**МиНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное образовательное учреждение

высшего профессионального образования

**«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ**

**УНИВЕРСИТЕТ им. В.Г. ШУХОВА»**

**(БГТУ им. В.Г. Шухова)**

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Базы данных»

предметная область: Приёмная комиссия

|  |
| --- |
| Выполнила: студентка  группы ПВ-31  Браткова Ирина Олеговна  Проверил: старший  преподаватель Гарибов А.И. |

Белгород 2018

Оглавление

[Постановка задачи 3](#_Toc533718562)

[Описание предметной области 3](#_Toc533718563)

[Проектирование информационной системы 4](#_Toc533718564)

[Разработка приложения 8](#_Toc533718565)

[Описание классов для работы с базой данных 8](#_Toc533718566)

[Представления 9](#_Toc533718567)

[Хранимые процедуры и транзакции 10](#_Toc533718568)

[Триггеры 12](#_Toc533718569)

[Импорт и экспорт данных. XSD-схема 12](#_Toc533718570)

[Создание отчётов 13](#_Toc533718571)

[Список литературы 14](#_Toc533718572)

[Приложение A – Запросы на формирование базы данных 15](#_Toc533718573)

[Приложение B – Листинг программы 17](#_Toc533718574)

[Приложение С – XML-файл 21](#_Toc533718575)

## Постановка задачи

Исследовать предметную область «приёмная комиссия». Составить описание предметной области, основываясь на проведенном исследовании. Составить надлежащую модель данных. Разработать соответствующую структуру базы данных и реализовать её с помощью средств Microsoft SQL Server. Разработать пользовательский интерфейс для работы с базой данных.

## Описание предметной области

Приёмная комиссия – структурное подразделение высших и специальных учебных заведений. В данном случае, рассмотрена работа приёмной комиссии ВУЗа.

Абитуриент при поступлении должен указать ФИО, паспортные данные, адрес, дату рождения, контактные данные, необходимость общежития, результаты экзаменов, а так же направление, на котором он хотел бы обучаться. Направление закреплено за определенной кафедрой. На одной кафедре может быть несколько направлений. За направлением закреплены форма и основа обучения.

Абитуриент может написать согласие на зачисления и оставить оригиналы заявлений. Согласие и оригинал могут храниться только в одном экземпляре.

Для того чтобы подать заявления на определенное направление, абитуриент должен был сдавать экзамены по указанным для направления дисциплинам. Для участия в конкурсе, абитуриенту необходимо набрать минимальные проходные баллы.

Абитуриент может подать заявления три раза. Если абитуриент отзовет заявление, то попытка для подачи заявления не восстанавливается.

## 

## Проектирование информационной системы

Основными сущностями для хранения информации являются:

– Заявление;

– Абитуриент;

– Форма обучения;

– Основа обучения;

– Личные достижения;

– Достижения абитуриента;

– Баллы;

– Минимальный проходной балл;

– Дисциплина;

– Места;

– Направление;

– Кафедра.

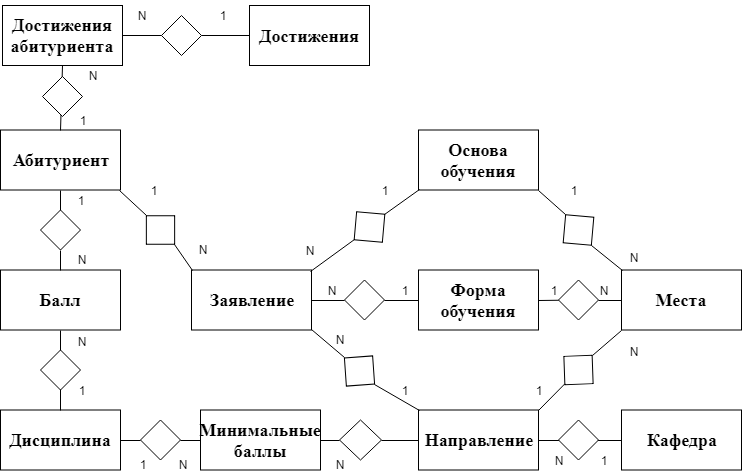


Рисунок 1 – Диаграмма «Сущность – связь»

Также необходимо выделить атрибуты сущностей. Атрибуты - это существенные свойства объекта, интересующие пользователя. Каждая из введенных сущностей определяется набором свойств (атрибутов), которые позволяют отличать конкретные экземпляры. Таким образом получим:

– Заявление (Statement): cогласие на зачисление, оригинал документа;

– Абитуриент (Entrant): ФИО, паспортные данные, адрес, дата рождения, контактные данные, необходимость общежития;

– Форма обучения (FormTraning): наименование;

– Основа обучения (BasisTraning): наименование;

– Кафедра (Department): наименование;

– Направление (Speciality): наименование;

– Места (Places): количество мест;

– Личные достижения (Achievement): наименование, доп.балл;

– Минимальный проходной балл (MinimumScore): балл;

– Дисциплина (Discipline): наименование;

–Достижения абитуриента (EntrantAchieve);

– Баллы (Point): балл.

