1. МОДЕЛИРОВАНИЕ БАЗЫ ДАННЫХ

1.1. Разработка технического задаия

Информационная система электронного учета учащихся представляет собой совокупность аппаратного и программного обеспечения, предназначенного для выполнения следующих задач:

* учет студентов и групп;
* учет преподавателей и кафедр;
* ведение учета пройденных уроков;
* ведение учета посещаемости на уроках;
* ведение учета экзаменов (зачетов);
* ведение учета оценок на экзаменах (зачетах).

С точки зрения программной архитектуры, информационная система представляет собой распределенное клиент-серверное приложение [1]. Оно включает в себя базу данных, расположенную на отдельном сервере и клиентские компьютеры (терминалы), которые используются сотрудниками учебного заведения для ввода данных и получения отчетов.

С системой могут работать следующие группы лиц:

* преподаватели;
* сотрудники кафедр;
* студенты.

Сотрудники кафедры будут обеспечивать следующие функции:

* ввод данных о новом преподавателе и кафедре;
* ввод данных о студентах и группах;
* ввод данных об изучаемых предметах и расписании занятий.

Преподаватели будут обеспечивать следующие функции:

* учет посещаемости занятий;
* учет результатов прохождения экзаменов и зачетов.

Студенты, преподаватели и сотрудники кафедры смогут получать следующие отчеты:

* отчет о проведенных и планируемых занятиях;
* отчет о проведенных и планируемых экзаменах;
* отчет о посещаемости занятий;
* отчет о результатах экзаменов.

1.2. Выделение объектов предметной области, их характеристик и построение ER-модели

Так как перенос технического задания непосредственно на реляционную модель данных довольно сложен, разработка базы данных ведется в несколько этапов с использованием семантических моделей [2]:

* формирование концептуальной схемы данных на основе выбранной семантической модели;
* преобразование концептуальной схемы данных к логической схеме;
* преобразование логической схемы данных к физической схеме.

Модель сущность-связь (ER) является одной из наиболее популярных семантических моделей [3]. Модель может быть реализована как в текстовом, так и в графическом виде (диаграмма сущность-связь, ER-диаграмма).

Этапы проектирование концептуальной схемы данных с использованием ER -модели включают:

* выделение сущностей;
* определение связей;
* выделение атрибутов;
* ограничения целостности.

Концептуальная схема данных для БД журнала учащихся показана на рисунке 1. Разработка схемы данных проводилась в программе SAP PowerDesigner. Выбор данной программы обусловлен тем, что она обеспечивает сквозную разработку БД начиная от концептуальной схемы и заканчивая скриптом на языке SQL для генерации таблиц базы данных.

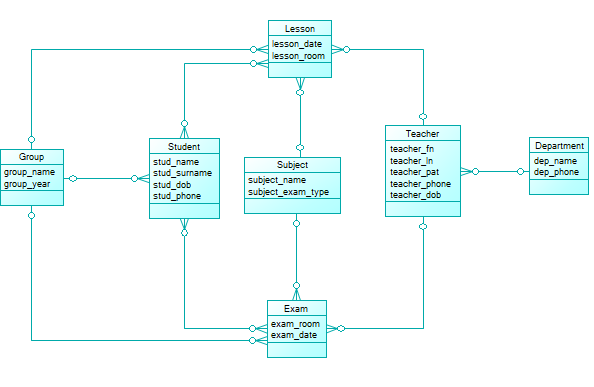


Рисунок 1 ‑ Концептуальная схема базы данных

Как следует из рисунка, в концептуальной схеме присутствуют следующие сущности:

* Group – описывает группы студентов.
* Student – описывает учащихся.
* Department – описывает кафедры учебного заведения.
* Teacher – описывает преподавателей.
* Subject – описывает изучаемые предметы.
* Lesson – описывает уроки.
* Exam – описывает экзамены.

В схеме между сущностями используются связи один-ко-многим и многие-ко-многим (для связи множества студентов с множеством уроков и экзаменов)

В таблицах 1-7 представлены описания атрибутов перечисленных сущностей.

Таблица 1 – атрибуты сущности Group.

|  |  |
| --- | --- |
| Название атрибута | Описание атрибута |
| group\_name | Название группы |
| group\_year | Год начала обучения группы |

Таблица 2 – атрибуты сущности Student.

|  |  |
| --- | --- |
| Название атрибута | Описание атрибута |
| stud\_name | Имя студента |
| stud\_surname | Фамилия студента |
| stud\_dob | Дата рождения |
| stud\_phone | Телефон |

Таблица 3 – атрибуты сущности Department.

|  |  |
| --- | --- |
| Название атрибута | Описание атрибута |
| dep\_name | Название кафедры |
| dep\_phone | Телефон кафедры |

Таблица 4 – атрибуты сущности Teacher.

|  |  |
| --- | --- |
| Название атрибута | Описание атрибута |
| teacher\_fn | Имя преподавателя |
| teacher\_ln | Фамилия преподавателя |
| teacher\_pat | Отчество преподавателя |
| teacher\_phone | Телефон |
| teacher\_dob | День рождения |

Таблица 5 – атрибуты сущности Subject.

|  |  |
| --- | --- |
| Название атрибута | Описание атрибута |
| subject\_name | Название предмета |
| subject\_exam\_typet | Тип итогового контроля (экзамен/зачет) |

Таблица 6 – атрибуты сущности Lesson.

|  |  |
| --- | --- |
| Название атрибута | Описание атрибута |
| lesson\_date | Дата проведения занятия |
| lesson\_room | Аудитория |

Таблица 7 – атрибуты сущности Exam.

|  |  |
| --- | --- |
| Название атрибута | Описание атрибута |
| exam\_date | Дата проведения экзамена |
| exam\_room | Аудитория |

Студент может состоять в одной группе, поэтому сущности Group и Student соединены связью «один-ко-многим». В то же время один студент может множество занятий и на одном занятии таких студентов может быть несколько. Поэтому сущности Student и Lesson соединены связью «многие-ко-многим». На основании подобных рассуждений составлены связи для остальных сущностей модели.

1.3. Выбор модели базы данных и построение логической модели

Рассмотрим правила перехода от концептуальной схемы к логической [4]:

* каждый простой тип сущности отображается в отношение;
* атрибуты ER-модели отображаются в атрибуты отношений;
* компоненты уникального идентификатора отображаются в первичный ключ;
* связь “один ко многим” и “один к одному” отображаются в виде внешних ключей;
* связи “многие ко многим” отображаются с созданием дополнительного отношения;
* сложные связи (более, чем между двумя типами сущностей) декомпозируются с выделением нового типа сущности;
* многозначные атрибуты декомпозируются с выделением нового типа сущности.

Логическая схема данных для БД журнала учета учащихся представлена на рисунке 2.

