorpNumber == corps.CorpNumber).Count() == 0)

{

DBObject.context.Corps.Add(corps);

DBObject.context.Teachers.Add(teacher);

DBObject.context.AudienceType.Add(audience);

MessageBox.Show("Запись добавлена!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

DBObject.context.SaveChanges();

this.Dispose();

}

catch (Exception exc)

{

MessageBox.Show("Проверьте введеные данные!\n\n"+exc.Message, "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Error);

}