На основе модели можно разработать структуру базы данных:

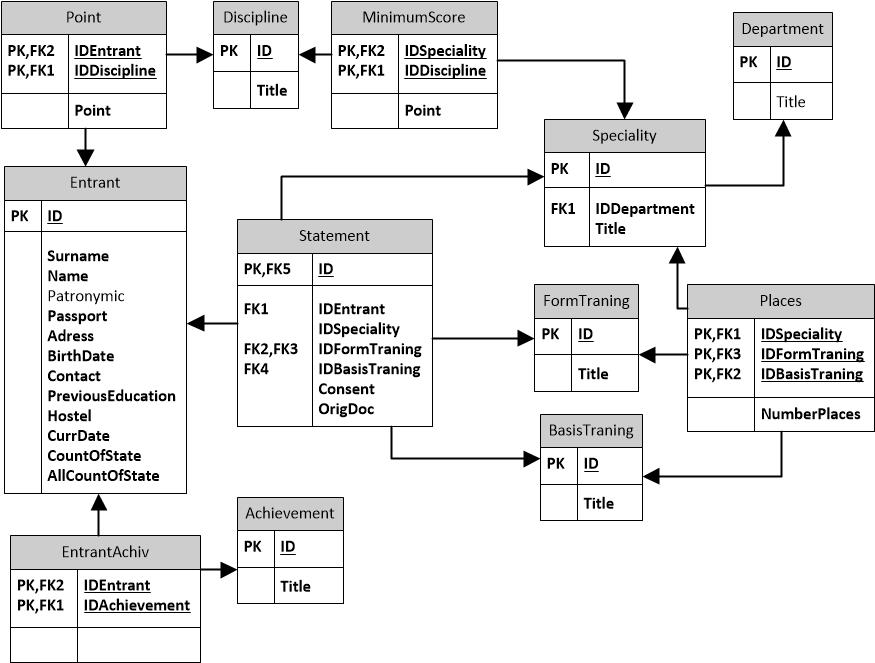


Рисунок 2 – Структура базы данных

Исходя из этого можно разработать структуру основных таблиц базы данных «Приёмная комиссия»:

Таблица 1 — Структура данных Statement

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| ID | Номер заявления | целое | Да |
| IDEntrant | Код абитуриента | целое | Да |
| IDSpeciality | Код направления | целое | Да |
| IDFormTraning | Код формы обучения | целое | Да |
| IDBasisTraning | Код основы обучения | целое | Да |
| Consent | Согласие на зачисление | логический | Да |
| OrigDoc | Оригинал документа | логический | Да |

Таблица 2 — Структура данных Entrant

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| ID | Код абитуриента | целое | Да |
| Surname | Фамилия | строка | Да |
| Name | Имя | строка | Да |
| Patronymic | Отчество | строка | Нет |
| Passport | Паспорт | строка | Да |
| Adress | Адрес | строка | Да |
| BirthDate | Дата рождения | дата | Да |
| Contact | Контактные данные | строка | Да |
| PreviousEducation | Пред. Образование | строка | Да |
| Hostel | Общежите | логический | Да |
| AllCountOfState | Всего заявлений | целое | Да |
| CountOfState | Сейчас заявлений | целое | Да |
| CurrDate | Дата добавления | дата | да |

Таблица 3 — Структура данных FormTraning

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| ID | Код формы обучения | целое | Да |
| Title | Наименование | строка | Да |

Таблица 4 — Структура данных BasisTraning

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| ID | Код оснвовы обучения | целое | Да |
| Title | Наименование | строка | Да |

Таблица 5 — Структура данных Department

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| ID | Код кафедры | целое | Да |
| Title | Наименование | строка | Да |

Таблица 6 — Структура данных Speciality

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| ID | Код специальности | целое | Да |
| IDDEpartment | Код кафедры | целое | Да |
| Title | Наименование | строка | Да |

Таблица 7 — Структура данных Places

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| IDSpeciality | Код специальности | целое | Да |
| IDFormTraning | Код формы обучения | целое | Да |
| IDBasisTraning | Код основы обучения | целое | Да |
| NumberPlaces | Количество мест | целое | Да |

Таблица 8 — Структура данных Achievement

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| ID | Код достижения | целое | Да |
| Title | Наименование | строка | Да |

Таблица 9 — Структура данных EntrantAchieve

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| IDEntrant | Код абитуриента | целое | Да |
| IDAchievement | Код достижения | целое | Да |

Таблица 10 — Структура данных MinimumScore

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| IDSpeciality | Код специальности | целое | Да |
| IDDiscipline | Код дисциплины | целое | Да |
| Point | Балл | целое | Да |

Таблица 11 — Структура данных Discipline

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| ID | Код дисциплины | целое | Да |
| Title | Наименование | строка | Да |

Таблица 12 — Структура данных Point

|  |  |  |  |
| --- | --- | --- | --- |
| **Имя** | **Описание** | **Тип** | **Обязательное поле** |
| IDEntrant | Код абитуриента | целое | Да |
| IDDiscipline | Код дисциплины | целое | Да |
| Point | Балл | целое | Да |

В данном курсовом проекте использовался Microsoft SQL Server 2016, что влечёт за собой использование синтаксических особенностей данного сервера.