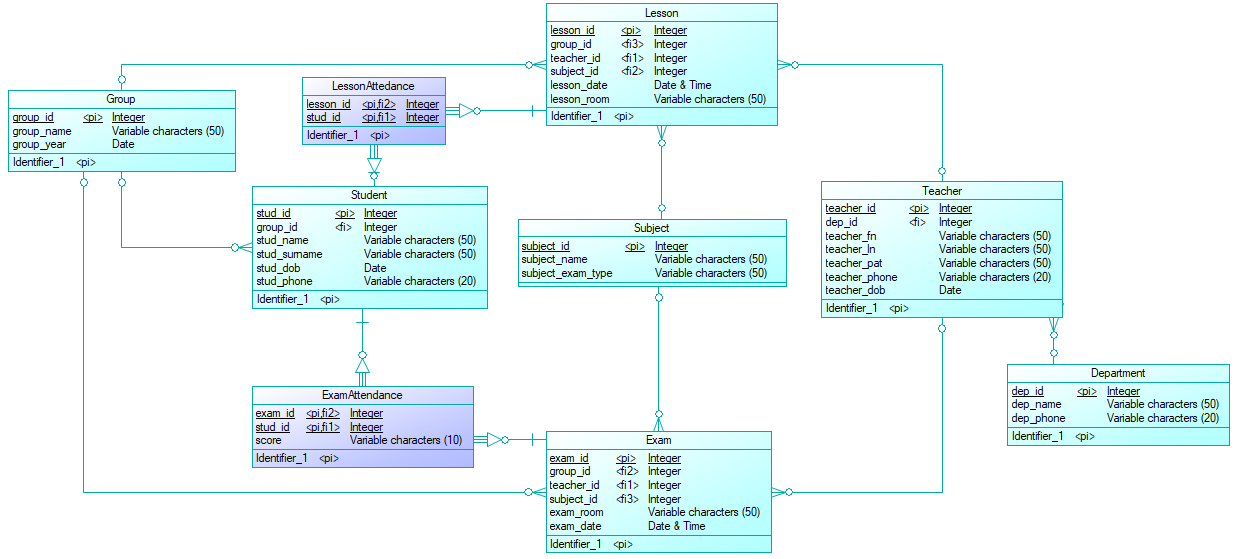


Рисунок 2 ‑ Логическая схема данных

Реляционная модель данных не может корректно описать отношение «многие-ко-многим», поэтому для разрешения двух этих связей в концептуальной базе данных создано отношение LessonAttendance (посещаемость занятий) и ExamAttendance (посещаемость экзаменов), которые содержат атрибуты, показанные в таблицах 8 и 9.

Таблица 8 – атрибуты отношения LessonAttendance.

|  |  |
| --- | --- |
| Название атрибута | Описание атрибута |
| lesson\_id | Идентификатор занятия |
| stud\_id | Идентификатор студента |

Таблица 9 – атрибуты отношения ExamAttendance.

|  |  |
| --- | --- |
| Название атрибута | Описание атрибута |
| exam\_id | Идентификатор экзамена |
| stud\_id | Имя студента |
| score | Оценка студента |

1.4. Выбор программной среды для реализации проекта базы данных и проектирование физической модели

При переносе логической модели в физическую (для конкретной СУБД) выполняются следующие действия [4]:

* определение типов данных для атрибутов, соответствующих конкретной СУБД;
* определение допустимости неопределенных значений;
* выбор стратегии обработки исключительных ситуаций при попытках нарушения ссылочной целостности;
* уточнение названий таблиц, атрибутов;
* реализация ограничений целостности, процедурная обработка;
* формирование описания модели данных в рамках выбранной СУБД;
* формирование индексов;
* секционирование и партицирование.

В среде PowerDesigner имеется возможность автоматической генерации логической модели в физическую [5] (рисунок 3).

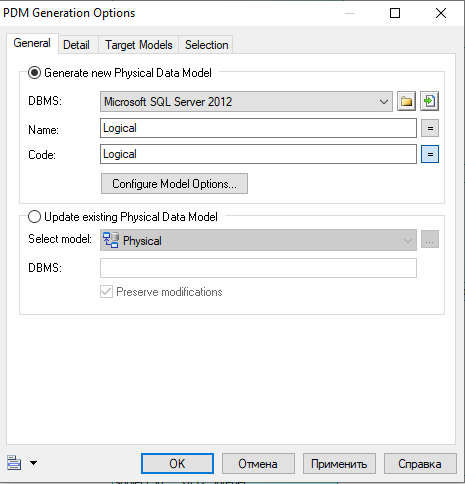


Рисунок 3 ‑ Генерация физической модели

Как видно из рисунка, в качестве целевой СУБД выбран Microsoft SQL Server 2012. Данная СУБД имеет высокую степень совместимости с операционной системой Microsoft Windows, имеет встроенные инструменты для работы с Microsoft Visual Studio, в которой будет разрабатываться интерфейсная часть БД, а также доступная для свободного скачивания и использования в учебных целях.

Физическая модель базы данных показана на рисунке 4.

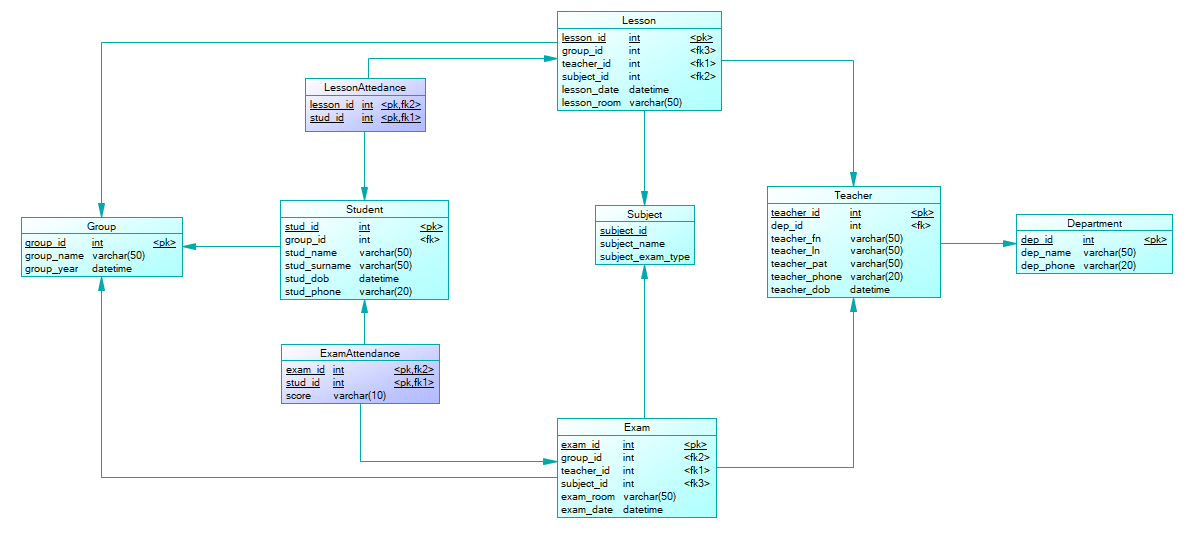


Рисунок 4 ‑ Физическая модель данных

Как видно из рисунка, типы данных атрибутов в физической модели адаптированы под типы данных, поддерживаемых СУБД Microsoft SQL Server.

2. РЕАЛИЗАЦИЯ ПРОЕКТА БАЗЫ ДАННЫХ

2.1. Описание создания таблиц базы данных

Для создания таблиц базы данных на основании разработанной ранее физической модели, использовался SQL-скрипт, который позволяет сгенерировать программа PowerDesigner. На рисунке 5 показано окно генерации скрипта.

