МИНОБРНАУКИ РОССИИ

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

**КУРСОВАЯ РАБОТА**

**по дисциплине «Управление данными»**

Тема: Разработка и реализация базы данных

|  |  |  |
| --- | --- | --- |
| Студент гр. 8363 |  |  |
| Преподаватель |  | Татарникова Т.М. |

Санкт-Петербург

2020

ЗАДАНИЕ

**НА КУРСОВУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| Студент | | |
| Группа 8363 | | |
| Тема работы: Разработка и реализация базы данных | | |
| Исходные данные:  Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для завуча школы.  **Для каждого учителя в БД должны храниться:** сведения о предметах, которые он преподает, номере кабинета, который за ним закреплен, номера классов, в которых он ведет занятия, номере урока и дне, когда он ведет уроки. Существуют учителя, которые не имеют собственного кабинета.  **Об учениках должны храниться следующие сведения:** фамилия и имя, в каком классе учится, какая оценка по каждому предмету получена. Ученик может исправить полученную оценку.  **Завуч может** добавить информацию о новом учителе или ученике, а также удалить - выбывших.  **Завучу могут потребоваться следующие сведения:**   * какой предмет будет в заданном классе, например, во вторник на заданном уроке; * кто из учителей преподает в заданном классе; * в каком кабинете будет 5-й урок в среду у некоторого класса; * в каких классах преподает учитель заданный предмет; * расписание на заданный день недели для класса.   **Завуч может вносить следующие изменения:**   * вносить информацию о новом учителе; * удалять запись об ученике; * изменить оценку ученику.   Необходимо предусмотреть возможность выдачи справки о количестве учеников в данном классе и отчета о работе школы (количество учителей по предметам, количество кабинетов, число учеников в каждом классе, число троечников, хорошистов и отличников). | | |
| Содержание пояснительной записки:  «Задание на курсовую работу», «Содержание», «Введение», «Анализ предметной области», «Обоснование модели данных», «Проектирование БД», «Обоснование выбора СУБД», «Описание функций управления данными», «Организация защиты БД», «Заключение», «Список использованных источников», «Приложение А. Руководство пользователя БД», «Приложение Б. Листинг программного кода SQL», «Приложение В. Листинг программного кода C#». | | |
| Предполагаемый объем пояснительной записки:  Не менее 40 страниц. | | |
| Дата выдачи задания: 14.09.2020 | | |
| Дата сдачи курсовой работы: 21.12.2020 | | |
| Дата защиты курсовой работы: 21.12.2020 | | |
| Студент |  |  |
| Преподаватель |  | Татарникова Т.М. |

Аннотация

База данных «Школа» предназначена для завуча школы и обеспечивает в режиме диалога доступ к информации предметах, которые преподают, номере кабинета, который за закреплен за учителем, номера классов, в которых ведутся занятия, номере урока и дне, когда проходит уроки. Предусмотрена возможность как внесения изменений, так и получения справок с выводом информации на документ MS Word. Также программа по взаимодействию с БД позволяет выводить сведения о работе школы (количество учителей по предметам, количество кабинетов, число учеников в каждом классе, число троечников, хорошистов и отличников), а также другую полезную информацию.

База данных функционирует с ядром базы данных Microsoft SQL Server с интерфейсом Windows Forms на C# (Microsoft .NET Framework).

**Summary**

The "School" database is intended for the head teacher of the school and provides, in a dialogue mode, access to information on the subjects taught, the number of the classroom assigned to the teacher, the numbers of the classes in which classes are conducted, the number of the lesson and the day when lessons are held. It is possible to both make changes and get help with the output of information to a MS Word document. Also, the program for interacting with the database allows you to display information about the work of the school (the number of teachers in subjects, the number of classrooms, the number of students in each class, the number of C grade students, good students and excellent students), as well as other useful information.

The database operates with a Microsoft SQL database engine with a Windows Forms interface, project development language is C# (Microsoft .NET Framework).

Содержание

[Введение 7](#_Toc59029934)

[1. Анализ предметной области 8](#_Toc59029935)

[1.1 Подробное описание объектов предметной области 8](#_Toc59029936)

[1.2 Определение сущностей (объектов) 9](#_Toc59029937)

[1.3 Информационные объекты (сущности) 9](#_Toc59029938)

[1.4 Схема данных 9](#_Toc59029939)

[1.5 Формулировка задачи 10](#_Toc59029940)

[1.6 Ограничение на информацию БД 10](#_Toc59029941)

[1.7 Краткое описание алгоритмов решения задач 11](#_Toc59029942)

[1.8 Определение групп пользователей 11](#_Toc59029943)

[1.9 Описание выходных документов, которые должны генерироваться в системе 12](#_Toc59029944)

[1.10 Описание входных документов для заполнения БД 12](#_Toc59029945)

[2. Обоснование модели данных 14](#_Toc59029946)

[3. Проектирование БД 16](#_Toc59029949)

[3.1 Нормализация, схема базы данных 16](#_Toc59029950)

[3.2 Физическое проектирование базы данных 18](#_Toc59029951)

[4. Обоснование выбора СУБД 22](#_Toc59029952)

[4.1 Тип модели данных и её адекватность потребностям рассматриваемой предметной области. 22](#_Toc59029956)

[4.2 Характеристики производительности и набор функциональных возможностей 22](#_Toc59029957)

[4.3 Стоимость СУБД и дополнительного ПО 23](#_Toc59029958)

[4.4 Удобство и надежность СУБД в эксплуатации 23](#_Toc59029959)

[5. Описание функций управления данными 24](#_Toc59029960)

[5.1 Хранение (создание информационных объектов) 24](#_Toc59029962)

[5.2 Манипулирование (добавление, изменение, удаление, поиск данных) 25](#_Toc59029963)

[5.3 Доступ к данным (назначение прав доступа) 27](#_Toc59029964)

[5.4 Предоставление запрашиваемых данных пользователю (генерация справок, отчетов, итогов) 29](#_Toc59029965)

[6. Организация защиты БД 33](#_Toc59029966)

[6.1 Описание ограничений целостности для каждого информационного объекта. 33](#_Toc59029972)

[6.2 Рекомендуемые средства физической защиты (виды резервного копирования и периодичность проведения резервного копирования) 37](#_Toc59029973)

[6.3 Описание процедуры подтверждения личности 39](#_Toc59029974)

[6.4 Применение протоколов шифрования для защиты базы данных 40](#_Toc59029977)

[Заключение 42](#_Toc59029978)

[Список использованных источников 44](#_Toc59029979)

[Приложение А. Руководство ползователя БД 45](#_Toc59029980)

[Приложение Б. Листинг программного кода SQL 54](#_Toc59029991)

[Приложение В. Листинг программного кода C# 57](#_Toc59029993)

Введение

Цель выполнения курсовой работы – применение на практике знаний, полученных в процессе изучения курса «Управление данными», и получение практических навыков проектирования баз данных.

Процесс автоматизации обработки информации обычно рассматривается с учетом двух компонентов: данных и алгоритма обработки. Сформулированы стандартные требования к организации данных:

* интеграция данных, в этом случае создается динамическая модель предметной области, в рамках которой работает автоматизированная информационная система;
* максимально возможная независимость прикладных программ от данных.

Выполнение этих требований привело к появлению единого блока данных (базы данных (БД) и разработке одной управляющей программы для манипулирования данными на физическом уровне (системы управления данными СУБД).

В качестве системы управления базой данных в данной курсовой работе используется СУБД MS SQL Server.

1. Анализ предметной области

* 1. Подробное описание объектов предметной области

Предметной областью базы данных является информация о школе, об учителях, учеников, их оценок.

* Для каждого учителя в БД должны храниться: сведения о предметах, которые он преподает, номере кабинета, который за ним закреплен, номера классов, в которых он ведет занятия, номере урока и дне, когда он ведет уроки. Существуют учителя, которые не имеют собственного кабинета.
* Об учениках должны храниться следующие сведения: фамилия и имя, в каком классе учится, какая оценка по каждому предмету получена. Ученик может исправить полученную оценку.
* В БД также должно хранится расписание школы.

Программой по взаимодействию с БД будет управлять завуч школы.

Завуч может добавить информацию о новом учителе или ученике, а также удалить - выбывших.

Завучу могут потребоваться следующие сведения:

* какой предмет будет в заданном классе, например, во вторник на заданном уроке;
* кто из учителей преподает в заданном классе;
* в каком кабинете будет 5-й урок в среду у некоторого класса;
* в каких классах преподает учитель заданный предмет;
* расписание на заданный день недели для класса.

Завуч может вносить следующие изменения:

* вносить информацию о новом учителе;
* удалять запись об ученике;
* изменить оценку ученику.

Необходимо предусмотреть возможность выдачи справки и

В виде отчета (генерируется документ Word) завучу доступны следующие данные:

* справка о количестве учеников в данном классе
* отчета о работе школы (количество учителей по предметам, количество кабинетов, число учеников в каждом классе, число троечников, хорошистов и отличников).
  1. Определение сущностей (объектов)

После анализа предметной области выделим следующие сущности:

* Учитель;
* Ученик;
* Расписание.
  1. Информационные объекты (сущности)

Сущности имеют следующие атрибуты:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Учитель |  | Ученик |  | Расписание |
| ID учителя |  | ID ученика |  | День |
| ФИО |  | ФИО |  | Класс |
| Кабинет |  | Класс |  | Номер урока |
|  |  |  |  | Предмет |
|  |  |  |  | Преподаватель |
|  |  |  |  | Кабинет |

* 1. Схема данных

Для определения схемы данных проектируемой базы данных, то есть построения ее инфологической модели, необходимо определить сущности и связи БД. С помощью сущности моделируется класс однотипных объектов. Сущность имеет имя, уникальное в пределах моделируемой системы. Сущность соответствует некоторому классу однотипных объектов, поэтому предполагается, что в системе существует множество экземпляров данной сущности. Объект, которому соответствует понятие сущности, имеет свой набор атрибутов - характеристик, определяющих свойства данного представителя класса. Между сущностями могут быть установлены связи - бинарные ассоциации, показывающие, каким образом сущности соотносятся или взаимодействуют друг с другом. Связь может существовать между двумя разными сущностями или между сущностью и ей же самой (рекурсивная связь).

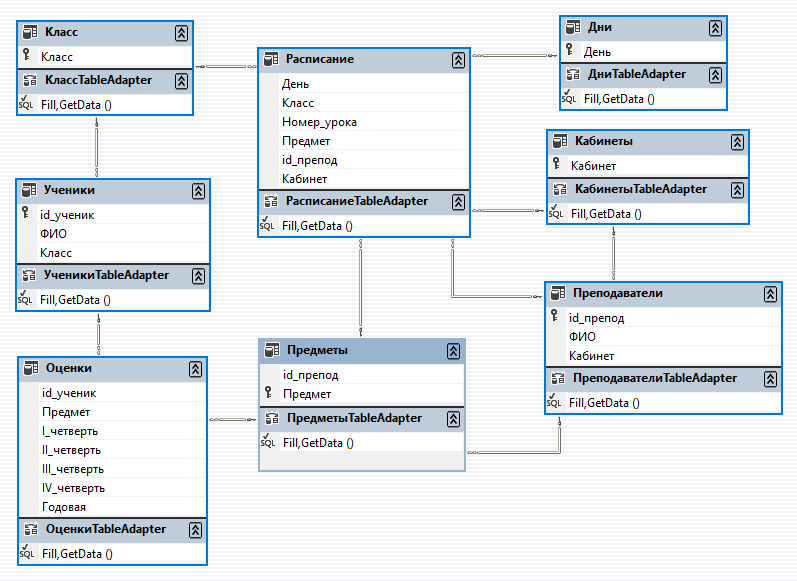


Рисунок 1 – Инфологическая модель БД школы

* 1. Формулировка задачи

Хранить информацию об учениках, преподавателях, расписании и оценках учеников.

Печать справки о количестве учащихся в определенном классе и отчета по школе.

* 1. Ограничение на информацию БД
* Каждый ученик может быть только в одном учебном классе,
* Не каждый преподаватель имеет свой кабинет,
* Ученик не может получить оценку по несуществующему предмету,
* В расписании нельзя поставить урок на воскресенье,
  1. Краткое описание алгоритмов решения задач

1. Хранение информации об учениках, преподавателях, расписании и оценках учеников.

Информация об учениках, преподавателях, расписании и оценках учеников хранятся в 8 связанных (по внешним ключам) таблицах, связи организованы таким образом, чтобы обеспечить корректную работу системы и целостность базы данных. Например, чтобы уволить преподавателя (удалить данные преподавателя с базы данных), надо сначала снять его с расписания. Такое ограничение не позволяет появления ситуации, когда урок по расписании есть, а учителя нет (его уволили).

1. Печать справок о числе учеников и отчета по школе.