Код SQL-запросов для создания базы данных и её таблиц представлен в приложении А.

## Разработка приложения

Взаимодействие базы данных, развёрнутой на базе MS SQL Server, с приложением организовано средствами языка С #.

В программе реализован подход (ORM), при котором на основе существующей базы данных разрабатываются классы, сопоставляющиеся с соответствующими таблицами базы. Эти классы можно изменять по своему усмотрению, добавляя в них необходимые CRUD-методы. CRUD – четыре базовые функции, используемые при работе с хранилищами данных: create, read, update, delete. Проведение аналогии с языком SQL позволяет заметить, что это методы: insert, select, update и delete.

Для подключения к базе данных используется класс DBConn, который представлен в листинге 1.

Листинг 1 — Класс DBConn

class DBConn

{

public void OpenConn(SqlConnectionStringBuilder sb, SqlConnection conn)

{

try

{

sb.DataSource = "DESKTOP-6637QU0";

sb.InitialCatalog = "BratckovaPV31Comission";

sb.IntegratedSecurity = true;

conn.ConnectionString = sb.ConnectionString;

conn.Open();

}

catch (System.Data.SqlClient.SqlException)

{

MessageBox.Show("Соединение не установлено", "Ошибка!");

}

}

}

## Описание классов для работы с базой данных

Для взаимодействия с базой данных «Приёмная комиссия» используются классы, именованные так же, как и таблицы, которые содержит данная база. В листинге 2 представлено описание наименьшего по объему класса.

Реализация методов исключена в целях экономии.

Листинг 2 — Класс Department

class Department

{

public int ID { get; set; }

public string Title { get; set; }

public static List<Department> getAll()

public static Department getByTitle(string id)

public static Department getById(int id)

public void insert()

public static void insert(Department dep)

public static Department insert (string title)

public bool update()

public bool delete()

}

Все остальные классы для работы с базой данных реализованы аналогичным образом. Эта информация вынесена в приложение В.

Для отображения информации базы данных в приложении используются элементы DataGridView, а так же TreeView и ListBox. В листинге 3 можно увидеть, как заполняется элемент типа TreeView информацией из двух связанных таблиц – Department и Speciality.

Листинг 3 — Заполнение дерева treeView\_InfoDep

private void showDepartmentTree()

{

treeView\_InfoDep.Nodes.Clear();

List<Department> dep = Department.getAll();

for (int i = 0; i < dep.Count; i++)

{

TreeNode tn = new TreeNode();

tn.Text = dep[i].Title;

showSpecialityTree(tn, dep[i].ID);

treeView\_InfoDep.Nodes.Add(tn);

}

}

private void showSpecialityTree(TreeNode root, int idd)

{

List<Speciality> dep = Speciality.getAll();

for (int i = 0; i < dep.Count; i++)

{

if (dep[i].Department.ID == idd)

{

TreeNode tn = new TreeNode();

tn.Text = dep[i].Title;

root.Nodes.Add(tn);

}

}

}

## Представления

Изначально на уровне СУБД, а затем и в программе были реализованы представления для отображения средних баллов по специальностям (AvgPointsBySpeciality) и для отображения полной информации о заявлении абитуриента (SpecialityAndPointEntrant), в котором отображается общий суммарный балл (баллы за экзамены и за личные достижения).

Первое представление можно увидеть в листинге 4, приведенном ниже.

Листинг 4 — Представление AvgPointsBySpeciality

CREATE OR ALTER VIEW AvgPointsBySpeciality AS

SELECT Sp.Title, ISNULL(AVG(Tmp.Summ), 0) as AvgPoint

FROM

Speciality AS Sp

LEFT JOIN(

SELECT E.ID, ISNULL(SUM(P.Point),0) As Summ, Sp.Title

FROM Entrant as E, Point as P, MinimumScore as M, Discipline as D, Speciality as Sp, Statement as S, FormTraning as Ft, BasisTraning as Bt

WHERE P.IDEntrant = E.ID AND M.IDDiscipline = D.ID AND P.IDDiscipline = D.ID AND Sp.ID = M.IDSpeciality AND S.IDEntrant = E.ID AND S.IDSpeciality = Sp.ID AND BT.ID = S.IDBasisTraning AND Ft.ID = S.IDFormTraning

GROUP BY E.ID, Sp.Title, Ft.ID, Bt.ID) as Tmp on Sp.Title = Tmp.Title

GROUP BY Sp.Title

## Хранимые процедуры и транзакции

В базе данных были реализованы хранимые процедуры – объекты базы данных, представляющие собой набор инструкций и хранящиеся на сервере.

Для базы данных «Приёмная комиссия» были разработаны хранимые процедуры:

– Удаление абитуриентов, у которых отсутствуют заявления (при условии, что новые заявления не могут быть поданы);

– Подсчёт баллов за личные достижения.

Вторая процедура используется для вывода суммы баллов в соответствующее текстовое поле в приложении.