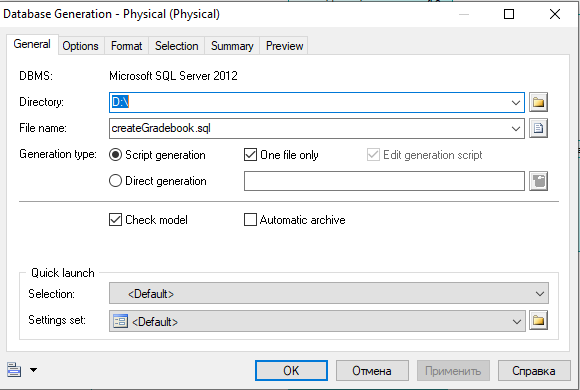


Рисунок 5 ‑ Генерация SQL скрипта

Как видно из рисунка, в качестве целевой СУБД выбрана Microsoft SQL Server 2012.

Сгенерированный скрипт (рисунок 6) затем был открыт и выполнен в программе SQL Server Management Studio, в предварительно созданной пустой базе данных Gradebook.

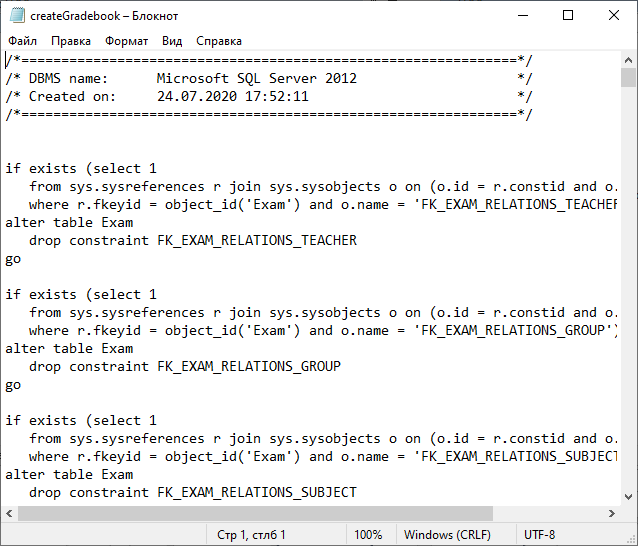


Рисунок 6 ‑ Фрагмент скрипта генерации таблиц базы данных

После выполнения скрипта, в базе данных были созданы все необходимые таблицы. Для их наглядного представления была создана диаграмма базы данных (рисунок 7)

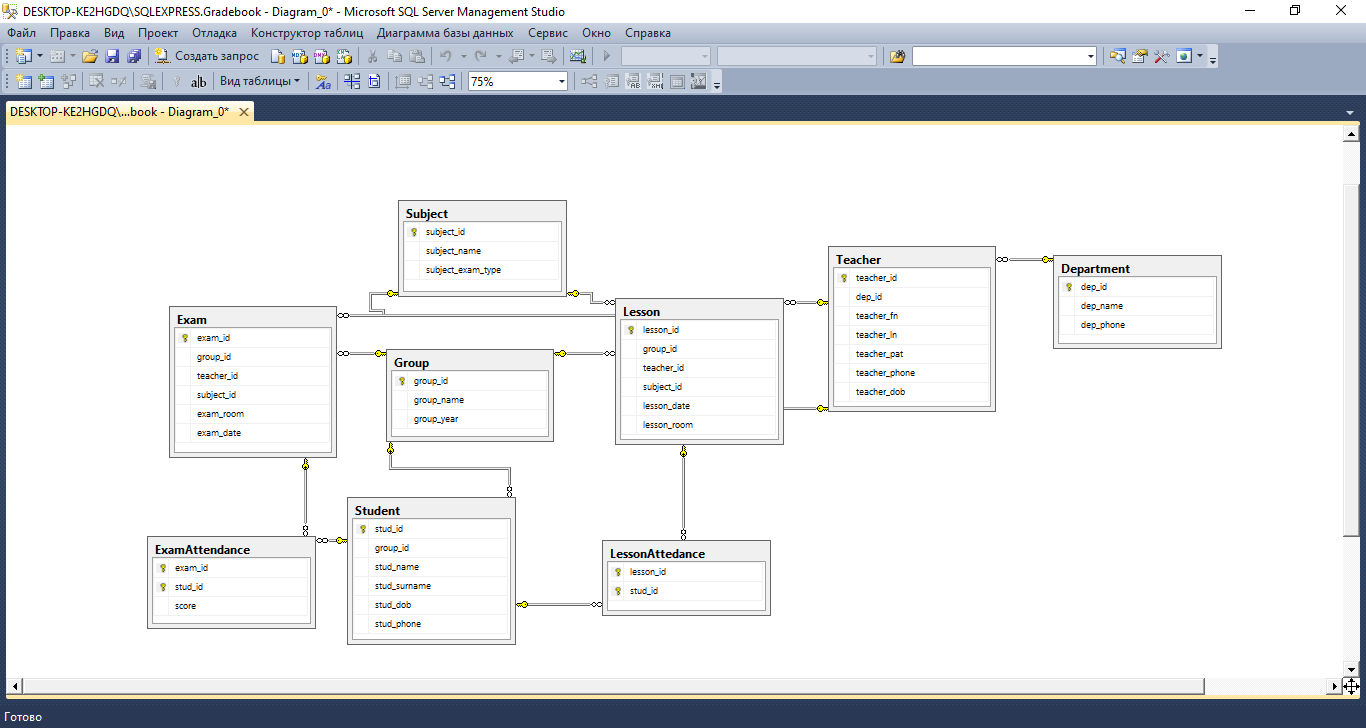


Рисунок 7 ‑ Диаграмма базы данных

Как можно убедиться из рисунка 7, созданная база данных соответствует показанной в предыдущей главе физической модели.

Далее таблицы базы данных были заполнены тестовыми данными, как показано на рисунках 8 – 16.