Для печати справок используется запрос к базе данных, а с полученными данными создается MS Word документ и заполняется (подробное описание приведено в 5-ом разделе).

* 1. Определение групп пользователей

Пользователь будет один – завуч школы.

Завучу доступны следующие сведения:

* информация об учителях;
* информация об учениках;
* информация о расписании;
* информация об оценках учеников;
* какой предмет будет в заданном классе, например, во вторник на заданном уроке;
* кто из учителей преподает в заданном классе;
* в каком кабинете будет 5-й урок в среду у некоторого класса;
* в каких классах преподает учитель заданный предмет;
* расписание на заданный день недели для класса.

В виде отчета (генерируется документ Word) завучу доступны следующие данные:

* справка о количестве учеников в данном классе;
* отчет о работе школы (количество учителей по предметам, количество кабинетов, число учеников в каждом классе, число троечников, хорошистов и отличников).

Завуч может вносить следующие изменения:

* вносить информацию о новом учителе;
* удалить информацию о учителе;
* добавить запись об ученике;
* удалять запись об ученике;
* изменить оценку ученику.
* изменить оценку ученика;
* добавить позицию в расписании;
* удалить позицию из расписания;
  1. Описание выходных документов, которые должны генерироваться в системе
* справка о количестве учеников в данном классе;
* отчет о работе школы (количество учителей по предметам, количество кабинетов, число учеников в каждом классе, число троечников, хорошистов и отличников).
  1. Описание входных документов для заполнения БД
* Паспорт преподавателя;

(документ, для удостоверения личности).

* Свидетельство о рождении ученика;

(документ, для удостоверения личности).

* справка об оценках.

(если ученик был переведен с другой школы)

2. Обоснование модели данных

В базе данных будут хранятся структурированные данные, как следствие, выбираем модель реляционной базы данных.

Реляционная модель — это набор отношений нескольких двумерных таблиц, в каждой из которых хранится определенная информация.

Основными достоинствами этой модели является удобство, наглядность, возможность осуществления связи многие ко многим, порядок строк и столбцов не существенен, поэтому для данной предметной области эта модель наиболее приемлема.

В реляционной модели достигается гораздо более высокий уровень абстракции данных, чем в иерархической или сетевой. Реляционная модель предоставляет средства описания данных на основе только их естественной структуры, т.е. без потребности введения какой-либо дополнительной структуры для целей машинного представления. Другими словами, представление данных не зависит от способа их физической организации. Это обеспечивается за счет использования математической теории отношений (само название "реляционная" происходит от английского relation - "отношение").

Структура информации дает основание предполагать, что наиболее подходящей для даталогического проектирования будет реляционная модель данных, т.к. она способна обеспечит целостность данных при вставке, удалении и изменении записей, а так же дает возможность организации всех видов связей 1:1, 1:М и М:М (при этом связи М:М раскрываются). К недостаткам традиционных реляционных моделей данных можно отнести является избыточность по полям (из-за создания связей), а также факт того, что в качестве основного и, часто, единственного механизма, обеспечивающего быстрый поиск и выборку отдельных строк таблице (или в связанных через внешние ключи таблицах), обычно используются различные модификации индексов, основанных на B-деревьях. Такое решение оказывается эффективным только при обработке небольших групп записей и высокой интенсивности модификации данных в базах данных.



На рисунке 2 показана реализация реляционной модели данных.

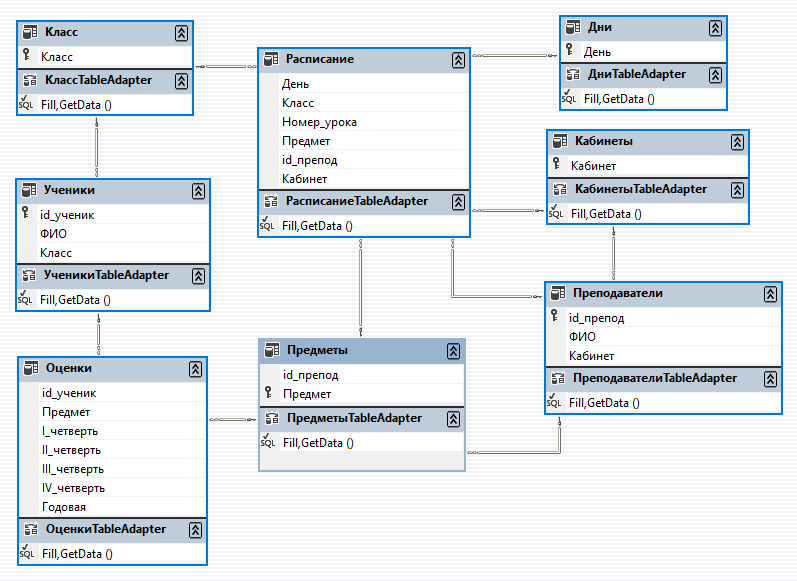


Рисунок 2. Реляционная модель данных.

2. Проектирование БД
   1. Нормализация, схема базы данных

Проектирование схемы БД может быть выполнено двумя путями:

* путем декомпозиции (разбиения), когда исходное множество отношений, входящих в схему БД, заменяется другим множеством отношений (число их при этом возрастает), являющихся проекциями исходных отношений;
* путем синтеза, т.е. путем компоновки из заданных исходных элементарных зависимостей между объектами предметной области схемы БД.

Классическая технология проектирования реляционных БД связана с теорией нормализации, основанной на анализе функциональных зависимостей между атрибутами отношений.

Процесс проектирования с использованием декомпозиции представляет собой процесс последовательной нормализации схем отношений, при этом каждая последующая итерация соответствует нормальной форме более высокого уровня и обладает лучшими свойствами по сравнению с предыдущей.

В теории реляционных БД обычно выделяется следующая последовательность нормальных форм:

* 1-я (1NF);
* 2-я (2 NF);
* 3-я (3 NF);
* Бойса-Кодда (ВС-NF);
* 4-я (4 NF);
* 5-я (5 NF) или форма проекции-соединения (PJNF).

Отношение находится в 1-й нормальной форме тогда и только тогда, когда на пересечении каждого столбца, и каждой строки находятся только элементарные значения атрибутов.

Отношения, находящиеся в 1-й нормальной форме, часто называют просто нормализованными отношениями, поэтому ненормализованные отношения могут интерпретироваться как таблицы с неравномерным заполнением.

В проектируемой БД имеется отношение расписание, которое имеет вид:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| День | Класс | № урока | Предмет | Id преподавателя | Кабинет |
| Понедельник | 1А | 1 | Русский язык | 1 | 12 |
|  |  | 2 | Математика | 2 |  |
|  |  | 3 | Физкультура | 4 | 1 |

Теперь приведем данное отношение к первой нормальной форме:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| День | Класс | № урока | Предмет | Id преподавателя | Кабинет |
| Понедельник | 1А | 1 | Русский язык | 1 | 12 |
| Понедельник | 1А | 2 | Математика | 2 | 12 |
| Понедельник | 1А | 3 | Физкультура | 4 | 1 |

Отношение находится во 2-й нормальной форме тогда и только тогда, когда оно находится в 1-й нормальной форме и не содержит неполных функциональных зависимостей не первичных атрибутов первичного ключа.

В базе данных школы должны храниться оценки учеников. Структура данного отношения может иметь вид:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ID | ФИО | Класс | Предмет | I четверть | II | III | IV | Годовая |

Первичным ключом данного отношения могут атрибуты «ID ученика» и «Предмет». С другой стороны, атрибуты «ФИО» и «Класс» зависят только от части первичного ключа - от значения атрибута «ID», поэтому необходимо констатировать наличие неполных функциональных зависимостей в данном отношении. Для приведения данного отношения ко второй нормальной форме следует разбить его на проекции, при этом должно быть соблюдено условие восстановления исходного отношения без потерь.

Такими проекциями могут быть два отношения:

(ID, ФИО, класс),

(ID, предмет, I четверть, II, III, IV, Годовая оценка).

Этот набор отношений не содержит неполных функциональных зависимостей, поэтому эти отношения находятся во 2-й нормальной форме.

Отношение находится в 3-й нормальной форме тогда и только тогда, когда оно находится во 2-й нормальной форме и не содержит транзитивных зависимостей.

Отношение находится в нормальной форме Бойса-Кодда, если оно находится в 3-й нормальной форме и каждый детерминант отношения является возможным ключом отношения.

Отношение R находится в 4-й нормальной форме (4NF) в том и только в том случае, если в случае существования многозначной зависимости А->> В все остальные атрибуты R функционально зависят от А.

Все отношения проектируемой БД, которые представлены в пункте 1.4. Схема данных являются приведенными к четвертой нормальной форме.

* 1. Физическое проектирование базы данных

В спроектированной базе данных школы созданы следующие таблицы:

Кабинеты

Классы

Оценки

Предметы

Преподаватели

Расписание

Ученики

Дни

В таблице Кабинеты содержится список существующих кабинетов. Поле состоит из одного поля - Кабинет, которое является ключевым.

*Таблица 1. Кабинеты*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| Кабинет | INT | PRIMARY KEY,  NOT NULL |

Таблица Классы содержит список существующих классов. Состоит из одного поля - Класс, которое является ключевым.

*Таблица 2. Классы*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| Класс | NVARCHAR (5) | PRIMARY KEY,  NOT NULL |

Таблица Ученики состоит из следующих полей: ID (ключевое, счетчик), ФИО (текстовое), Класс (текстовое).

*Таблица 3. Ученики*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_ученик | INT | PRIMARY KEY,  IDENTITY (1, 1)  NOT NULL |
| ФИО | NVARCHAR (30) | NOT NULL |
| Класс | NVARCHAR (5) | NOT NULL |

В таблице Оценки хранятся оценки, полученные учениками. Таблица состоит из следующих полей: ID (ID ученика, числовое), Предмет (текстовое, источник строк - таблица Предметы), I четверть (числовое), II четверть, III четверть, IV четверть, Годовая.

*Таблица 4. Оценки*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_ученик | INT | NOT NULL |
| Предмет | NVARCHAR (30) | NOT NULL |
| I\_четверть | INT | NULL |
| II\_четверть | INT | NULL |
| III\_четверть | INT | NULL |
| IV\_четверть | INT | NULL |
| Годовая | INT | NULL |

В таблице Преподаватели хранится список преподавателей. Таблица состоит из следующих полей: id\_препод (ID преподавателя, ключевое, счетчик), Преподаватель (ФИО, текстовое), Кабинет (закрепленный кабинет - необязательное поле, источник строк - таблица Кабинеты, числовое).

*Таблица 5. Преподаватели*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_препод | INT | PRIMARY KEY,  IDENTITY (1, 1)  NOT NULL |
| ФИО | NVARCHAR (30) | NOT NULL |
| Кабинет | INT | NULL |

Преподаватель может вести несколько предметов. В таблице Предметы хранятся данные, какие предметы ведет тот или иной преподаватель. Таблица состоит из полей id\_препод (ID преподавателя) и поля Предмет (текстовое)..

*Таблица 6. Преподаватели*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_препод | INT | NOT NULL |
| Предмет | NVARCHAR (20) | NOT NULL |

В таблице Расписание хранится расписание занятий школы. Таблица состоит из следующих полей: День (текстовое поле, источник строк - список значений, дни недели), Класс, Номер\_урока (числовое поле), Предмет, id\_преп (ID преподавателя), Кабинет.

*Таблица 7. Расписание*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| День | NVARCHAR (20) | NOT NULL |
| Класс | NVARCHAR (5) | NOT NULL |
| Номер\_урока | INT | NOT NULL |
| Предмет | NVARCHAR (20) | NOT NULL |
| id\_препод | INT | NOT NULL |
| Кабинет | INT | NOT NULL |

В таблице Дни хранится дни недели. Таблица состоит из одного поля: День (текстовое поле, ключевое).

*Таблица 8. Дни*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| День | NVARCHAR (20) | PRIMARY KEY,  NOT NULL |

Установим связь между таблицами с помощью внешних ключей (FOREIGN KEY).