Листинг данной процедуры представлен ниже.

Листинг 5 — Хранимая процедура AchievePointOfEntrant

CREATE PROCEDURE AchievePointOfEntrant @id INT, @count INT OUTPUT AS

BEGIN

SET @count = 0

DECLARE @cur CURSOR

DECLARE @tt VARCHAR(100)

DECLARE @pp INT

SET @cur = CURSOR FOR

SELECT A.Point

FROM Achievement as A, EntrantAchiv as EA, Entrant AS E

WHERE A.ID = EA.IDAchieve AND EA.IDEntrant = E.ID AND E.ID = @id

FOR READ ONLY

OPEN @cur

FETCH NEXT

FROM @cur

INTO @pp

WHILE @@FETCH\_STATUS = 0

BEGIN

SET @count = @count + @pp

FETCH NEXT

FROM @cur

INTO @pp

END

RETURN @count

END

Процедура удаления абитуриентов, у которых нет заявлений, вызывается нажатием на соответствующую кнопку.

Кроме того, данная процедура выполнена в рамках транзакции.

Листинг её вызова представлен ниже.

Листинг 6 — Хранимая процедура в рамках транзакции

public static bool delEntrantWithoutStatement()

{

DBConn dbc = new DBConn();

bool flag = false;

try

{

SqlConnectionStringBuilder sb = new SqlConnectionStringBuilder();

using (SqlConnection conn = new SqlConnection())

{

SqlCommand cmd = new SqlCommand();

dbc.OpenConn(sb, conn);

SqlTransaction trap = conn.BeginTransaction();

try

{

cmd.Transaction = trap;

cmd.Connection = conn;

cmd.CommandText = "SELECT COUNT(\*) FROM " +

"Entrant WHERE NowCount " +

"<= 0 and AllCount <= 0";

int count = (int)cmd.ExecuteScalar();

if (MessageBox.Show("Будет удалено абитуриентов: " +

"" + count.ToString() + ". " +

"Вы уверены?",

"Информация",

MessageBoxButtons.OKCancel)

== System.Windows.Forms.DialogResult.OK)

{

string str = @"DelEntrantWithoutStatement";

cmd.CommandType = CommandType.StoredProcedure;

cmd.CommandText = str;

cmd.Connection = conn;

cmd.ExecuteNonQuery();

trap.Commit();

return true;

}

else

{

trap.Rollback();

return false;

}

}

catch (Exception ex)

{

trap.Rollback();

throw ex;

}

}

}

catch (System.Data.SqlClient.SqlException)

{

MessageBox.Show("Ошибка соединения", "Ошибка!");

}

return flag;

}

## Триггеры

Триггеры (хранимые процедуры, вызываемые автоматически при возникновении определенного события) так же были реализованы для базы данных.

Примером может послужить триггер, представленный в листинге 7. Он необходим для изменения значений полей текущего количества заявлений и общего количества заявлений.

Данный триггер срабатывает при добавлении заявления в базу.

Листинг 6 — Триггер при добавлении записи в Statement

CREATE TRIGGER NewStatement

ON Statement

AFTER INSERT

AS

BEGIN

DECLARE @id INT

DECLARE cur CURSOR FOR

SELECT Stat.IDEntrant as Id

FROM inserted as Stat, Entrant as E

WHERE E.ID = Stat.IDEntrant

OPEN cur

FETCH NEXT FROM cur

INTO @id

WHILE @@FETCH\_STATUS = 0

BEGIN

UPDATE Entrant

SET NowCount = NowCount + 1, AllCount = AllCount - 1

WHERE Id = @id

FETCH NEXT FROM cur

INTO @id

END

CLOSE cur

DEALLOCATE cur

END

## Импорт и экспорт данных. XSD-схема

В программе реализован импорт и экспорт данных для нескольких таблиц в файл. Для этого были выбраны таблицы Entrant, Statement, Point и EntrantAchieve.

С помощью XSD-языка для описания структуры XML-файла была разработана соответствующая схема. Кроме того, были разработаны методы для реализации импорта и экспорта с учётом разработанной схемы, находящиеся в исходном коде программы.

XML-файл после его формирования можно увидеть в приложении С.

**Создание отчётов**

Очень часто в приёмной комиссии необходимо предоставлять информацию о заявлениях, поданных на различные специальности, например, директору института или заведующим кафедрами. С целью упрощения ведения отчётности в базе данных была реализована возможность формирования отчёта для всех специальностей или только для указанной специальности.

Выбор был сделан в пользу таблицы Statement именно по той причине, что она является «главной» таблицей базы данных и содержит в себе информацию об абитуриенте (ФИО и паспортные данные) из таблицы Entrant, наименование специальности (таблица Speciality), форму и основу обучения (FormTraning и BasisTraning).

Отчёт формируется средствами приложения в MS Excel.

На рисунке 3 можно увидеть отчёт после его создания.