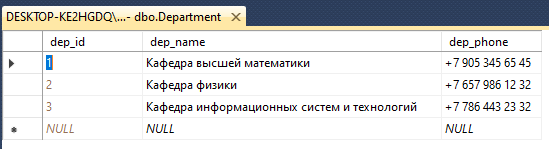


Рисунок 8 ‑ Тестовые данные таблицы Department

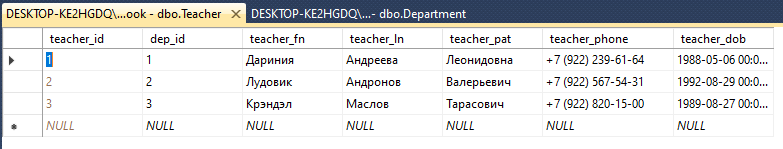


Рисунок 9 ‑ Тестовые данные таблицы Teacher

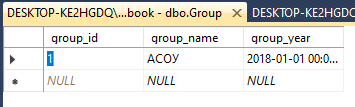


Рисунок 7 ‑ Тестовые данные таблицы Group

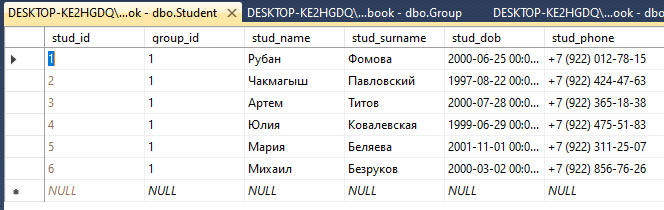


Рисунок 11 ‑ Тестовые данные таблицы Student

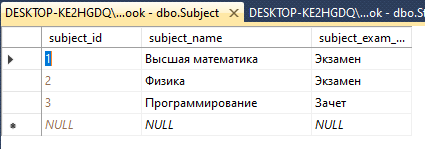


Рисунок 8 ‑ Тестовые данные таблицы Subject

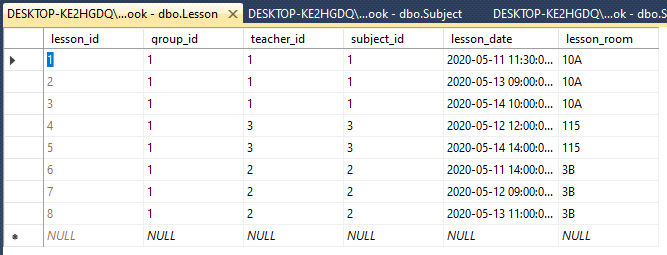


Рисунок 9 ‑ Тестовые данные таблицы Lesson

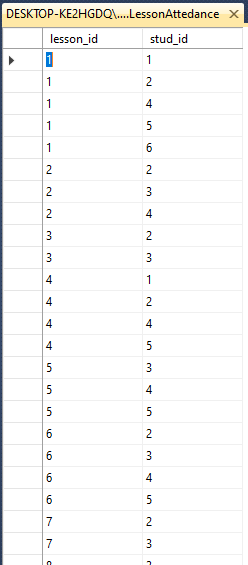


Рисунок 10 ‑ Тестовые данные таблицы LessonAttendance

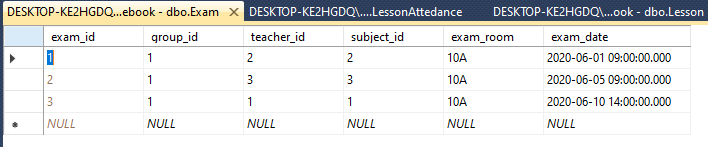


Рисунок 11 ‑ Тестовые данные таблицы Exam

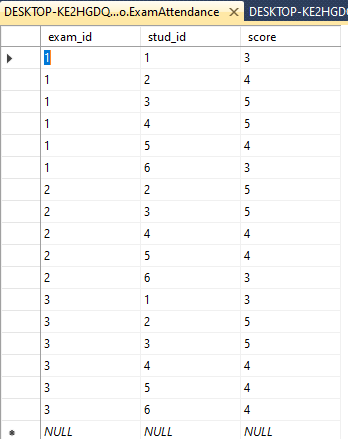


Рисунок 12 ‑ Тестовые данные таблицы ExamAttendance

2.2. Решение задач пользователей с помощью построения представлений

Представления представляют собой сохраненные запросы к базе данных, содержащие выборки из нескольких связанных таблиц. Представление формирует на выходе таблицу, которая формируется динамически и содержит указанные поля базы данных. С помощью представлений удобно формировать разнообразные отчеты.

В данной работе создадим представления, отображающие списки уроков, экзаменов, посещаемости и результатов экзаменов.

Среда Microsoft SQL Server Management Studio позволяет создавать представления как в графическом режиме, так и путем реализации SQL-запроса. Представление, возвращающее список уроков (Get\_lessons), показано на рисунке 17.

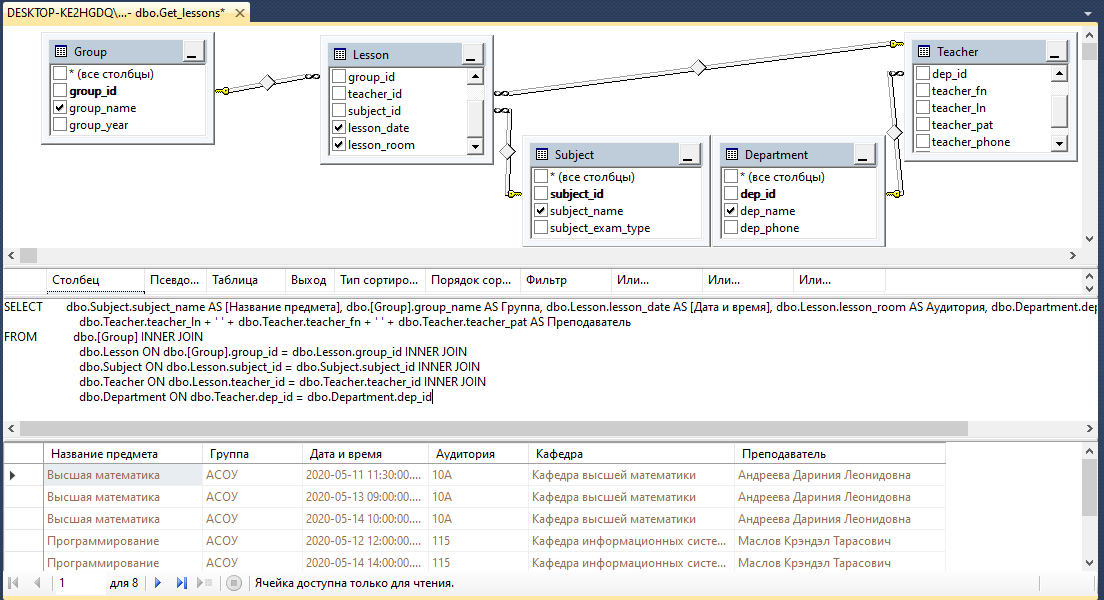


Рисунок 13 ‑ Проект представления Get\_lessons

На рисунке 18 показано представления, возвращающее отчет об экзаменах (Get\_exams).

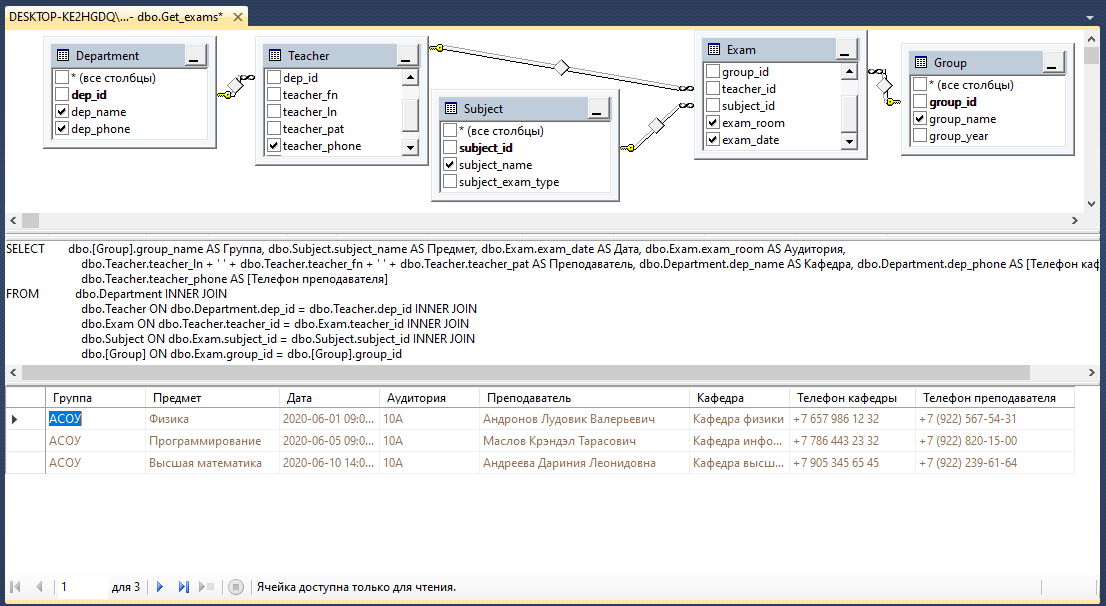


Рисунок 14 ‑ Графическое отображение представления Get\_exams

На рисунке 19 показано представления, возвращающее отчет о посещении занятий (Get\_lesson\_attendance).