Построим схему получившейся БД (см. рис. 3):

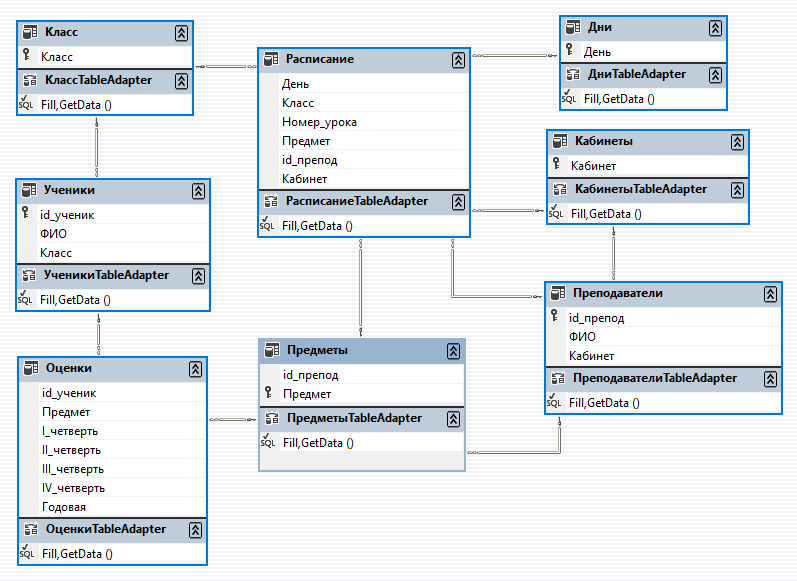


Рисунок 3. Схема реляционной модели БД «Школа».

4. Обоснование выбора СУБД

3. 1. Тип модели данных и её адекватность потребностям рассматриваемой предметной области.

Для реализации курсового проекта я выбрал СУБД Microsoft SQL Server 2019. Microsoft SQL Server 2019 — система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов — Transact-SQL (T-SQL). T-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

* 1. Характеристики производительности и набор функциональных возможностей

В отличие от остальных СУБД, SQL Server обеспечивает интеграцию с Microsoft Office, гарантирует повышенную безопасность и производительность средств разработки, содержит более мощные инструменты бизнес-аналитики по сравнению с решением Oracle. Кроме того, SQL Server имеет более низкую совокупную стоимость владения.

Среди основных преимуществ MS SQL Server можно выделить следующие:

* Масштабируемость и производительность.
* База данных менее уязвима.
* Инструменты бизнес-аналитики с поддержкой самообслуживания.

MS SQL Server предоставляет такие опции, как высокая доступность, усиленная безопасность, улучшенное сжатие данных, сервисы интеграции.

* 1. Стоимость СУБД и дополнительного ПО

Из представленных на рынке в данный момент решений для систем управления реляционными базами данных Microsoft SQL Server является наилучшим по соотношению цена-возможности.

Из дополнительное ПО для данного проекта требуется наличие офисного пакета Microsoft Office, в частности Microsoft Word, для генерации справок и отчетов.

Все взаимодействия с базой данных осуществляются с помощью графической оболочки программы в режиме диалогового окна Windows Forms, написанной на языке программирования C# (Microsoft .NET Framework 4.7.2).

* 1. Удобство и надежность СУБД в эксплуатации

Высокая доступность обеспечивается в первую очередь за счёт отказоустойчивой кластеризации. Отказоустойчивая кластеризация обеспечивает защиту не только базы данных, но и сервера. Данная функция позволяет предотвратить любую потерю данных, что является важным аспектом для заказчика.

Кроме того, высокая доступность обеспечивается за счёт зеркалирования базы данных. В случае сбоя на главном сервере, клиенты автоматически перенаправляются на зеркальный сервер.

В MS SQL Server особое внимание уделяется безопасности. Базовые выпуски SQL Server обладают расширенными функциями обеспечения безопасности, тогда как, например, стандартные и корпоративные выпуски Oracle обеспечивают лишь базовую безопасность.

5. Описание функций управления данными

1. 1. Хранение (создание информационных объектов)

Информационные объекты были созданы согласно таблицам 1-8 (см. раздел 3).

CREATE TABLE [dbo].[Кабинеты] (

    [Кабинет] INT NOT NULL,

    PRIMARY KEY CLUSTERED ([Кабинет] ASC)

)

GO

CREATE TABLE [dbo].[Класс] (

    [Класс]  NVARCHAR (5)  NOT NULL,

    PRIMARY KEY CLUSTERED ([Класс] ASC)

)

GO

CREATE TABLE [dbo].[Ученики] (

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

    [ФИО]           NVARCHAR (30) NOT NULL,

    [Класс]         NVARCHAR (5)  NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_ученик] ASC)

)

GO

CREATE TABLE [dbo].[Оценки] (

    [id\_ученик]     INT           NOT NULL,

    [Предмет]       NVARCHAR (20) NOT NULL,

    [I\_четверть]    INT           NOT NULL,

    [II\_четверть]   INT           NOT NULL,

    [III\_четверть]  INT           NOT NULL,

    [IV\_четверть]   INT           NOT NULL,

    [Годовая]       INT           NOT NULL

)

GO

CREATE TABLE [dbo].[Преподаватели] (

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

    [ФИО]           NVARCHAR (30) NOT NULL,

    [Кабинет]       INT           NULL,

    PRIMARY KEY CLUSTERED ([id\_препод] ASC)

)

GO

CREATE TABLE [dbo].[Предметы] (

    [id\_препод]     INT           NOT NULL,

    [Предмет]       NVARCHAR (20) NOT NULL,

    PRIMARY KEY CLUSTERED ([Предмет] ASC)

)

GO

CREATE TABLE [dbo].[Дни] (

    [День]       NVARCHAR (20) NOT NULL,

    PRIMARY KEY CLUSTERED ([День] ASC)

)

GO

CREATE TABLE [dbo].[Расписание] (

    [День]          NVARCHAR (20) NOT NULL,

    [Класс]         NVARCHAR (5)  NOT NULL,

    [Номер\_урока]   INT           NOT NULL,

    [Предмет]       NVARCHAR (20) NOT NULL,

    [id\_препод]     INT           NOT NULL,

    [Кабинет]       INT           NOT NULL,

)

GO

* 1. Манипулирование (добавление, изменение, удаление, поиск данных)

Для изначального добавления данных в базу использовались запросы SQL, пример запроса показано листинге ниже (полный текст запроса в приложении Б).

INSERT INTO Дни (День) VALUES(N'Понедельник')

INSERT INTO Кабинеты (Кабинет) VALUES(1)

INSERT INTO Преподаватели (ФИО, Кабинет) VALUES(N'Александрова Евa Васильевна', 16)

INSERT INTO Класс (Класс) VALUES(N'1А')

INSERT INTO Ученики (ФИО, Класс) VALUES(N'Лаврентьев Корней Федосеевич',N'1А')

INSERT INTO Предметы (id\_препод, Предмет) VALUES(19, N'Физика')

Для добавления данных из интерфейса программы используются параметризированное запросы. Для добавления нового ученика его данные заносятся в базу данных следующим запросом:

INSERT INTO [Ученики] (ФИО, Класс)

VALUES(@ФИО, @Класс)

С помощью встроенных средств языка C# вместо параметров @ФИО, @Класс подставляются соответствующие данные из определенных текстовых полей из интерфейса программы. В случае удачного выполнения в строке состояния программы выводится соответствующее сообщение

В случае возникновения какой-либо ошибки программа корректно обработает и сообщит об ошибке.

Запросы удаления данных из базы также реализованы с помощью параметров. Параметризированный запрос удаления аналогичен запросу добавления по части получения данных от пользователя.

При попытке увольнения преподавателя (удаление данных преподавателя из базы), показывается диалоговое окно с подтверждением.

Если в расписании есть урок с данным преподавателем, то его нельзя уволить. Сначала нужно будет снять преподавателя с расписания, а потом уже уволить. В случае, если преподаватель есть в расписании и завуч подтвердит удаление данных преподавателя, программа корректно обработает ошибку и выведет на экран диалоговое окно с описанием ошибки и действий по ее устранению.

Пример запроса удаления данных:

DELETE FROM [Ученики]

WHERE [id\_ученик] = @id\_ученик

Пример запроса обновления данных:

UPDATE [Оценки]

SET [Годовая] = @ Годовая

WHERE id\_ученик = @id\_ученик

Значения параметров @id\_ученик, @Годовая и @Класс берутся из соответствующих текстовых полей из интерфейса программы.

* 1. Доступ к данным (назначение прав доступа)

При запуске программы, запускается форма авторизации пользователя, в данном случае – завуча. Завуч вводит свой логин и пароль и в случае правильного ввода запускается основное окно программы (см. рис 4,5).

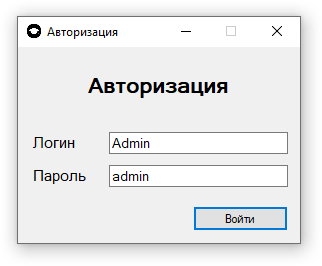


Рисунок 4 – Окно авторизации

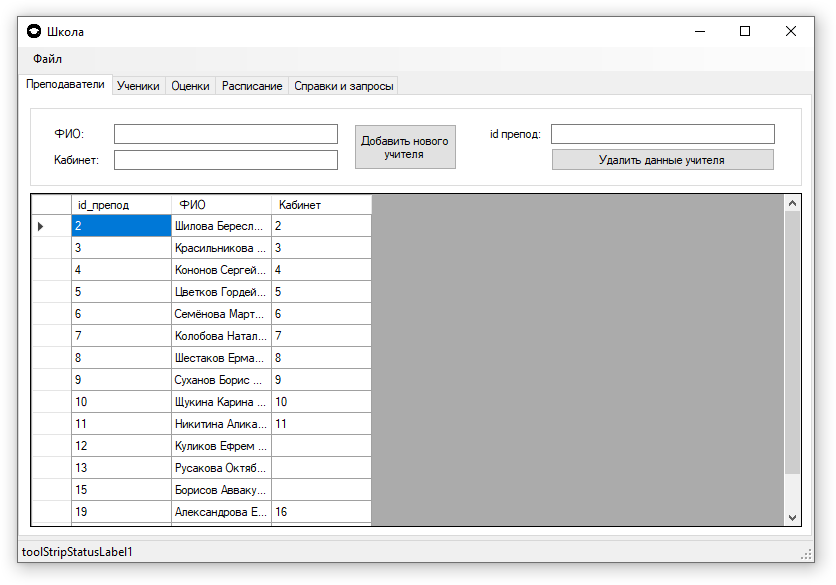


Рисунок 5 – Основное окно программы

Завуч использует свою учетную запись на сервере. Программа создает строку подключения к базе данных, заносит введенные данные для авторизации и пытается установить связь с базой данных, в случае успеха загружается основное окно программы.

Запрос для создания пользователя Завуч на сервере:

USE [master]

GO

CREATE LOGIN [Admin]

WITH PASSWORD='admin',

DEFAULT\_DATABASE=[SCHOOL],

CHECK\_EXPIRATION=OFF,

CHECK\_POLICY=OFF

GO

USE [SCHOOL]

GO

CREATE USER [Admin] FOR LOGIN [Admin]

GO

USE [SCHOOL]

GO

EXEC sp\_addrolemember N'db\_datawriter', N'Admin'

GO

Запрос на предоставление прав этому пользователю к тем же базам:

USE [SCHOOL]

GO

GRANT DELETE TO [Admin]

GO

USE [SCHOOL]

GO

GRANT INSERT TO [Admin]

GO

USE [SCHOOL]

GO

GRANT SELECT TO [Admin]

GO

USE [SCHOOL]

GO

GRANT UPDATE TO [Admin]

GO

* 1. Предоставление запрашиваемых данных пользователю (генерация справок, отчетов, итогов)

В программе все справки и отчеты можно получить во вкладке «Справки и запросы» (см. рис. 6).

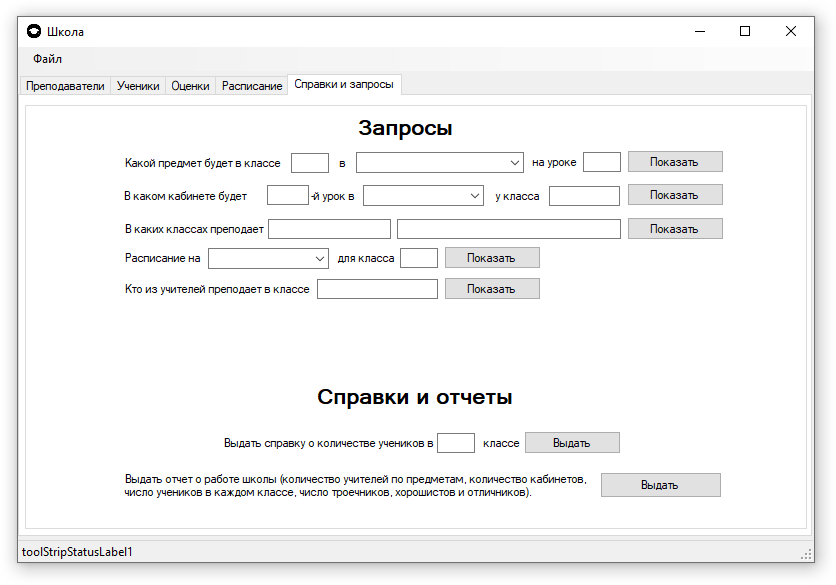


Рисунок 6 – Вкладка «Справки и запросы»

В разделе «Запросы» собраны все короткие запросы, которые выводятся на экран в режиме диалогового окна и показывают результат запроса (см. рис. 7).