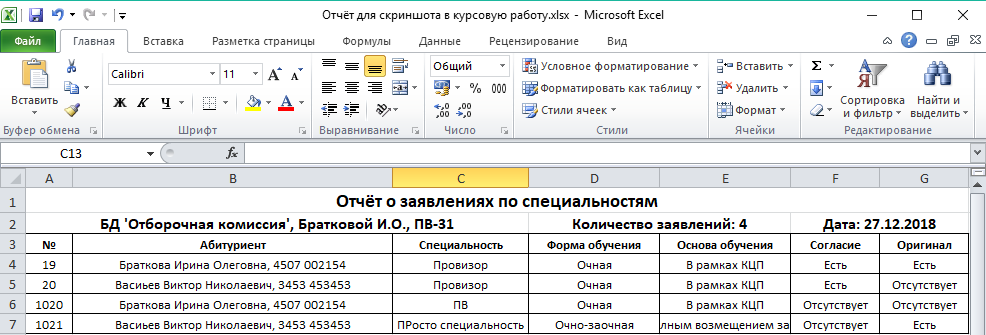


Рисунок 3 – Отчёт

Код формирования отчёта можно просмотреть в исходом коде данной программы.

## 

## Список литературы

1. Бен-Ган И. Microsoft SQL Server 2012. Основы T-SQL – М.: Эксмо, 2015. – 298 с.
2. Дейт К. Дж. SQL и реляционная теория. Как грамотно писать код на SQL. – М.: Символ-плюс, 2010. – 474 с.
3. Кириллов В.В., Громов Г.Ю. Введение в реляционные базы данных. – СПб.: БХВ-Петербург, 2009. – 464 с.
4. Котов О.М. Язык C#. Краткое описание и введение в технологии программирования. Учебное пособие. – Екатеринбург: Изд-во Урал.ун-та, 2014. – 208 с.
5. Лобел Л., Браст Дж. Э., Форте С. Разработка приложений на основе Microsoft SQL Server 2008. – СПб.: БХВ-Петербург, 2010. – 295 с.
6. Молинаро Э. SQL. Сборник рецептов. – СПб.: Символ-Плюс, 2009. – 349 с.
7. Сарка Д., Талмейдж Р., Бен-Ган И. Microsoft SQL Server 2012. Создание запросов. – М.: Изд-во «Русская редакция», 2014. – 347 с.
8. Хендерсон К. Microsoft SQL Server: структура и реализация. Профессиональное руководство. – СПб.: Вильямс, 2012. – 1056 с.
9. Экспорт и импорт данных из/в Excel. [Электронный ресурс] URL: <https://www.kv.by/content/321757-realnoe-programmirovanie-eksport-i-import-dannykh-izv-excel> (дата обращения 20.12.2018)
10. Microsoft Developer Network. [Электронный ресурс] URL: <https://msdn.microsoft.com/ru-ru/library/system.configuration.configurationmanager(v=vs.110).aspx> (дата обращения 15.12.2018)

## Приложение A – Запросы на формирование базы данных

CREATE TABLE BasisTraning(

ID INT IDENTITY (1,1) PRIMARY KEY,

Title VARCHAR(100) NOT NULL CHECK (Title <> '')

)

CREATE TABLE FormTraning(

ID INT IDENTITY (1,1) PRIMARY KEY,

Title VARCHAR(100) NOT NULL CHECK (Title <> '')

)

CREATE TABLE Department(

ID INT IDENTITY(1,1) PRIMARY KEY,

Title VARCHAR(100) NOT NULL CHECK (Title <> '')

)

CREATE TABLE Discipline(

ID INT IDENTITY(1,1) PRIMARY KEY,

Title VARCHAR(100) NOT NULL CHECK (Title <> '')

)

CREATE TABLE Speciality(

ID INT IDENTITY(1,1) PRIMARY KEY,

IDDepartment INT NOT NULL FOREIGN KEY REFERENCES Department(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

Title VARCHAR(100) NOT NULL CHECK (Title <> '')

)

CREATE TABLE Places(

IDSpeciality INT NOT NULL FOREIGN KEY REFERENCES Speciality(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

IDFormTraning INT NOT NULL FOREIGN KEY REFERENCES FormTraning(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

IDBasisTraning INT NOT NULL FOREIGN KEY REFERENCES BasisTraning(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

NumberPlaces INT NOT NULL CHECK (NumberPlaces > 0)

PRIMARY KEY (IDSpeciality, IDFormTraning, IDBasisTraning)

)

CREATE TABLE MinimumScore(

IDDiscipline INT NOT NULL FOREIGN KEY REFERENCES Discipline(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

IDSpeciality INT NOT NULL FOREIGN KEY REFERENCES Speciality(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

Point INT NOT NULL CHECK (Point>= 20)

PRIMARY KEY (IDSpeciality, IDDiscipline)

)

CREATE TABLE Achievement(

ID INT IDENTITY(1,1) PRIMARY KEY,

Title VARCHAR(100) NOT NULL CHECK (Title <> ''),

Point INT NOT NULL CHECK(Point >=1 AND Point <= 10)

)