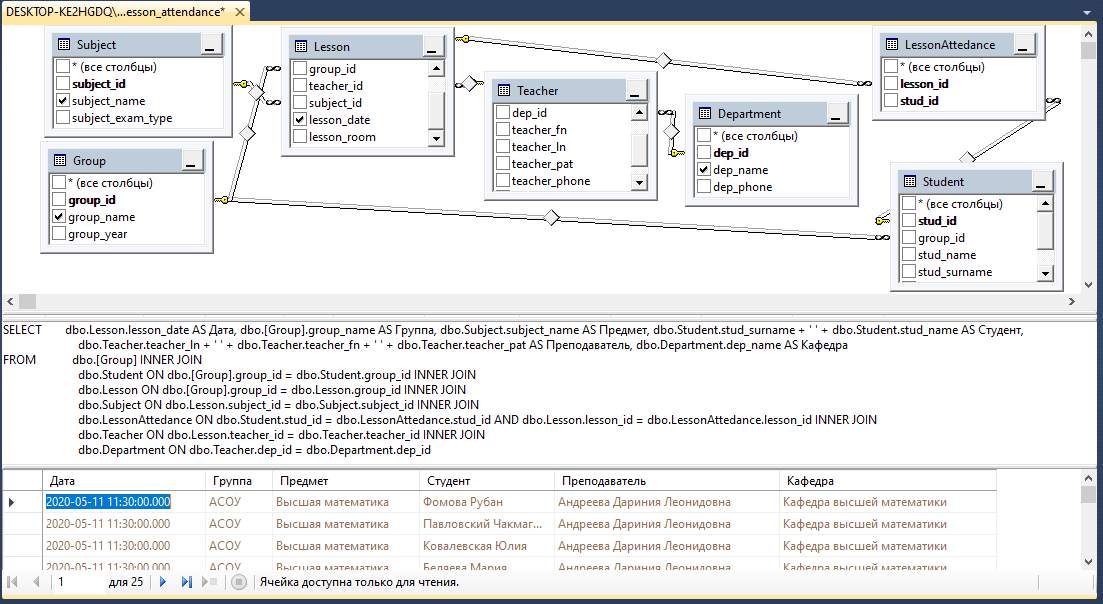


Рисунок 15 ‑ Графическое отображение представления Get\_lesson\_attendance

На рисунке 20 показано представления, возвращающее отчет о посещении занятий (Get\_exam\_attendance).

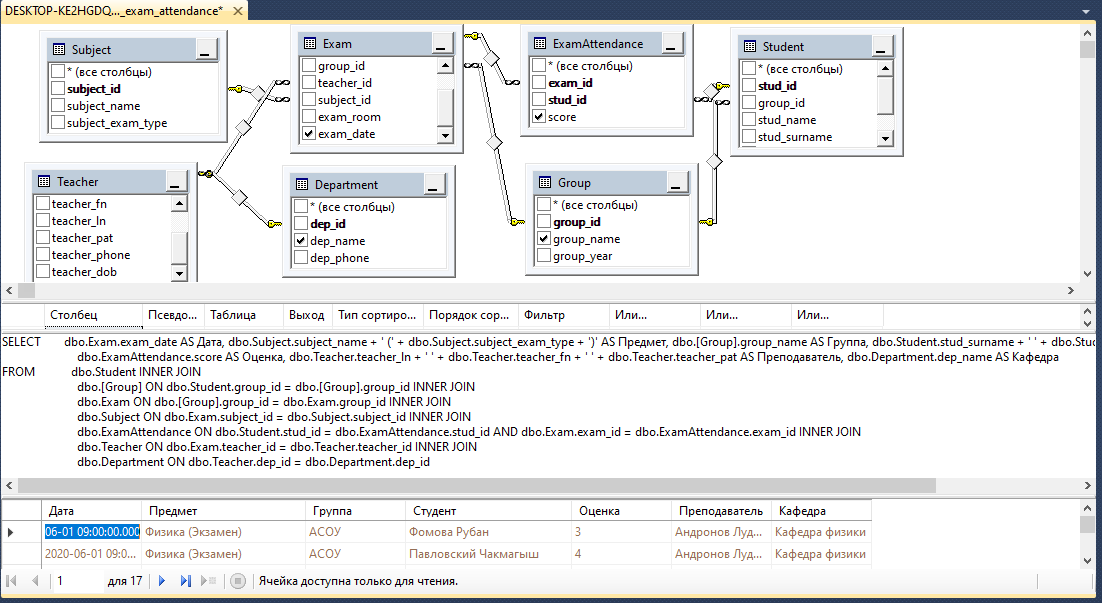


Рисунок 20 ‑ Графическое отображение представления Get\_exam\_attendance

2.3. Визуализация базы данных

Для разработки средств визуализации данных разработанной БД использовалась среда Microsoft Visual Studio 2019 и язык программирования C#. Был разработан проект Gradebook, который построен на шаблоне Windows Forms Application (рисунок 21).