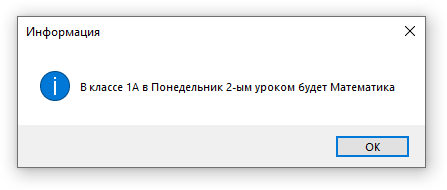


Рисунок 7 – Результат короткого запроса

Запросы возвращающие таблицы, выводятся в отдельном окне. (см. рис. 8). Окно работает в отдельном потоке, что позволяет, не закрывая результат запроса, продолжать работу в основной окне программы, что несомненно является большим плюсом.

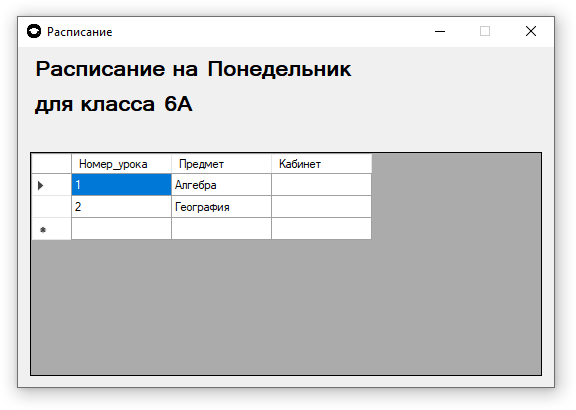


Рисунок 8 – Запросы на расписания класса 6А в понедельник

В разделе «Справки и отчеты» собраны все документы на печать, которые генерируются в программе.

Для получения справки о количестве учеников, необходимо заполнить соответствующее поле и нажать кнопку «Выдать» (см. рис. 9).

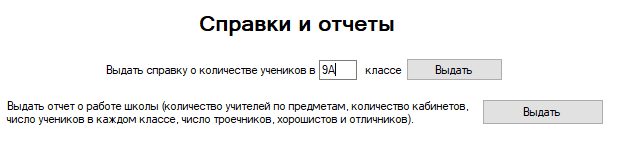


Рисунок 9 – Раздел «Справки и отчеты»

В результате открывается новый документ Microsoft Office Word, заполняется и форматируется документ.

Создание MS Word документов происходит в несколько этапов. Сначала выполняется запрос в базу, для получения нужных данных. Затем открывается MS Word, создается новый документ и включается отображение окна MS Word.

Word.\_Application oWord;

Word.\_Document oDoc;

oWord = new Word.Application();

oWord.Visible = true;

Далее начинается передача текста в MS Word и его форматирование (в данном случае: жирный, интервал после строки 30пт, выровнен по центру, перевод строки).

Word.Paragraph oPara1;

oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);

oPara1.Range.Text = "ООО \"СПРАВКА\"";

oPara1.Range.Font.Bold = 1;

oPara1.Format.SpaceAfter = 30;

oPara1.Range.ParagraphFormat.Alignment =

Word.WdParagraphAlignment.wdAlignParagraphCenter;

oPara1.Range.InsertParagraphAfter();

Аналогичными средствами строится остальная часть документа.

В разделе «Справки и отчеты» есть возможность генерации отчета о работе школы (количество учителей по предметам, количество кабинетов, число учеников в каждом классе, число троечников, хорошистов и отличников). До генерации отчета выполняются множество запросов по базе, все данные нужные данные собираются, потом создается документ.

6. Организация защиты БД

5. 1. Описание ограничений целостности для каждого информационного объекта.

Контроль достоверности данных осуществляется с помощью ограничений целостности. Для ограничения полей данных используются следующие виды ограничений:

* Тип и формат поля. Тип поля определяет допустимые для данного поля символы, а иногда и более жесткие ограничения на допустимые значения (как, например, для полей типа дата или логическое). В случае несоответствия, выводится диалоговое окно с ошибкой.
* Задание диапазона значений. Используется для числовых полей. Признак непустого поля, поля, состоящей только из пробелов, табуляций и null-значений. Характеризует недопустимость пустого значения поля в БД.
* На уровне программы выполняется проверка правильности введенных данных, попытки ввести строку в поле, предназначенное для целых значений, сопровождаются выводом ошибок в диалоговом окне с указанием на проблему и методы его решения.
* Задание домена. Поле может принимать значение из заданного множества. Например, нельзя добавить ученика в базу в класс, который не определен в базе.
* Специфическим ограничением на значение поля является признак его уникальности. Это ограничение проверяет допустимость значения данного поля, но при этом просматривается вся таблица.
* Очень важным ограничением целостности являются функциональные зависимости. Внешние ключи (*Foreign Key*) являются ограничением для поддержания логической целостности БД, гарантирующие непротиворечивость информации. Наличие внешних ключей не позволит пользователю удалить критически важные элементы базы данных. Например, до того, уволить преподавателя, нужно снять его с расписания.

Ограничение CHECK ограничивает данные, которые пользователь может ввести в определенную колонку указанными значениями. Следующий пример добавляет ограничение, чтобы гарантировать, что Предмет в таблице «Оценки» соответствует зарегистрированному в базе данных Предмету в таблице «Предметы»:

Создание внешних ключей для таблицы *Оценки:*

ALTER TABLE [dbo].[Оценки]

WITH CHECK ADD

CONSTRAINT [FK\_Оценики\_Предметы]

FOREIGN KEY([Предмет])

REFERENCES [dbo].[Предметы] ([Предмет])

GO

ALTER TABLE [dbo].[Оценки]

CHECK CONSTRAINT [FK\_Оценики\_Предметы]

GO

ALTER TABLE [dbo].[Оценки]

WITH CHECK ADD  CONSTRAINT [FK\_Оценики\_Ученики]

FOREIGN KEY([id\_ученик])

REFERENCES [dbo].[Ученики] ([id\_ученик])

GO

ALTER TABLE [dbo].[Оценки]

CHECK CONSTRAINT [FK\_Оценики\_Ученики]

GO

Создание внешних ключей для таблицы *Преподаватели:*

ALTER TABLE [dbo].[Преподаватели]

WITH CHECK ADD  CONSTRAINT [FK\_Преподаватели\_Кабинеты]

FOREIGN KEY([Кабинет])

REFERENCES [dbo].[Кабинеты] ([Кабинет])

GO

ALTER TABLE [dbo].[Преподаватели]

CHECK CONSTRAINT [FK\_Преподаватели\_Кабинеты]

GO

Создание внешних ключей для таблицы *Расписание:*

ALTER TABLE [dbo].[Расписание]

WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Кабинеты]

FOREIGN KEY([Кабинет])

REFERENCES [dbo].[Кабинеты] ([Кабинет])

GO

ALTER TABLE [dbo].[Расписание]

CHECK CONSTRAINT [FK\_Расписание\_Кабинеты]

GO

ALTER TABLE [dbo].[Расписание]

WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Предметы]

FOREIGN KEY([Предмет])

REFERENCES [dbo].[Предметы] ([Предмет])

GO

ALTER TABLE [dbo].[Расписание]

CHECK CONSTRAINT [FK\_Расписание\_Предметы]

GO

ALTER TABLE [dbo].[Расписание]

WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Преподаватели]

FOREIGN KEY([id\_препод])

REFERENCES [dbo].[Преподаватели] ([id\_препод])

GO

ALTER TABLE [dbo].[Расписание]

CHECK CONSTRAINT [FK\_Расписание\_Преподаватели]

GO

ALTER TABLE [dbo].[Расписание]

WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Класс]

FOREIGN KEY([Класс])

REFERENCES [dbo].[Класс] ([Класс])

GO

ALTER TABLE [dbo].[Расписание]

CHECK CONSTRAINT [FK\_Расписание\_Класс]

GO

ALTER TABLE [dbo].[Расписание]

WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Дни]

FOREIGN KEY([День])

REFERENCES [dbo].[Дни] ([День])

GO

ALTER TABLE [dbo].[Расписание]

CHECK CONSTRAINT [FK\_Расписание\_Дни]

GO

Создание внешних ключей для таблицы *Ученики:*

ALTER TABLE [dbo].[Ученики]

WITH CHECK ADD  CONSTRAINT [FK\_Ученики\_Класс]

FOREIGN KEY([Класс])

REFERENCES [dbo].[Класс] ([Класс])

GO

ALTER TABLE [dbo].[Ученики]

CHECK CONSTRAINT [FK\_Ученики\_Класс]

GO

Создание внешних ключей для таблицы *Предметы:*

ALTER TABLE [dbo].[Предметы]

WITH CHECK ADD  CONSTRAINT [FK\_Предметы\_Преподаватели]

FOREIGN KEY([id\_препод])

REFERENCES [dbo].[Преподаватели] ([id\_препод])

GO

ALTER TABLE [dbo].[Предметы]

CHECK CONSTRAINT [FK\_Предметы\_Преподаватели]

GO

* 1. Рекомендуемые средства физической защиты (виды резервного копирования и периодичность проведения резервного копирования)

Резервное копирование баз данных является самым простым и дешевым средством обеспечения сохранности данных. Резервная копия базы данных представляет собой копию данных, структур и объектов безопасности, содержащихся в базе данных. База данных резервируется по своему графику в зависимости от количества выполняемых за день транзакций записи. Для минимизации потерь при сбое базы данных выполняется резервное копирование базы данных. Чтобы убедиться, что резервная копия работоспособна, проверяется его работа после операции восстановления.

В данной работе применяется стратегия полного резервирования, что является самой простой для понимания и реализации. В конце каждого рабочего дня просто запускается процедура полного резервирования базы данных. При этом не нужно выполнять отдельное резервирование журналов и не требуется использовать дополнительные параметры. Управление файлами в таком режиме резервирования также не требует особого внимания, так как речь идет о единственном файле полной резервной копии. Восстановление из полной резервной копии тоже очень простое: необходимо просто восстановление из единственного файла. Использование полных резервных копий – хороший выбор для такого маленького проекта.

Когда SQL Server осуществляет полное резервирование базы данных, сначала выполняется сохранение на диск всех экстентов (экстент представляет собой восемь идущих последовательно страниц, размер каждой составляет 8 Кбайт). Затем SQL Server резервирует журнал транзакций, чтобы все изменения базы данных, которые могли произойти за время резервирования, также были сохранены в файле полной резервной копии.

Когда используется только полное резервирование, то в случае краха системы часть данных может быть потеряна – в первую очередь изменения, выполненные с момента последнего резервирования.

Для выполнения полного резервирования базы данных выполняется следующий код:

string name = "School-BackUP";

string path = @"C:\SQLdata\BACKUPS\";

// если каталог не существует то создать

DirectoryInfo dirInfo = new DirectoryInfo(path);

if (!dirInfo.Exists)

{

    Directory.CreateDirectory(path);

}

// "собираем" имя для резервной копии

var backupFileName = String.Format("{0}{1}-{2}.bak",

    path, name, DateTime.Now.ToString("yyyy-MM-dd-hh-mm-ss"));

//"собираем" запрос

var query = String.Format("BACKUP DATABASE [SCHOOL] TO DISK = '{0}'", backupFileName);

// выполняем запрос

SqlCommand command = new SqlCommand(query, sqlConnection);

command.ExecuteNonQuery();

В итоге запрос на создание резервной копии выглядит так

BACKUP DATABASE [SCHOOL]

TO DISK =

'C:\SQLdata\BACKUPS\SCHOOL-BackUP-yyyy-MM-dd-hh-mm-ss.bak'

где yyyy-MM-dd-hh-mm-ss текущая дата и время на компьютере.

Резервное копирование базы данных выполняется по расписанию.

* 1. Описание процедуры подтверждения личности

Программа запускается с окна авторизации (см. рис. 10)

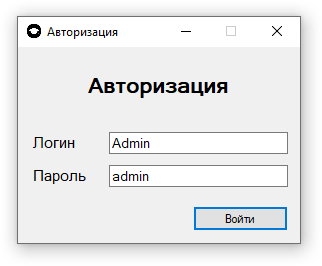


Рисунок 10 – Окно авторизации

Данной программой по взаимодействию с базой данных будет управлять только один пользователь – завуч школы. Для всех запросов, указанных в задании на курсовой проект, которые должны быть доступны завучу школы, есть соответствующая кнопка в интерфейсе приложения. Операции, которые не должны быть доступны завучу, например удаление таблицы из базы, или всей базы, не могут быть выполнены, потому что для таких действий нет кнопок в программе.



Программа запускается с окна входа, проверяется правильность логина и пароля, если все верно, запускается основное окно, а форма авторизации закрывается автоматически.

При условии правильно введенных учетных данных устанавливается соединение с базой данных и загружается основное окно программы, в противном случае выводится ошибка о неправильно введенных учетных данных (см. рис. 11).