CREATE TABLE Entrant(

ID INT IDENTITY(1,1) PRIMARY KEY,

Surname VARCHAR(30) NOT NULL CHECK(Surname <> ''),

IName VARCHAR(30) NOT NULL CHECK(IName <> ''),

Patronymic VARCHAR(30) NULL,

Passport VARCHAR(15) NOT NULL CHECK(Passport <> ''),

Adress VARCHAR(50) NOT NULL CHECK(Adress <> ''),

BirthDate DATE NOT NULL,

Contact CHAR(15) NULL,

PreviousEducation VARCHAR(100) NOT NULL CHECK(PreviousEducation <> ''),

Hostel BIT NOT NULL DEFAULT 0,

CurrDate DATE NOT NULL DEFAULT GETDATE()

)

CREATE TABLE Point (

IDDiscipline INT NOT NULL FOREIGN KEY REFERENCES Discipline(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

IDEntrant INT NOT NULL FOREIGN KEY REFERENCES Entrant(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

Point INT NOT NULL CHECK (Point >= 20)

PRIMARY KEY (IDDiscipline, IDEntrant)

)

CREATE TABLE EntrantAchiv(

IDEntrant INT NOT NULL FOREIGN KEY REFERENCES Entrant(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

IDAchieve INT NOT NULL FOREIGN KEY REFERENCES Achievement(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

PRIMARY KEY (IDEntrant, IDAchieve)

)

CREATE TABLE Statement(

ID INT IDENTITY(1,1) PRIMARY KEY,

IDEntrant INT NOT NULL FOREIGN KEY REFERENCES Entrant(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

IDSpeciality INT NOT NULL FOREIGN KEY REFERENCES Speciality(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

IDFormTraning INT NOT NULL FOREIGN KEY REFERENCES FormTraning(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

IDBasisTraning INT NOT NULL FOREIGN KEY REFERENCES BasisTraning(ID) ON DELETE CASCADE ON UPDATE NO ACTION,

Consent BIT NOT NULL DEFAULT 0,

OriginalDocument BIT NOT NULL DEFAULT 0

)

## 

## Приложение B – Листинг программы

Класс MinimumScore

class Places

{

public int CountPlaces { get; set; }

public Speciality Speciality { get; set; }

public FormTraning FormTraning { get; set; }

public BasisTraning BasisTraning { get; set; }

public static List<Places> getBySpec(int id)

{

List<Places> list = new List<Places>();

DBConn dbc = new DBConn();

try

{ SqlConnectionStringBuilder sb = new SqlConnectionStringBuilder();

using (SqlConnection conn = new SqlConnection())

{

dbc.OpenConn(sb, conn);

string str = "SELECT DISTINCT " +"IDSpeciality, IDFormTraning, " +

"IDBasisTraning, NumberPlaces " + "FROM Places WHERE " +

"IDSpeciality = @id";

SqlCommand cmd = new SqlCommand(str, conn);

cmd.Parameters.AddWithValue("id", id);

using (SqlDataReader reader = cmd.ExecuteReader())

{ while (reader.Read())

{ Places plas = new Places();

plas.CountPlaces =

Convert.ToInt32(reader["NumberPlaces"]);

plas.Speciality =

Speciality.getByIdA(Convert.ToInt32(reader["IDSpeciality"]));

plas.FormTraning =

FormTraning.getById(Convert.ToInt32(reader["IDFormTraning"]));

plas.BasisTraning =

BasisTraning.getById(Convert.ToInt32(reader["IDBasisTraning"]));

list.Add(plas);

}

}

conn.Close();

}

}

catch (System.Data.SqlClient.SqlException)

MessageBox.Show("Ошибка соединения", "Ошибка!!2!");

return list;

}

public static Places getBy3id(int ids, int idb, int idf)

{

DBConn dbc = new DBConn();

try

{

SqlConnectionStringBuilder sb = new SqlConnectionStringBuilder();

using (SqlConnection conn = new SqlConnection())

{ dbc.OpenConn(sb, conn);

string str = "SELECT DISTINCT IDSpeciality, " +

"IDFormTraning, IDBasisTraning, " +

"NumberPlaces FROM Places " +"WHERE IDSpeciality = @id " +

"AND IDFormTraning = @idf " +"AND IDBasisTraning = @idb";

SqlCommand cmd = new SqlCommand(str, conn);

cmd.Parameters.AddWithValue("id", ids);

cmd.Parameters.AddWithValue("idf", idf);

cmd.Parameters.AddWithValue("idb", idb);

SqlDataReader reader = cmd.ExecuteReader();

if (reader.HasRows)

{ reader.Read();

Places dep = new Places();

dep.CountPlaces =

Convert.ToInt32(reader["NumberPlaces"]);

dep.BasisTraning =

BasisTraning.getById(Convert.ToInt32(reader["IDBasisTraning"]));

dep.FormTraning =

FormTraning.getById(Convert.ToInt32(reader["IDFormTraning"]));

dep.Speciality =

Speciality.getByIdA(Convert.ToInt32(reader["IDSpeciality"]));

conn.Close();

return dep;

}

conn.Close();

}

}

catch (System.Data.SqlClient.SqlException)

MessageBox.Show("Ошибка соединения", "Ошибка!!2!");

return null;