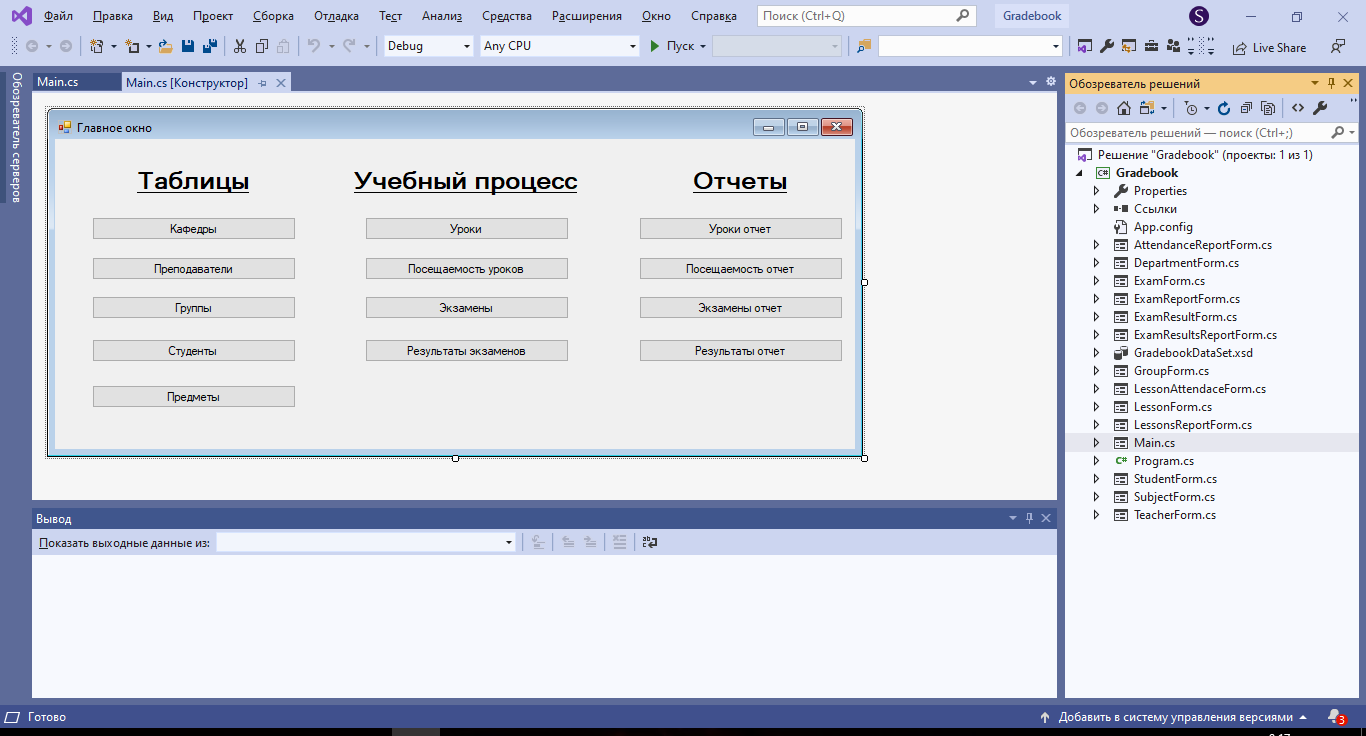


Рисунок 21 ‑ Проект Gradebook в среде Visual Studio

Данные проект содержит в себе следующие окна:

* Главное окно (Main.cs)
* Окно работы с таблицей кафедр (DepartmentForm.cs)
* Окно работы с таблицей преподавателей (TeacherForm.cs)
* Окно работы с таблицей групп (GroupForm.cs)
* Окно работы с таблицей студентов (StudentForm.cs)
* Окно работы с таблицей предметов (SubjectForm.cs)
* Окно работы с таблицей занятий (LessonForm.cs)
* Окно работы с таблицей посещаемости (LessonAttendanceForm.cs)
* Окно работы с таблицей экзаменов (ExamForm.cs)
* Окно работы с таблицей результатов экзаменов (ExamResultForm.cs)
* Окно отчета о занятиях (LessonsReportForm.cs)
* Окно отчета о посещаемости занятий (AttendanceReportForm.cs)
* Окно отчета об экзаменах (ExamReportForm.cs)
* Окно отчета о результатах экзаменов (ExamsResultReportForm.cs)

На рисунке 22 показана главная форма приложения

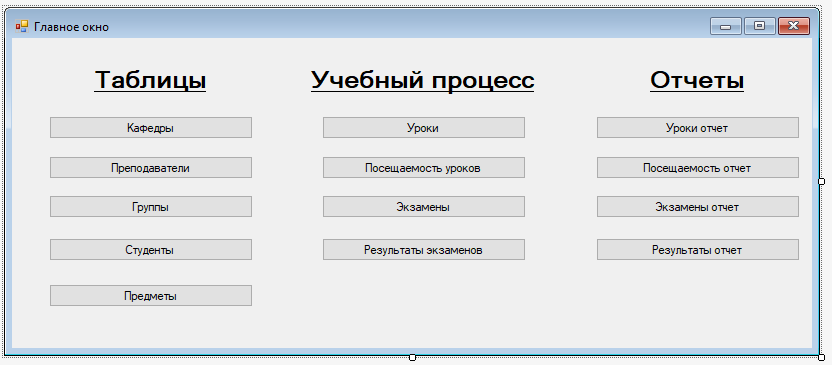


Рисунок 22 ‑ Главное окно программы

Из данного окна вызываются все остальные формы приложения. Его закрытие приведет к закрытию программы. Для открытия окон используется код, аналогичный приведенному ниже:

private void button13\_Click(object sender, EventArgs e)

{

Form examResultsReport = new ExamResultsReportForm();

examResultsReport.Show();

}

Следующая группа окон работает напрямую с таблицами базы данных. Рассмотрим их на примере окна SubjectForm.cs (рисунок 23).

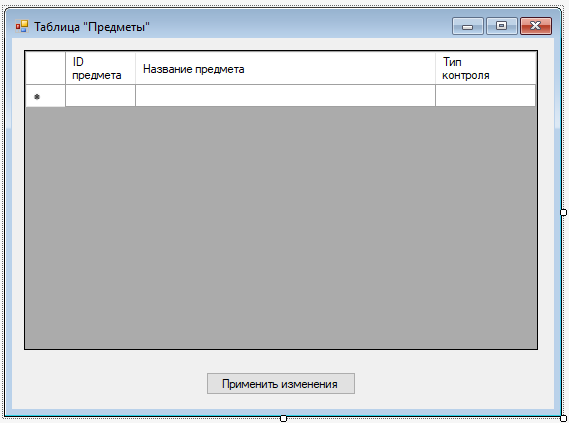


Рисунок 16 ‑ Окно CreditForm.cs

В данном окне расположен элемент DataGridView, который представляет собой таблицу, связанную с заданной таблицей (отношением) базы данных (в контексте данного окна с таблицей Credit). При инициализации окна в таблице отображаются записи из связанного отношения базы данных.

Пользователь имеет возможность добавлять, изменять и удалять данные в DataGridView. При нажатии кнопки «Применить изменения» изменения записываются в базу данных и DataGridView автоматически обновляется. Для этого используется следующий код:

this.subjectTableAdapter.Update(this.gradebookDataSet.Subject);

this.subjectTableAdapter.Fill(this.gradebookDataSet.Subject).

Для полей, которые содержат внешние ключи, используется замена текстового поля DataGridView на ComboBox, который позволяет вместо индекса внешнего ключа отображать осмысленное значение из связанной таблицы.

Таким образом, использование DataGridView с привязкой к конкретной таблице базы данных позволяет значительно упростить программирование интерфейсной части БД.

Для окон отчета также используется DataGridView, только в этом случае он привязывается не к таблицам базы данных, а к разработанным ранее представлениям (рисунок 24).

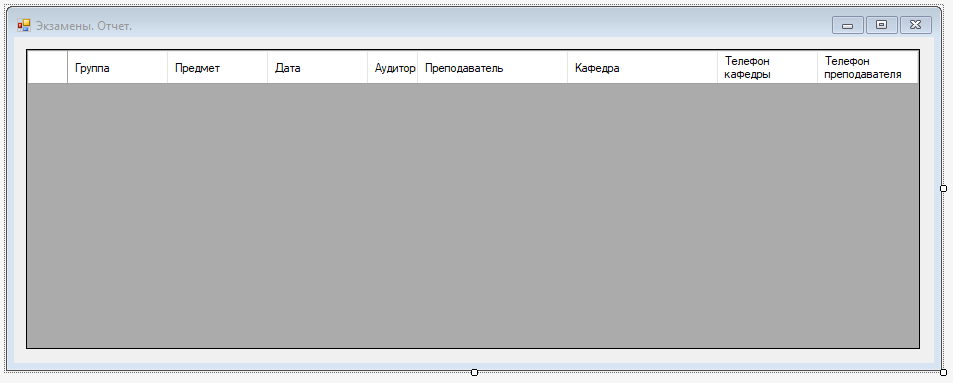


Рисунок 17 ‑ Окно отчета об экзаменах

В отчетах запрещено редактировать, удалять или добавлять новые записи.

2.4. Руководство пользователя

Программа предназначена для работы в операционных системах Windows, начиная с Windows 7. Необходимо убедиться в наличии СУБД Microsoft SQL Server 2012 или более поздней версии, а также наличии фреймворка .NET версии не ниже 4.8. Рассмотрим работу с программой на примере тестовых данных.

После запуска программы появляется главное окно (рисунок 25).