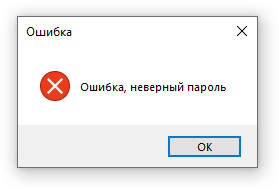
****

Рисунок 11 – Диалоговое окно ошибки о неправильных учетных данных



Количество групп пользователей: 1;

Группа №1 – Завуч школы.

Права доступа к каждому информационному объекту задаются согласно таблице 9:

*Таблица 9. Назначение прав доступа к информационным объектам*

|  |  |
| --- | --- |
| **Объект** | **Группа №1** |
| Ученики | SIUD |
| Преподаватели | SIUD |
| Расписание | SIUD |
| Оценки | SIUD |

Используемые сокращения:

|  |  |
| --- | --- |
| S – чтение данных (SELECT); | U – модификация данных (UPDATE); |
| I – добавление данных (INSERT); | D – удаление данных (DELETE). |

* 1. Применение протоколов шифрования для защиты базы данных

Шифрование представляет собой способ скрытия данных с помощью ключа или пароля. Это делает данные бесполезными без соответствующего ключа или пароля для дешифрования. Шифрование не решает проблемы управления доступом. Однако оно повышает защиту за счет ограничения потери данных даже при обходе системы управления доступом. Например, если компьютер, на котором установлена база данных, был настроен неправильно, и злоумышленник смог получить конфиденциальные данные, то украденная информация будет бесполезна, если она была предварительно зашифрована.

В SQL Server можно шифровать соединения, данные и хранимые процедуры, и позволяет использовать алгоритмы шифрования вплоть до AES с 256-разрядным ключом.

SQL Server шифрует данные, используя иерархическую структуру средств шифрования и управления ключами. На каждом уровне данные низшего уровня шифруются на основе комбинации сертификатов, асимметричных ключей и симметричных ключей. Асимметричные и симметричные ключи можно хранить вне модуля расширенного управления ключами SQL Server.

Шифрование может потребоваться, если пользователи получают доступ к данным через открытую сеть. Использование шифрования включает политику управления паролями, ключами и сертификатами.

Несмотря на то, что шифрование является полезным средством обеспечения безопасности, в рамках данной курсовой работы шифрование не применено.

Заключение

В рамках данного курсового проекта была спроектирована база данных и построена программа, обеспечивающая взаимодействие с базой данных в режиме диалога, для завуча школы. В БД хранятся информация о школе, об учителях, учеников, их оценок.

База данных функционирует в среде реляционной СУБД Microsoft SQL Server 2019, используемый язык запросов — Transact-SQL. Все взаимодействия с базой данных осуществляются с помощью графической оболочки программы в режиме диалогового окна Windows Forms, написанной на языке программирования C# (Microsoft .NET Framework 4.7.2).

Проведен анализ предметной области, инфологическое проектирование, нормализация схем, физическое проектирование, назначение прав доступа, определения и реализация ограничений целостности и другие процедуры для корректного функционирования и защиты базы данных.

Решены следующие задачи:

* хранение информации об учителях, учениках, оценках и расписании;
* выдача справки о числе учеников в заданном классе и отчета о работы школы;
* предусмотрена возможность корректировки БД в случаях поступления на работу нового преподавателя, отчисления ученика, введения нового или изменения оценки;
* разработана реляционная СУБД, содержащая элементы автоматизации и обработки данных.

Предусмотрена возможность выдачи следующих сведений:

* какой предмет будет в заданном классе, например, во вторник на заданном уроке;
* кто из учителей преподает в заданном классе;
* в каком кабинете будет 5-й урок в среду у некоторого класса;
* в каких классах преподает учитель заданный предмет;
* расписание на заданный день недели для класса.

Предусмотрена возможность внесения следующих изменений:

* вносить информацию о новом учителе;
* удалять запись об ученике;
* изменить оценку ученику.

В ходе выполнения данной курсовой работы были приобретены практические навыки работы и создания баз данных в среде Microsoft SQL Server 2019 и программ в среде Microsoft .NET Framework 4.7.2.

Список использованных источников

1. Рихтер Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. 4-е изд. — СПб.: Питер, 2013. — 896 с.
2. Шехтман В. Е. Базы данных, SQL и все такое: курс лекций / В. Е. Шехтман; НФИ КемГУ. - Новокузнецк, 2006. – 195 с.
3. Базы данных: учеб. пособие для студ. высших учебн. заведений / А.В. Кузин, С.В. Левонисова. – 2 – е изд. стер. – М.: Издательский центр «Академия», 2008.

Приложение А. Руководство ползователя БД

1. Краткое описание программы

Программа «Школа» предназначена для обеспечивания взаимодействия с базой данных в режиме диалога, для завуча школы. В БД информация о школе, об учителях, учеников, их оценок.

1. Системные требования

**Операционная система:** Windows 10 или более поздние версии

**Видеоадаптер:** DirectX версии не ниже 9 с драйвером WDDM 1.1.

**Дисплей:** не меньше 1024 x 768.

**Дополнительное ПО:** Microsoft .NET Framework версии 4.7.2

**Дополнительное ПО:** Microsoft Office Word 2019

1. Установка программы

Программа представляется в виде архивного файла «School.zip». Необходимо распаковать архив и запустить установочный файл «setup.exe» (см. рис. 12).

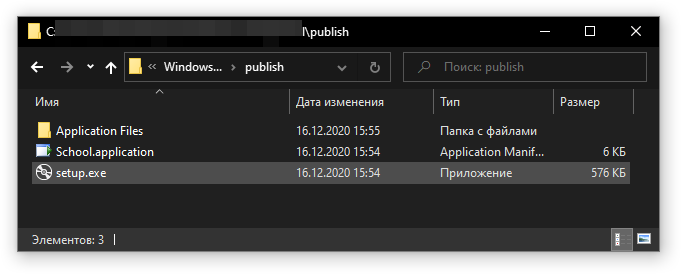


Рисунок 12– файл «setup.exe»

После двойного нажатия левой кнопки мыши происходит запуск установочного файла (см. рис. 13).

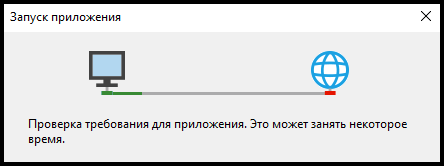


Рисунок 13 – запуск файла «setup.exe»

На экран выводится диалоговое окно «Установка приложения – Предупреждение о безопасности». Для подтверждения установки нажмите кнопку «Установить» (см. рис. 14).

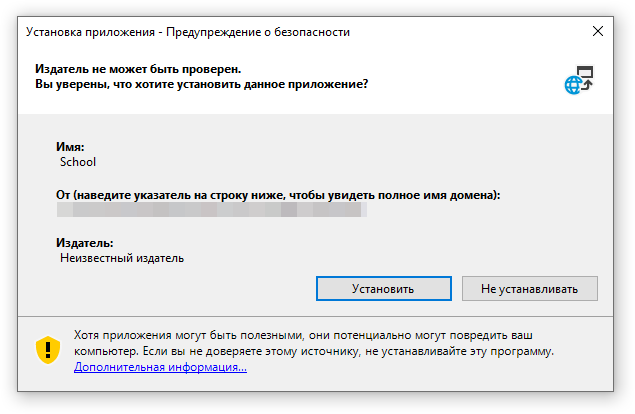


Рисунок 14 – Диалоговое окно «Установка приложения – Предупреждение о безопасности»

Начнется установка приложения, дождитесь его окончания.

После окончании процесса установки программа запуститься и выведет на экран форму авторизации (см. рис. 15). Кроме того, программа добавится к списку приложений и его можно будет найти в меню Пуск.

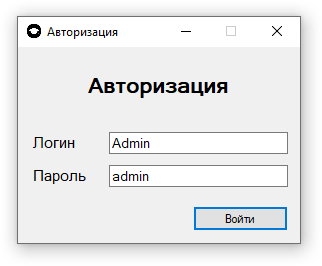


Рисунок 15 – Окно авторизации приложения «Школа»

1. Работа с программой

При открытии программы на экран выводится окно авторизации (см. рис. 15). Введите свои учетные данные нажмите кнопку «Войти». Логин и пароль по умолчанию представлены ниже, и они загружаются автоматически. В вы можете изменить логин и пароль, а также отключить автозаполнение в коде программы.

Логин: Admin

Пароль: admin

При правильно введенных учетных данных откроется основное окно программы (см. рис. 16).

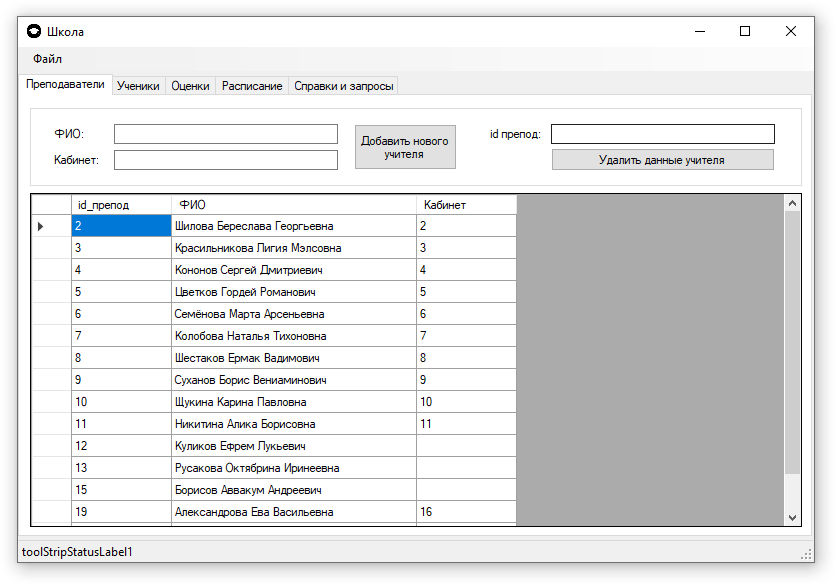


Рисунок 16 – Основное окно программы

Главное окно программы «Школа» имеет 5 вкладок: «Преподаватели», «Ученики», «Оценки», «Расписание» и «Справки и запросы». Подробно рассмотрим функционал каждой из них.

### 4.1 Вкладка «Преподаватели»

Вкладка «Преподаватели» предназначен для работы с данными преподавателей: добавления и удаления данных преподавателей (см. рис. 17)

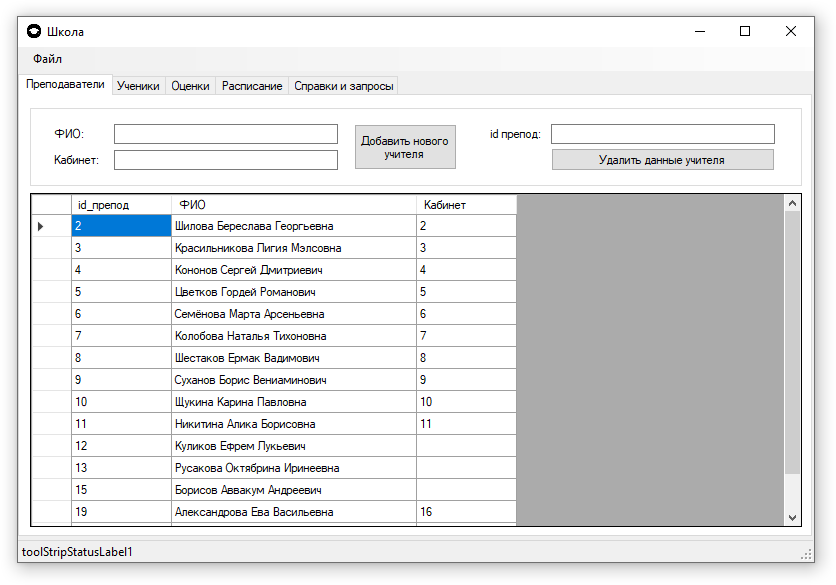


Рисунок 17 – Вкладка «Преподаватели»

Если пользователь попытается удалить преподавателя, которому назначен урок в расписании, то программа выдаст ошибку в новом диалоговом окне с описанием проблемы и возможным решением (см. рис. 18). Также текст ошибки будет дублироваться в строке состояния.

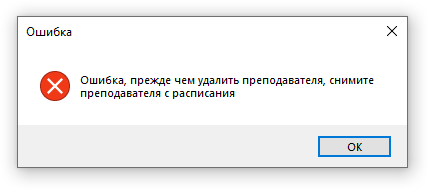


Рисунок 18 – Диалоговое окно ошибки

### 4.2 Вкладка «Ученики»

Вкладка «Ученики» предназначен для работы с данными учеников: добавления нового и удаления зарегистрированного ученика (см. рис. 19).