}

public bool delete()

{

DBConn dbc = new DBConn();

try

{ SqlConnectionStringBuilder sb = new SqlConnectionStringBuilder();

using (SqlConnection conn = new SqlConnection())

{ dbc.OpenConn(sb, conn);

string str = "DELETE FROM Places " +"WHERE IDSpeciality = @ids " +

"AND IDFormTraning = @idf " +"AND IDBasisTraning = @idb";

SqlCommand cmd = new SqlCommand(str, conn);

cmd.Parameters.AddWithValue("ids", Speciality.ID);

cmd.Parameters.AddWithValue("idf", FormTraning.ID);

cmd.Parameters.AddWithValue("idb", BasisTraning.ID);

cmd.ExecuteNonQuery();

conn.Close();

return true;

}

}

catch (System.Data.SqlClient.SqlException)

{

MessageBox.Show("Ошибка соединения", "Ошибка!");

return false;

}

}

public void insert()

{

DBConn dbc = new DBConn();

try

{ SqlConnectionStringBuilder sb = new SqlConnectionStringBuilder();

using (SqlConnection conn = new SqlConnection())

{ dbc.OpenConn(sb, conn);

string str = "INSERT INTO Places " +"(IDSpeciality, IDFormTraning, " +

"IDBasisTraning, NumberPlaces) " + "VALUES(@ids, @idf, @idb, @num)";

SqlCommand cmd = new SqlCommand(str, conn);

cmd.Parameters.AddWithValue("ids", Speciality.ID);

cmd.Parameters.AddWithValue("idf", FormTraning.ID);

cmd.Parameters.AddWithValue("idb", BasisTraning.ID);

cmd.Parameters.AddWithValue("num", CountPlaces);

cmd.ExecuteNonQuery();

conn.Close();

}

}

catch (System.Data.SqlClient.SqlException)

MessageBox.Show("Ошибка соединения", "Ошибка!");

}

public static void insert(Places dep)

{dep.insert();}

public static Places insert(int count, int spec, int form, int basi)

{

Places dep = new Places();

dep.CountPlaces = count;

dep.BasisTraning = BasisTraning.getById(basi);

dep.FormTraning = FormTraning.getById(form);

dep.Speciality = Speciality.getByIdA(spec);

dep.insert();

return dep;

}

public bool update()

{

DBConn dbc = new DBConn();

try

{ SqlConnectionStringBuilder sb = new SqlConnectionStringBuilder();

using (SqlConnection conn = new SqlConnection())

{ dbc.OpenConn(sb, conn);

string str = "UPDATE Places " + "SET NumberPlaces = @np " +

"WHERE IDFormTraning = @idf " +"AND IDSpeciality = @ids " +

"AND IDBasisTraning = @idb";

SqlCommand cmd = new SqlCommand(str, conn);

cmd.Parameters.AddWithValue("np", CountPlaces);

cmd.Parameters.AddWithValue("idf", FormTraning.ID);

cmd.Parameters.AddWithValue("idb", BasisTraning.ID);

cmd.Parameters.AddWithValue("ids", Speciality.ID);

cmd.ExecuteNonQuery();

conn.Close();

return true;

}

}

catch (System.Data.SqlClient.SqlException)

{

MessageBox.Show("Ошибка соединения", "Ошибка!");

return false;

}

}

}

Создание отчёта в Excel

public static bool exportToExcel(string pathFile, int speciality)

{try

{

Excel.Application exApp = new Excel.Application();

exApp.Visible = true; exApp.Workbooks.Add();

Excel.Worksheet worksheet = (Excel.Worksheet)exApp.ActiveSheet;

worksheet.Cells[3, 1] = "№";

worksheet.Cells[3, 2] = "Абитуриент, паспорт";

worksheet.Cells[3, 3] = "Специальность";

worksheet.Cells[3, 4] = "Форма обучения";

worksheet.Cells[3, 5] = "Основа обучения";

worksheet.Cells[3, 6] = "Согласие";

worksheet.Cells[3, 7] = "Оригинал";

int rowExcel = 4;

List<Statement> lstStatement = new List<Statement>();

if (speciality == 0) lstStatement = Statement.getAll();

else lstStatement = Statement.getBySpeciality(speciality);

Excel.Range range;

range = worksheet.Range["A3", System.Type.Missing];

range.EntireColumn.ColumnWidth = 6;

range = worksheet.Range["B3", System.Type.Missing];

range.EntireColumn.ColumnWidth = 45;

range = worksheet.Range["C3", System.Type.Missing];

range.EntireColumn.ColumnWidth = 17;

range = worksheet.Range["D3", System.Type.Missing];

range.EntireColumn.ColumnWidth = 18;

range = worksheet.Range["E3", System.Type.Missing];

range.EntireColumn.ColumnWidth = 18;

range = worksheet.Range["F3", System.Type.Missing];

range.EntireColumn.ColumnWidth = 12;

range = worksheet.Range["G3", System.Type.Missing];

range.EntireColumn.ColumnWidth = 12;

range = (Excel.Range)worksheet.Range["A3", "G3"].Cells;

range.Cells.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

range.Font.Bold = true;

range.Font.Size = 12;

for (int i = 0; i < lstStatement.Count; i++)