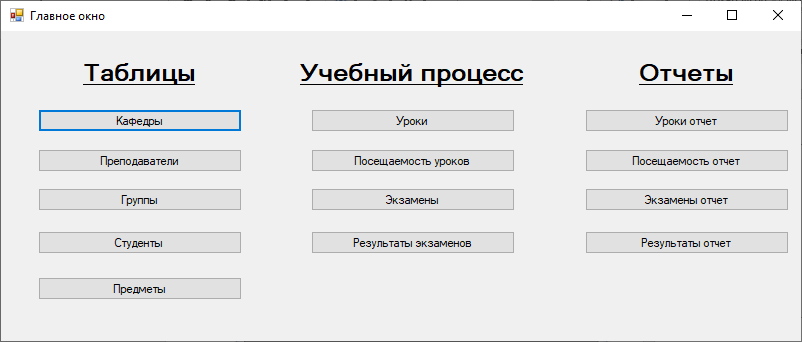


Рисунок 18 ‑ Главное окно программы

При нажатии кнопки закрытия главного окна, программа останавливает свою работу. По нажатию кнопки «Кафедры», открывается окно редактирования таблицы Department базы данных (рисунок 26)

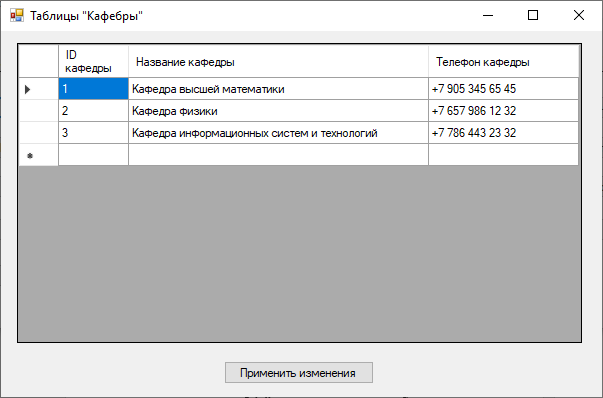


Рисунок 19 ‑ Работа с таблицей кафедр

Для добавления новой записи необходимо дважды кликнуть по пустому полю таблицы, ввести данные и нажать кнопку «Применить изменения».

Для изменения данных необходимо дважды кликнуть по нужному полю таблицы, внести изменения и затем нажать «Применить изменения».

Для удаления данных необходимо выбрать нужную строку таблицы, нажать Delete на клавиатуре и затем нажать ««Применить изменения».

Для закрытия данной формы и переходу к главному окну необходимо закрыть окно.

Все окна программы имеют единообразную структуру и одинаковые элементы на разных окнах ведут себя одинаково, следовательно описанное выше справедливо и для остальных окон.

Далее, на рисунках 27-30 показаны окна для работы с таблицами Teacher, Group, Student, Subject, базы данных соответственно.

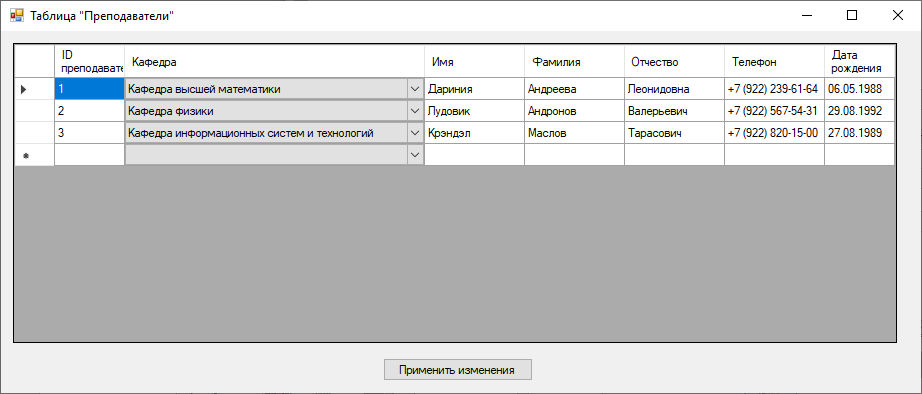


Рисунок 20 ‑ Работа с таблицей преподавателей

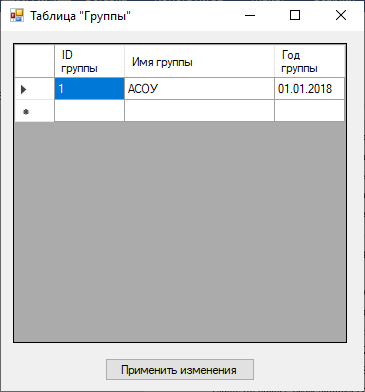


Рисунок 21 ‑ Работа с таблицей групп

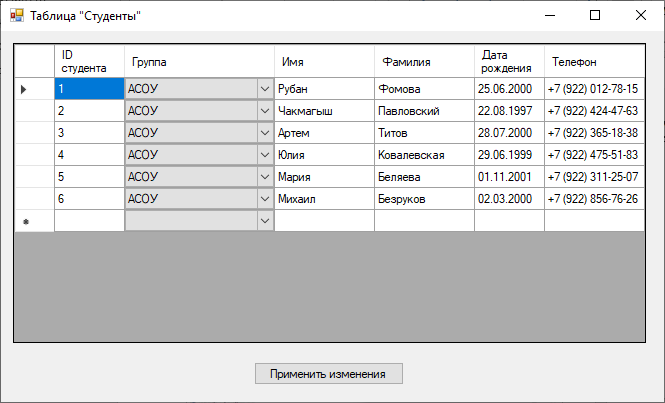


Рисунок 22 ‑ Работа с таблицей студентов

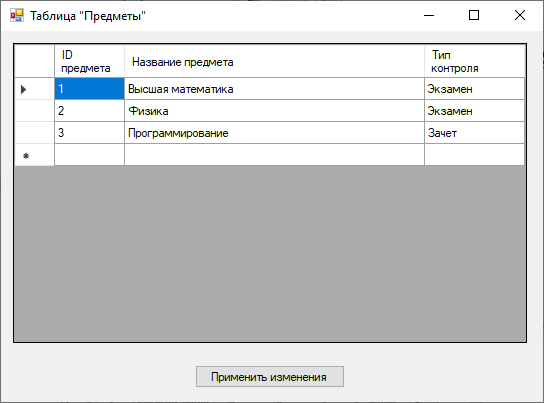


Рисунок 30 ‑ Работа с таблицей предметов

Работа с таблицами, выделенная в группу «учебный процесс» осуществляет схожие манипуляции с таблицами БД: Lesson, LessonAttendance, Exam, ExamAttendance (рисунки 31-34)