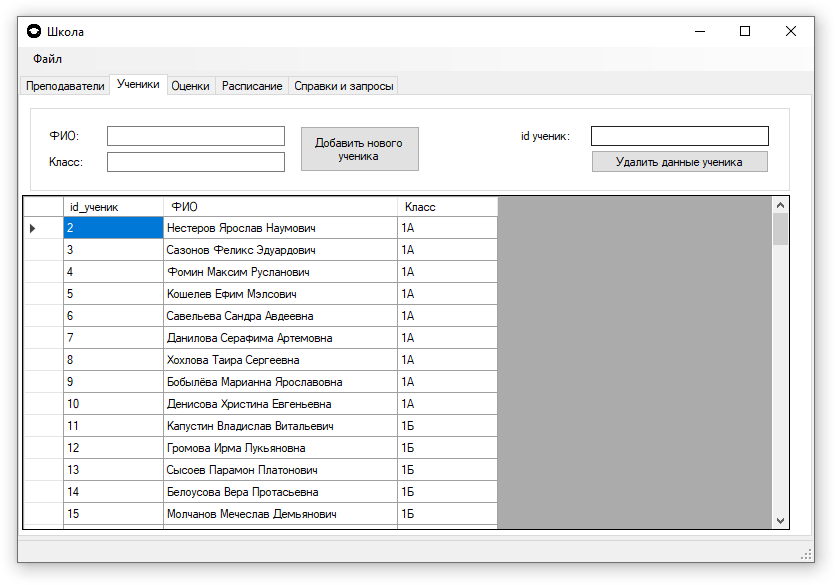


Рисунок 19 – Вкладка «Ученики»

Процесс добавления и удаления данных учеников аналогичен процессу добавления и удаления данных преподавателей.

### 4.3 Вкладка «Оценки»

Вкладка «Оценки» предназначен для просмотра и изменения оценок учеников (см. рис. 20).

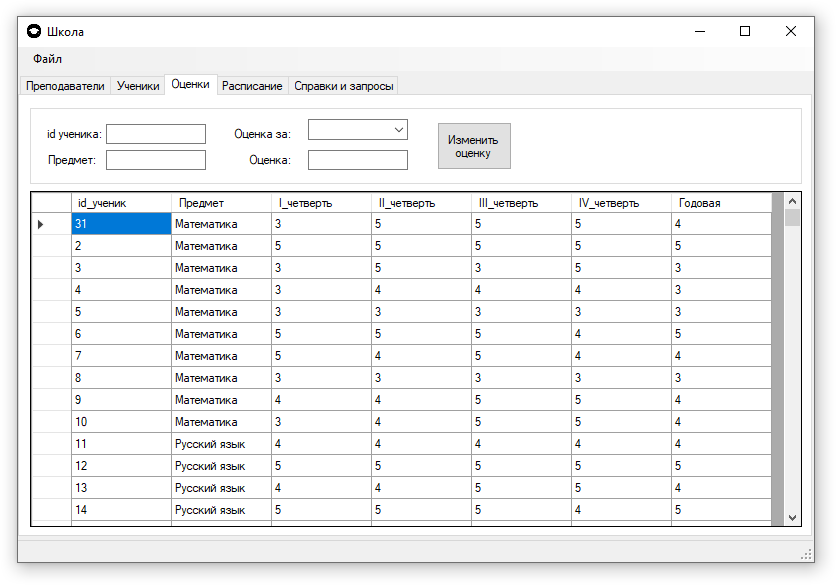


Рисунок 20– Вкладка «Оценки»

Для изменения оценки необходимо заполнить все поля: id\_ученика (для какого ученика будет изменена оценка), Предмет (за какой предмет), Оценка за (за какой четверть или годовая), Оценка (на какую оценку будет заменена оценка в базе) и нажать кнопку «Изменить оценку».

### 4.4 Вкладка «Расписание»

Вкладка «Расписание» предназначен для просмотра существующего расписания, а также для удаления и добавления позиций в расписании. (см. рис. 21). Как для добавления, так же для удаления позиций в расписании нужно заполнить все поля и нажать соответствующую кнопку в интерфейсе программы.

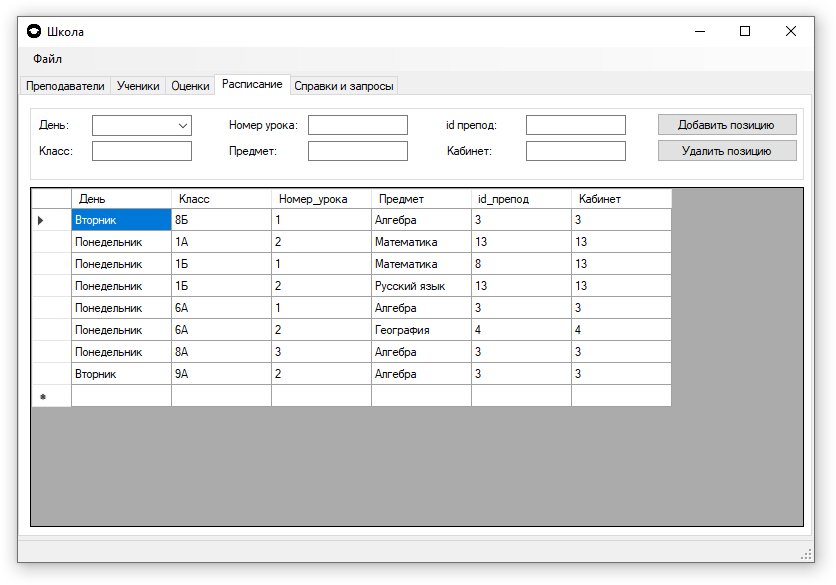


Рисунок 21 – Вкладка «Расписание»

### 4.5 Вкладка «Справки и запросы»

Вкладка «Справки и запросы» предназначена для предоставления справочной информации, а также генерации справок и отчетов (см. рис. 22).

Для выполнения запроса необходимо заполнить соответствующие поля перед кнопкой показать (разные поля для разных запросов) и нажать кнопку «Показать».

В зависимости от параметров запроса на экран выводится разное количество информации в разном виде. Если результат запроса содержит в себе несколько строк из соответствующей таблицы результат выводиться в новом окне в виде таблицы (см. рис. 23). Если параметры запроса содержат только одну строчку из соответствующей таблицы, то результат выводиться в виде текста в диалоговом окне «Информация» (см. рис. 24).

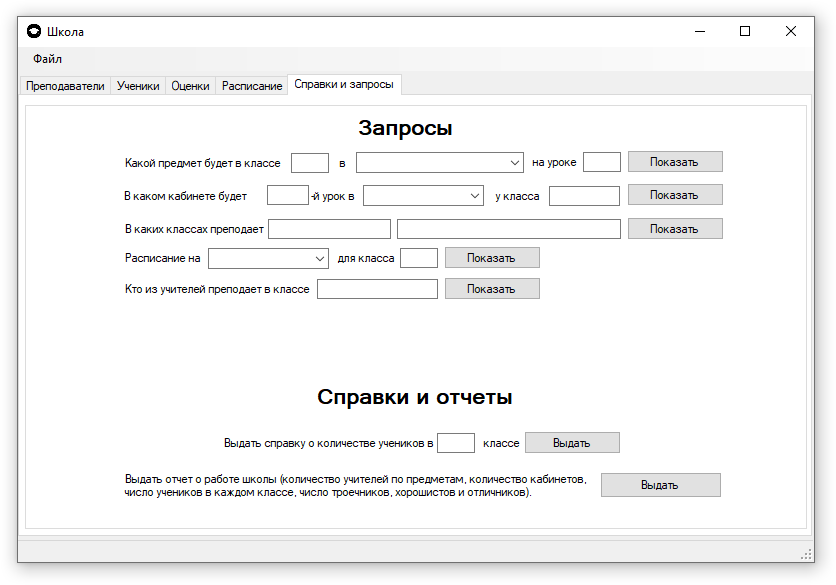


Рисунок 22 – Вкладка «Запросы и справки»

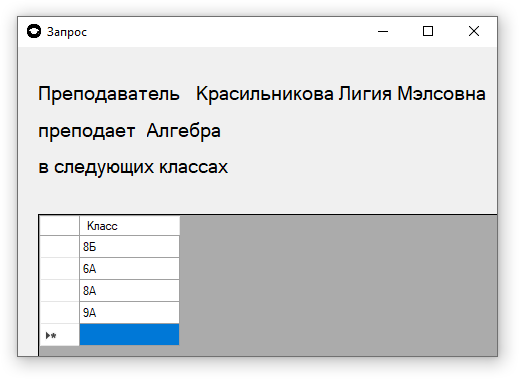


Рисунок 23 – Запрос списка классов с выбранным предметом для конкретного преподавателя

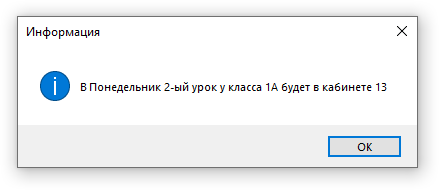


Рисунок 24 – Диалоговое окно с информацией

Для генерации справки необходимо заполнить поле для какого именно класса нужно сгенерировать справку и нажать кнопку «Выдать». Обратите внимание что классы названы русскими буквами (1А, А-символ русского алфавита).

### 4.7 Контекстное меню

Контекстное меню «Файл» содержит следующие элементы (см. рис. 25):

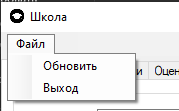


Рисунок 25 – Контекстное меню «Файл»

* Обновить – обновляет содержимое вкладки.
* Выход – Закрывает программу.

Приложение Б. Листинг программного кода SQL

Листинг 1. Создание таблиц и установка зависимостей (внешних ключей)

CREATE TABLE [dbo].[Кабинеты] (

    [Кабинет] INT NOT NULL,

    PRIMARY KEY CLUSTERED ([Кабинет] ASC)

)

GO

CREATE TABLE [dbo].[Класс] (

    [Класс]  NVARCHAR (5)  NOT NULL,

    PRIMARY KEY CLUSTERED ([Класс] ASC)

)

GO

CREATE TABLE [dbo].[Ученики] (

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

    [ФИО]           NVARCHAR (30) NOT NULL,

    [Класс]         NVARCHAR (5)  NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_ученик] ASC)

)

GO

CREATE TABLE [dbo].[Оценки] (

    [id\_ученик]     INT           NOT NULL,

    [Предмет]       NVARCHAR (20) NOT NULL,

    [I\_четверть]    INT           NOT NULL,

    [II\_четверть]   INT           NOT NULL,

    [III\_четверть]  INT           NOT NULL,

    [IV\_четверть]   INT           NOT NULL,

    [Годовая]       INT           NOT NULL

)

GO

CREATE TABLE [dbo].[Преподаватели] (

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

    [ФИО]           NVARCHAR (30) NOT NULL,

    [Кабинет]       INT           NULL,

    PRIMARY KEY CLUSTERED ([id\_препод] ASC)

)

GO

CREATE TABLE [dbo].[Предметы] (

    [id\_препод]     INT           NOT NULL,

    [Предмет]       NVARCHAR (20) NOT NULL,

    PRIMARY KEY CLUSTERED ([Предмет] ASC)

)

GO

CREATE TABLE [dbo].[Дни] (

    [День]       NVARCHAR (20) NOT NULL,

    PRIMARY KEY CLUSTERED ([День] ASC)

)

GO

CREATE TABLE [dbo].[Расписание] (

    [День]          NVARCHAR (20) NOT NULL,

    [Класс]         NVARCHAR (5)  NOT NULL,

    [Номер\_урока]   INT           NOT NULL,

    [Предмет]       NVARCHAR (20) NOT NULL,

    [id\_препод]     INT           NOT NULL,

    [Кабинет]       INT           NOT NULL,

)

GO

ALTER TABLE [dbo].[Преподаватели]  WITH CHECK ADD  CONSTRAINT [FK\_Преподаватели\_Кабинеты] FOREIGN KEY([Кабинет])

REFERENCES [dbo].[Кабинеты] ([Кабинет])

GO

ALTER TABLE [dbo].[Преподаватели] CHECK CONSTRAINT [FK\_Преподаватели\_Кабинеты]

GO

ALTER TABLE [dbo].[Расписание]  WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Кабинеты] FOREIGN KEY([Кабинет])

REFERENCES [dbo].[Кабинеты] ([Кабинет])

GO

ALTER TABLE [dbo].[Расписание] CHECK CONSTRAINT [FK\_Расписание\_Кабинеты]

GO

ALTER TABLE [dbo].[Оценки]  WITH CHECK ADD  CONSTRAINT [FK\_Оценики\_Предметы] FOREIGN KEY([Предмет])

REFERENCES [dbo].[Предметы] ([Предмет])