{worksheet.Cells[rowExcel, 1] = lstStatement[i].ID;

worksheet.Cells[rowExcel, 2] = lstStatement[i].Entrant.FIOToString() + ", " + lstStatement[i].Entrant.Pasport;

worksheet.Cells[rowExcel, 3] = lstStatement[i].Speciality.Title;

worksheet.Cells[rowExcel, 4] = lstStatement[i].FormTraning.Title;

worksheet.Cells[rowExcel, 5] = lstStatement[i].BasisTraning.Title;

worksheet.Cells[rowExcel, 6] = (lstStatement[i].Consent) ? "Есть" : "Отсутствует";

worksheet.Cells[rowExcel, 7] = (lstStatement[i].OriginalDoc) ? "Есть" : "Отсутствует";

rowExcel++;

}

int lstCount = lstStatement.Count + 3;

string newR = "G" + lstCount.ToString();

range = (Excel.Range)worksheet.Range["A3", newR].Cells;

range.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

range.Font.Size = 10;

range.Cells.Borders.LineStyle = Excel.XlLineStyle.xlContinuous;

range = (Excel.Range)worksheet.Range["A1", "G1"].Cells;

range.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

range.Font.Size = 14;

range.Font.Bold = true;

range.Merge(Type.Missing);

worksheet.Cells[1, 1] = "Отчёт о заявлениях по специальностям";

range = (Excel.Range)worksheet.Range["A2", "C2"].Cells;

range.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

range.Font.Size = 12;

range.Font.Bold = true;

range.Merge(Type.Missing);

worksheet.Cells[2, 1] = "БД 'Отборочная комиссия', Братковой И.О., ПВ-31";

range = (Excel.Range)worksheet.Range["D2", "E2"].Cells;

range.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

range.Font.Size = 12;

range.Font.Bold = true;

range.Merge(Type.Missing);

worksheet.Cells[2, 4] = "Количество заявлений: " + (lstCount - 3).ToString();

range = (Excel.Range)worksheet.Range["F2", "G2"].Cells;

range.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

range.Font.Size = 12;

range.Font.Bold = true;

range.Merge(Type.Missing);

string s = DateTime.Today.ToShortDateString();

worksheet.Cells[2, 6] = "Дата: " + s;

string pathToXmlFile = pathFile;

worksheet.SaveAs(pathToXmlFile);

exApp.Quit();

return true;

}

catch

{ MessageBox.Show("Произошла ошибка при формировании отчёта!");

return false;

}

}

## 

## Приложение С – XML-файл

<?xml version="1.0"?>

[<BratckovaPV31Comission>](file:///C:\Users\HYSTOU\Desktop\3kmmr\bd\tmp.xml)

<Entrant NowCount="2" AllCount="1" CurrDate="03.01.1989 0:00:00" Hostel="False" PreviousEducation="Белгород" BirthDate="03.01.1989 0:00:00" Adress="г. Белгород" Passport="3453 453453" Patronymic="Николаевич" IName="Виктор" Surname="Васиьев" Contact="7(887) 878-9787">

<Statement TitleBasisTraning="В рамках КЦП" TitleFormTraning="Очная" TitleSpeciality="Провизор" OriginalDocument="True" Consent="True" Count="2"/>

<Statement TitleBasisTraning="С полным возмещением затрат" TitleFormTraning="Очно-заочная" TitleSpeciality="ПРосто специальность" OriginalDocument="False" Consent="False" Count="2"/><Point Discipline="Математика" PointBall="100"/>

<Point Discipline="Химия" PointBall="99"/>

<EntrantAchiv Count="1" Title="ГТО"/>

</Entrant>

[<Entrant NowCount="2" AllCount="1" CurrDate="06.11.1996 0:00:00" Hostel="True" PreviousEducation="Россия" BirthDate="06.11.1996 0:00:00" Adress="Белгород" Passport="4507 002154" Patronymic="Олеговна" IName="Ирина" Surname="Браткова" Contact="7(890) 604-0634">](file:///C:\Users\HYSTOU\Desktop\3kmmr\bd\tmp.xml)

<Statement TitleBasisTraning="В рамках КЦП" TitleFormTraning="Очная" TitleSpeciality="Провизор" OriginalDocument="True" Consent="True" Count="2"/>

<Statement TitleBasisTraning="В рамках КЦП" TitleFormTraning="Очная" TitleSpeciality="ПВ" OriginalDocument="False" Consent="False" Count="2"/>

<Point Discipline="Русский язык" PointBall="95"/>

<Point Discipline="Математика" PointBall="60"/>

<Point Discipline="Информатика" PointBall="78"/>

<Point Discipline="Химия" PointBall="55"/>

<EntrantAchiv Count="3" Title="Диплом"/>

<EntrantAchiv Count="3" Title="Городская олимпиада"/>

<EntrantAchiv Count="3" Title="Всероссийская олимпиада"/>

</Entrant>

</BratckovaPV31Comission>