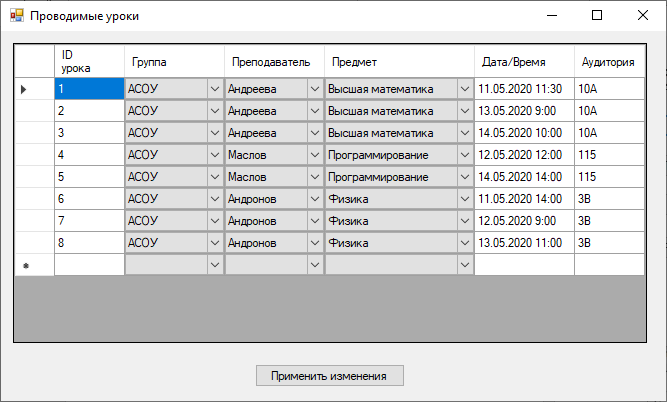


Рисунок 31 ‑ Работа с таблицей занятий

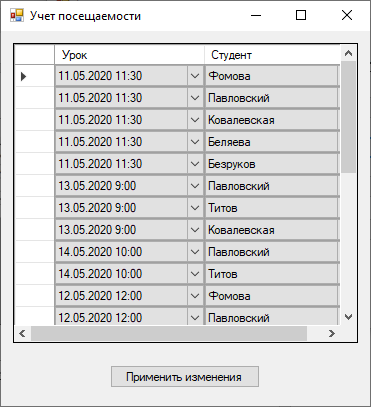


Рисунок 32 ‑ Работа с таблицей посещаемости

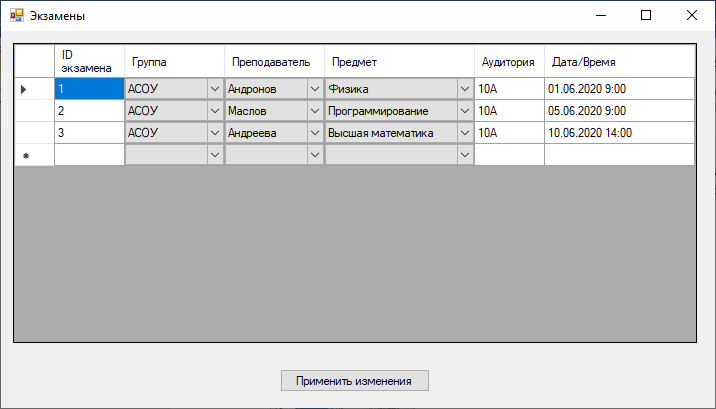


Рисунок 30 ‑ Работа с таблицей экзаменов

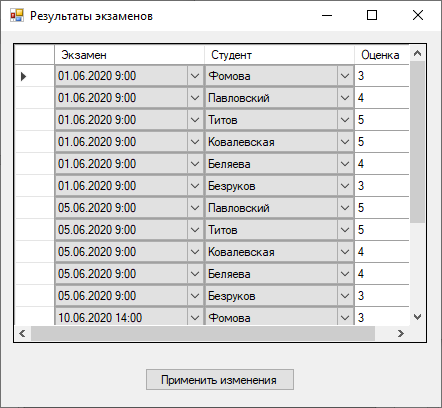


Рисунок 31 ‑ Работа с таблицей результатов экзаменов

Из рисунков видно, что если количество информации превышает размер таблицы в вертикальном или горизонтальном направлении, то появляются соответствующие полосы прокрутки. Кроме этого, при наведении на любое из полей таблицы появляется всплывающая подсказка со значением этого поля.

Таблица позволяет сортировать и группировать записи, нажимая на заголовки столбцов.

Нажатие кнопки «Уроки отчет» появляется окно, показывающее информацию о занятиях. (рисунок 32).

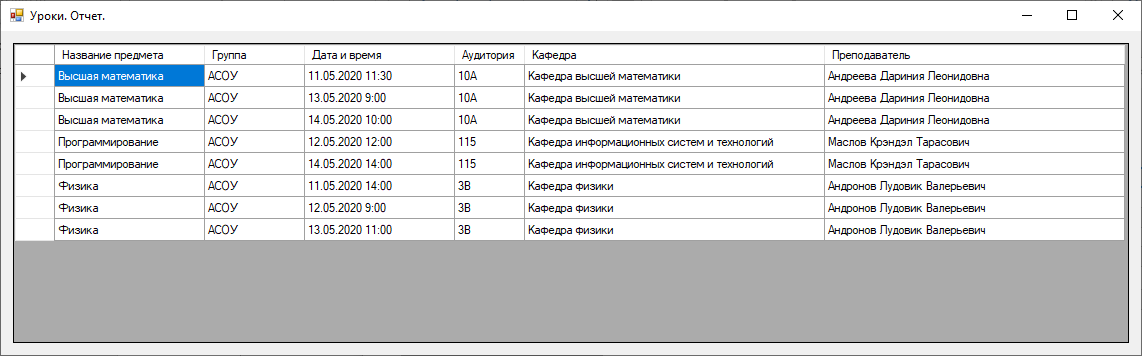


Рисунок 32 ‑ Отчет о занятиях

Отчеты представляют информацию в более удобной для восприятия форме, чем формы для редактирования таблиц. К примеру, в отчетах может конкатенироваться информация из нескольких разных полей одной таблицы или связанных таблиц. Так, на рисунке 32, поля столбца «Преподаватель» содержат объединенные значения трех различных полей таблицы базы данных.

При нажатии на кнопку «Посещаемость отчет» появляется информация о посещаемости студентами занятий (рисунок 33).



Рисунок 33 ‑ Отчет о посещаемости

По нажатию на кнопку «Экзамены отчет» появляется окно, отображающее отчет об экзаменах (рисунок 34)

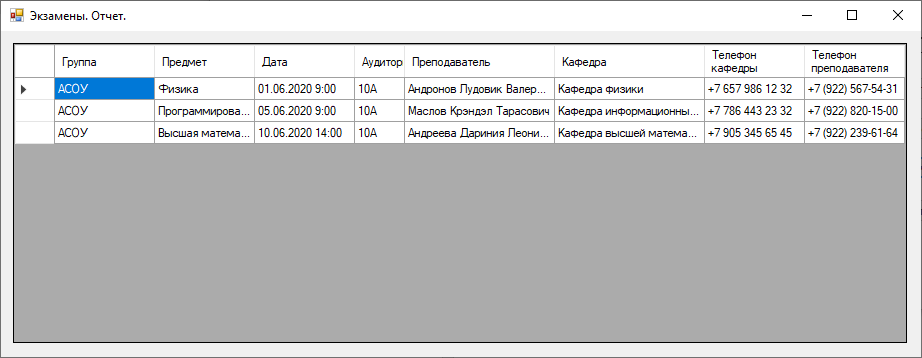


Рисунок 23 – Отчет об экзаменах

При нажатии кнопки «Результаты отчет» открывается отчет о результатах прохождения студентов экзаменов (рисунок 35).

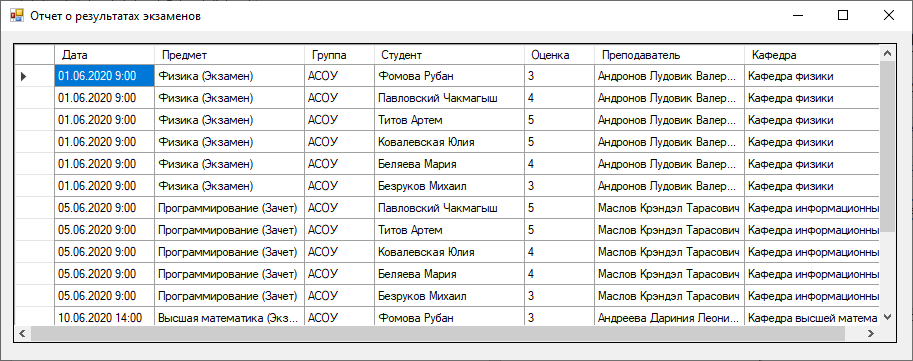


Рисунок 24 – Отчет о результатах экзаменов

Таким образом, разработанная программа позволяет эффективно вести работу с базой данных электронного журнала учащихся.