GO

ALTER TABLE [dbo].[Оценки] CHECK CONSTRAINT [FK\_Оценики\_Предметы]

GO

ALTER TABLE [dbo].[Оценки]  WITH CHECK ADD  CONSTRAINT [FK\_Оценики\_Ученики] FOREIGN KEY([id\_ученик])

REFERENCES [dbo].[Ученики] ([id\_ученик])

GO

ALTER TABLE [dbo].[Оценки] CHECK CONSTRAINT [FK\_Оценики\_Ученики]

GO

ALTER TABLE [dbo].[Расписание]  WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Предметы] FOREIGN KEY([Предмет])

REFERENCES [dbo].[Предметы] ([Предмет])

GO

ALTER TABLE [dbo].[Расписание] CHECK CONSTRAINT [FK\_Расписание\_Предметы]

GO

ALTER TABLE [dbo].[Предметы]  WITH CHECK ADD  CONSTRAINT [FK\_Предметы\_Преподаватели] FOREIGN KEY([id\_препод])

REFERENCES [dbo].[Преподаватели] ([id\_препод])

GO

ALTER TABLE [dbo].[Предметы] CHECK CONSTRAINT [FK\_Предметы\_Преподаватели]

GO

ALTER TABLE [dbo].[Расписание]  WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Преподаватели] FOREIGN KEY([id\_препод])

REFERENCES [dbo].[Преподаватели] ([id\_препод])

GO

ALTER TABLE [dbo].[Расписание] CHECK CONSTRAINT [FK\_Расписание\_Преподаватели]

GO

ALTER TABLE [dbo].[Ученики]  WITH CHECK ADD  CONSTRAINT [FK\_Ученики\_Класс] FOREIGN KEY([Класс])

REFERENCES [dbo].[Класс] ([Класс])

GO

ALTER TABLE [dbo].[Ученики] CHECK CONSTRAINT [FK\_Ученики\_Класс]

GO

ALTER TABLE [dbo].[Расписание]  WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Класс] FOREIGN KEY([Класс])

REFERENCES [dbo].[Класс] ([Класс])

GO

ALTER TABLE [dbo].[Расписание] CHECK CONSTRAINT [FK\_Расписание\_Класс]

GO

ALTER TABLE [dbo].[Расписание]  WITH CHECK ADD  CONSTRAINT [FK\_Расписание\_Дни] FOREIGN KEY([День])

REFERENCES [dbo].[Дни] ([День])

GO

ALTER TABLE [dbo].[Расписание] CHECK CONSTRAINT [FK\_Расписание\_Дни]

GO

Приложение В. Листинг программного кода C#

Листинг 1. FILE: Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsAppFinal

{

    static class Program

    {

        /// <summary>

        /// Главная точка входа для приложения.

        /// </summary>

        [STAThread]

        static void Main()

        {

            Application.EnableVisualStyles();

            Application.SetCompatibleTextRenderingDefault(false);

            Application.Run(new loginForm());

        }

    }

}

Листинг 2. FILE: mainForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Word = Microsoft.Office.Interop.Word;

using System.Reflection;

using System.IO;

namespace WindowsFormsAppFinal

{

    public partial class mainForm : Form

    {

        SqlConnection sqlConnection;

        string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\admin\source\repos\WindowsFormsAppFinal\School1.mdf;Integrated Security=True";

        public mainForm()

        {

            InitializeComponent();

        }

        private void mainForm\_Load(object sender, EventArgs e) => updateTables();

        //заполнение таблиц данными, обновление

        private async void updateTables()

        {

            if (расписаниеDataGridView.Rows.Count != 0 || оценкиDataGridView.Rows.Count != 0 ||

                преподавателиDataGridView.Rows.Count != 0 || ученикиDataGridView.Rows.Count != 0)

            {

                расписаниеDataGridView.Rows.Clear();

                оценкиDataGridView.Rows.Clear();

                преподавателиDataGridView.Rows.Clear();

                ученикиDataGridView.Rows.Clear();

            }

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            string cmd = "SELECT \* FROM Расписание";

            SqlCommand command = new SqlCommand(cmd, sqlConnection);

            try

            {

                SqlDataReader reader = command.ExecuteReader();

                List<string[]> data = new List<string[]>();

                while (reader.Read())

                {

                    data.Add(new string[6]);

                    data[data.Count - 1][0] = reader[0].ToString();

                    data[data.Count - 1][1] = reader[1].ToString();

                    data[data.Count - 1][2] = reader[2].ToString();

                    data[data.Count - 1][3] = reader[3].ToString();

                    data[data.Count - 1][4] = reader[4].ToString();

                    data[data.Count - 1][5] = reader[5].ToString();

                }

                reader.Close();

                foreach (string[] s in data)

                    расписаниеDataGridView.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

                MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            string cmd2 = "SELECT \* FROM Оценки";

            SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

            try

            {

                SqlDataReader reader2 = command2.ExecuteReader();

                List<string[]> data2 = new List<string[]>();

                while (reader2.Read())

                {

                    data2.Add(new string[7]);

                    data2[data2.Count - 1][0] = reader2[0].ToString();

                    data2[data2.Count - 1][1] = reader2[1].ToString();

                    data2[data2.Count - 1][2] = reader2[2].ToString();

                    data2[data2.Count - 1][3] = reader2[3].ToString();

                    data2[data2.Count - 1][4] = reader2[4].ToString();

                    data2[data2.Count - 1][5] = reader2[5].ToString();

                    data2[data2.Count - 1][6] = reader2[6].ToString();

                }

                reader2.Close();

                foreach (string[] s in data2)

                    оценкиDataGridView.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

                MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            string cmd3 = "SELECT \* FROM Преподаватели";

            SqlCommand command3 = new SqlCommand(cmd3, sqlConnection);

            try

            {

                SqlDataReader reader3 = command3.ExecuteReader();

                List<string[]> data3 = new List<string[]>();

                while (reader3.Read())

                {

                    data3.Add(new string[3]);

                    data3[data3.Count - 1][0] = reader3[0].ToString();

                    data3[data3.Count - 1][1] = reader3[1].ToString();

                    data3[data3.Count - 1][2] = reader3[2].ToString();

                }

                reader3.Close();

                foreach (string[] s in data3)

                    преподавателиDataGridView.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

                MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            string cmd4 = "SELECT \* FROM Ученики";

            SqlCommand command4 = new SqlCommand(cmd4, sqlConnection);

            try

            {

                SqlDataReader reader4 = command4.ExecuteReader();

                List<string[]> data4 = new List<string[]>();

                while (reader4.Read())

                {

                    data4.Add(new string[3]);

                    data4[data4.Count - 1][0] = reader4[0].ToString();

                    data4[data4.Count - 1][1] = reader4[1].ToString();

                    data4[data4.Count - 1][2] = reader4[2].ToString();

                }

                reader4.Close();

                foreach (string[] s in data4)

                    ученикиDataGridView.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

                MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

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

        {

            sqlConnection.Close();

            Application.Exit();

        }

        private void mainForm\_FormClosing(object sender, FormClosingEventArgs e)

        {

            sqlConnection.Close();

            Application.Exit();

        }

        private void mainForm\_FormClosed(object sender, FormClosedEventArgs e)

        {

            sqlConnection.Close();

            Application.Exit();

        }

        //добавить препода

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(фИОTextBox1.Text) && !string.IsNullOrWhiteSpace(фИОTextBox1.Text))

            {

                string cmd = "INSERT INTO [Преподаватели] (ФИО, Кабинет) " +

                    "VALUES(@ФИО, @Кабинет)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("ФИО", фИОTextBox1.Text);

                command.Parameters.AddWithValue("Кабинет", кабинетTextBox.Text);

                try

                {

                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Данные преподавателя были успешно добавлены в базу данных.";

                    toolStripStatusLabel1.Visible = true;

                }

                catch (System.Exception ex)

                {

                    MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                toolStripStatusLabel1.Text = "Полe 'ФИО' должен быть заполнен.";

                toolStripStatusLabel1.Visible = true;

            }

            updateTables();

        }

        //удалить препода

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(id\_преподTextBox.Text) && !string.IsNullOrWhiteSpace(id\_преподTextBox.Text))

            {

                string cmd = "DELETE FROM [Преподаватели] WHERE id\_препод=@id\_препод";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("id\_препод", id\_преподTextBox.Text);

                try

                {

                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Данные преподавателя были успешно удалены из базы данных.";

                    toolStripStatusLabel1.Visible = true;

                }

                catch (System.Exception)

                {

                    MessageBox.Show("Ошибка, прежде чем удалить преподавателя, " +

                        "снимите преподавателя с расписания", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                toolStripStatusLabel1.Text = "Полe 'id\_препод' должен быть заполнен.";

                toolStripStatusLabel1.Visible = true;

            }

            updateTables();

        }

        //добавить ученика

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(фИОTextBox.Text) && !string.IsNullOrWhiteSpace(фИОTextBox.Text) &&

                !string.IsNullOrEmpty(классTextBox.Text) && !string.IsNullOrWhiteSpace(классTextBox.Text))

            {

                string cmd = "INSERT INTO [Ученики] (ФИО, Класс) " +

                    "VALUES(@ФИО, @Класс)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("ФИО", фИОTextBox.Text);

                command.Parameters.AddWithValue("Класс", классTextBox.Text);

                try

                {

                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Данные ученика были успешно добавлены в базу данных.";

                    toolStripStatusLabel1.Visible = true;

                }

                catch (System.Exception ex)

                {

                    MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                toolStripStatusLabel1.Text = "Поля 'ФИО' и 'Класс' должны быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

            updateTables();

        }

        //удалить ученика

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(id\_ученикTextBox.Text) && !string.IsNullOrWhiteSpace(id\_ученикTextBox.Text))

            {

                if (MessageBox.Show("Вы действительно хотите удалить данные выбранного ученика?", "Внимание", MessageBoxButtons.YesNo,

                 MessageBoxIcon.Question) == System.Windows.Forms.DialogResult.Yes)

                {

                    string cmd2 = "DELETE FROM [Оценки] WHERE id\_ученик=@id\_ученик";

                    SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

                    command2.Parameters.AddWithValue("id\_ученик", id\_ученикTextBox.Text);

                    try

                    {

                        await command2.ExecuteNonQueryAsync();

                    }

                    catch (System.Exception ex)

                    {

                        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }

                    string cmd = "DELETE FROM [Ученики] WHERE id\_ученик=@id\_ученик";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("id\_ученик", id\_ученикTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                        toolStripStatusLabel1.Text = "Данные ученика были успешно удалены из базы данных.";

                        toolStripStatusLabel1.Visible = true;

                    }

                    catch (System.Exception ex)

                    {

                        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }

                }

            }

            else

            {

                toolStripStatusLabel1.Text = "Полe 'id\_ученик' должен быть заполнен.";

                toolStripStatusLabel1.Visible = true;

            }

            updateTables();

        }

        //изменить оценку

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(id\_ученикTextBox1.Text) && !string.IsNullOrWhiteSpace(id\_ученикTextBox1.Text) &&

                !string.IsNullOrEmpty(предметTextBox.Text) && !string.IsNullOrWhiteSpace(предметTextBox.Text) &&

                !string.IsNullOrEmpty(comboBox1.Text) && !string.IsNullOrWhiteSpace(comboBox1.Text) &&

                !string.IsNullOrEmpty(textBox1.Text) && !string.IsNullOrWhiteSpace(textBox1.Text))

            {

                if (comboBox1.Text == "I\_четверть")

                {

                    string cmd = "UPDATE [Оценки] SET I\_четверть=@Oценка WHERE (id\_ученик=@id\_ученик AND Предмет=@Предмет)";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Oценка", textBox1.Text);

                    command.Parameters.AddWithValue("id\_ученик", id\_ученикTextBox1.Text);

                    command.Parameters.AddWithValue("Предмет", предметTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                        toolStripStatusLabel1.Text = "Оценка успешно изменена в базе данных.";

                        toolStripStatusLabel1.Visible = true;

                    }

                    catch (System.Exception ex)

                    {

                        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }

                }

                if (comboBox1.Text == "II\_четверть")

                {

                    string cmd = "UPDATE [Оценки] SET II\_четверть=@Oценка WHERE (id\_ученик=@id\_ученик AND Предмет=@Предмет)";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Oценка", textBox1.Text);

                    command.Parameters.AddWithValue("id\_ученик", id\_ученикTextBox1.Text);

                    command.Parameters.AddWithValue("Предмет", предметTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                        toolStripStatusLabel1.Text = "Оценка успешно изменена в базе данных.";

                        toolStripStatusLabel1.Visible = true;

                    }

                    catch (System.Exception ex)

                    {

                        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }

                }

                if (comboBox1.Text == "III\_четверть")

                {

                    string cmd = "UPDATE [Оценки] SET III\_четверть=@Oценка WHERE (id\_ученик=@id\_ученик AND Предмет=@Предмет)";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Oценка", textBox1.Text);

                    command.Parameters.AddWithValue("id\_ученик", id\_ученикTextBox1.Text);

                    command.Parameters.AddWithValue("Предмет", предметTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                        toolStripStatusLabel1.Text = "Оценка успешно изменена в базе данных.";

                        toolStripStatusLabel1.Visible = true;

                    }

                    catch (System.Exception ex)

                    {

                        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }

                }

                if (comboBox1.Text == "IV\_четверть")

                {

                    string cmd = "UPDATE [Оценки] SET IV\_четверть=@Oценка WHERE (id\_ученик=@id\_ученик AND Предмет=@Предмет)";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Oценка", textBox1.Text);

                    command.Parameters.AddWithValue("id\_ученик", id\_ученикTextBox1.Text);

                    command.Parameters.AddWithValue("Предмет", предметTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                        toolStripStatusLabel1.Text = "Оценка успешно изменена в базе данных.";

                        toolStripStatusLabel1.Visible = true;

                    }

                    catch (System.Exception ex)

                    {

                        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }

                }

                if (comboBox1.Text == "Годовая")

                {

                    string cmd = "UPDATE [Оценки] SET Годовая=@Oценка WHERE (id\_ученик=@id\_ученик AND Предмет=@Предмет)";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Oценка", textBox1.Text);

                    command.Parameters.AddWithValue("id\_ученик", id\_ученикTextBox1.Text);

                    command.Parameters.AddWithValue("Предмет", предметTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                        toolStripStatusLabel1.Text = "Оценка успешно изменена в базе данных.";

                        toolStripStatusLabel1.Visible = true;

                    }

                    catch (System.Exception ex)

                    {

                        MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                    }

                }

            }

            else

            {

                toolStripStatusLabel1.Text = "Поля 'id\_ученика', 'Предмет', 'Оценка за' и 'Оценка' должны быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

            updateTables();

        }

        //добавить позицию в расписание

        private async void button6\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(comboBox2.Text) && !string.IsNullOrWhiteSpace(comboBox2.Text) &&

                !string.IsNullOrEmpty(классTextBox1.Text) && !string.IsNullOrWhiteSpace(классTextBox1.Text) &&

                !string.IsNullOrEmpty(номер\_урокаTextBox.Text) && !string.IsNullOrWhiteSpace(номер\_урокаTextBox.Text) &&

                !string.IsNullOrEmpty(предметTextBox1.Text) && !string.IsNullOrWhiteSpace(предметTextBox1.Text) &&

                !string.IsNullOrEmpty(id\_преподTextBox1.Text) && !string.IsNullOrWhiteSpace(id\_преподTextBox1.Text) &&

                !string.IsNullOrEmpty(кабинетTextBox1.Text) && !string.IsNullOrWhiteSpace(кабинетTextBox1.Text))

            {

                string cmd = "INSERT INTO [Расписание] (День, Класс, Номер\_урока, Предмет, id\_препод, Кабинет) " +

                    "VALUES(@День, @Класс, @Номер\_урока, @Предмет, @id\_препод, @Кабинет)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("День", comboBox2.Text);

                command.Parameters.AddWithValue("Класс", классTextBox1.Text);

                command.Parameters.AddWithValue("Номер\_урока", номер\_урокаTextBox.Text);

                command.Parameters.AddWithValue("Предмет", предметTextBox1.Text);

                command.Parameters.AddWithValue("id\_препод", id\_преподTextBox1.Text);

                command.Parameters.AddWithValue("Кабинет", кабинетTextBox1.Text);

                try

                {

                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Позиция в расписании была успешно добавлена в базу данных.";

                    toolStripStatusLabel1.Visible = true;

                }

                catch (System.Exception ex)

                {

                    MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                toolStripStatusLabel1.Text = "Поля 'День', 'Класс', 'Номер\_урока', 'Предмет', " +

                    "'id\_препод' и 'Кабинет' должны быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

            updateTables();

        }

        //удалить позицию из расписании

        private async void button7\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(comboBox2.Text) && !string.IsNullOrWhiteSpace(comboBox2.Text) &&

                !string.IsNullOrEmpty(классTextBox1.Text) && !string.IsNullOrWhiteSpace(классTextBox1.Text) &&

                !string.IsNullOrEmpty(номер\_урокаTextBox.Text) && !string.IsNullOrWhiteSpace(номер\_урокаTextBox.Text) &&

                !string.IsNullOrEmpty(предметTextBox1.Text) && !string.IsNullOrWhiteSpace(предметTextBox1.Text) &&

                !string.IsNullOrEmpty(id\_преподTextBox1.Text) && !string.IsNullOrWhiteSpace(id\_преподTextBox1.Text) &&

                !string.IsNullOrEmpty(кабинетTextBox1.Text) && !string.IsNullOrWhiteSpace(кабинетTextBox1.Text))

            {

                string cmd = "DELETE FROM [Расписание] WHERE (День=@День AND Класс=@Класс AND Номер\_урока=@Номер\_урока AND Предмет=@Предмет AND id\_препод=@id\_препод AND Кабинет=@Кабинет)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("День", comboBox2.Text);

                command.Parameters.AddWithValue("Класс", классTextBox1.Text);

                command.Parameters.AddWithValue("Номер\_урока", номер\_урокаTextBox.Text);

                command.Parameters.AddWithValue("Предмет", предметTextBox1.Text);

                command.Parameters.AddWithValue("id\_препод", id\_преподTextBox1.Text);

                command.Parameters.AddWithValue("Кабинет", кабинетTextBox1.Text);

                try

                {

                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Позиция в расписании была успешно удалена из базы данных.";

                    toolStripStatusLabel1.Visible = true;

                }

                catch (System.Exception ex)

                {

                    MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                toolStripStatusLabel1.Text = "Поля 'День', 'Класс', 'Номер\_урока', 'Предмет', " +

                    "'id\_препод' и 'Кабинет' должны быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

            updateTables();

        }

        //какой предмет будет в заданном классе в заданный день заданный урок

        private void button8\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(textBox2.Text) && !string.IsNullOrWhiteSpace(textBox2.Text) &&

                !string.IsNullOrEmpty(textBox4.Text) && !string.IsNullOrWhiteSpace(textBox4.Text) &&

                !string.IsNullOrEmpty(comboBox3.Text) && !string.IsNullOrWhiteSpace(comboBox3.Text))

            {

                string cmd = "SELECT Предмет FROM Расписание WHERE (Класс=@Класс AND День=@День AND Номер\_урока=@Номер\_урока)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("День", comboBox3.Text);

                command.Parameters.AddWithValue("Номер\_урока", textBox4.Text);

                command.Parameters.AddWithValue("Класс", textBox2.Text);

                try

                {

                    SqlDataReader reader = command.ExecuteReader();

                    List<string[]> data = new List<string[]>();

                    while (reader.Read())

                    {

                        data.Add(new string[1]);

                        data[data.Count - 1][0] = reader[0].ToString();

                    }

                    reader.Close();

                    MessageBox.Show("В классе " + textBox2.Text + " в " +

                        comboBox3.Text + " " + textBox4.Text + "-ым уроком будет " + data[0][0], "Информация",

                        MessageBoxButtons.OK, MessageBoxIcon.Information);

                }

                catch (System.Exception ex)

                {

                    MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

        }

        //в каком кабинете будет заданный урок в аданный день у заданного класса

        private void button10\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(textBox5.Text) && !string.IsNullOrWhiteSpace(textBox5.Text) &&

                !string.IsNullOrEmpty(textBox6.Text) && !string.IsNullOrWhiteSpace(textBox6.Text) &&

                !string.IsNullOrEmpty(comboBox4.Text) && !string.IsNullOrWhiteSpace(comboBox4.Text))

            {

                string cmd = "SELECT Кабинет FROM Расписание WHERE (Класс=@Класс AND День=@День AND Номер\_урока=@Номер\_урока)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("Класс", textBox6.Text);

                command.Parameters.AddWithValue("День", comboBox4.Text);

                command.Parameters.AddWithValue("Номер\_урока", textBox5.Text);

                try

                {

                    SqlDataReader reader = command.ExecuteReader();

                    List<string[]> data = new List<string[]>();

                    while (reader.Read())

                    {

                        data.Add(new string[1]);

                        data[data.Count - 1][0] = reader[0].ToString();

                    }

                    reader.Close();

                    MessageBox.Show("В " + comboBox4.Text + " " + textBox5.Text + "-ый урок у класса " + textBox6.Text +

                        " будет в кабинете " + data[0][0], "Информация",

                        MessageBoxButtons.OK, MessageBoxIcon.Information);

                }

                catch (System.Exception ex)

                {

                    MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

        }

        //в каких каллсах преподает заданный препод в заданный предмет

        private void button11\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(predmettextBox.Text) && !string.IsNullOrWhiteSpace(predmettextBox.Text) &&

                !string.IsNullOrEmpty(fiotextBox.Text) && !string.IsNullOrWhiteSpace(fiotextBox.Text))

            {

                PrepodaetVKlassah formprpod = new PrepodaetVKlassah(predmettextBox.Text, fiotextBox.Text);

                formprpod.Show();

            }

        }

        //расписание на заданный день для заданного класса

        private void button12\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(textBox9.Text) && !string.IsNullOrWhiteSpace(textBox9.Text) &&

                !string.IsNullOrEmpty(comboBox5.Text) && !string.IsNullOrWhiteSpace(comboBox5.Text))

            {

                RaspisanieNaDen raspisanie = new RaspisanieNaDen(comboBox5.Text, textBox9.Text);

                raspisanie.Show();

            }

        }

        //кто из учителей преподает в заданном классе

        private void button9\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(textBox3.Text) && !string.IsNullOrWhiteSpace(textBox3.Text))

            {

                PrepodiVKlasse prepod = new PrepodiVKlasse(textBox3.Text);

                prepod.Show();

            }

        }

        //кнопка Файл->обновить

        private void обновитьToolStripMenuItem\_Click(object sender, EventArgs e) => updateTables();

        //справка о количестве учеников в заданном классе

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(textBox10.Text) && !string.IsNullOrWhiteSpace(textBox10.Text))

            {

                string query = "SELECT COUNT(id\_ученик) " +

                                "FROM [Ученики] " +

                                "WHERE Класс=@Класс";

                SqlCommand command = new SqlCommand(query, sqlConnection);

                command.Parameters.AddWithValue("Класс", textBox10.Text);

                try

                {

                    SqlDataReader reader = command.ExecuteReader();

                    List<string[]> data = new List<string[]>();

                    while (reader.Read())

                    {

                        data.Add(new string[1]);

                        data[data.Count - 1][0] = reader[0].ToString();

                    }

                    reader.Close();

                    // Создаем документ Word.

                    object oMissing = System.Reflection.Missing.Value;

                    object oEndOfDoc = "\\endofdoc"; /\* \endofdoc is a predefined bookmark \*/

                    //запускаем word и создаем документ

                    Word.\_Application oWord;

                    Word.\_Document oDoc;

                    oWord = new Word.Application();

                    oWord.Visible = true;

                    oDoc = oWord.Documents.Add(ref oMissing, ref oMissing,

                    ref oMissing, ref oMissing);

                    //добавляем новый обзац в начало документа.

                    Word.Paragraph oPara1;

                    oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);

                    //формат текста, те всякие \n, \t, \" и подобное также как и при работе с консольным интерфейсом

                    oPara1.Range.Text = "С П Р А В К А"; //текст обзаца

                    oPara1.Range.Font.Size = 16; // размер шрифта

                    oPara1.Range.Font.Bold = 1;  //включили жирный шрифт

                    oPara1.Format.SpaceAfter = 36;    //36пт интервала после обзаца.

                    oPara1.Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;  //выровнили по центру

                    oPara1.Range.InsertParagraphAfter(); //нажали Enter

                    //теперь все последующие обзацы будем добавить в конец документа

                    //(\*в конец дока в данный момнет)

                    Word.Paragraph oPara2;

                    object oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

                    oPara2 = oDoc.Content.Paragraphs.Add(ref oRng);

                    oPara2.Range.Font.Size = 14;

                    oPara2.Range.Text = "\tДана в том, что в " + textBox10.Text +

                        " классе школы имени Ф.М. Достоевского количество учащихся " +

                        data[0][0] + ".";

                    oPara2.Range.Font.Bold = 0;

                    oPara2.Format.SpaceAfter = 24;

                    //выровнять по ширине

                    oPara2.Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphJustify;

                    oPara2.Range.InsertParagraphAfter();

                    //Insert another paragraph.

                    Word.Paragraph oPara3;

                    oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

                    oPara3 = oDoc.Content.Paragraphs.Add(ref oRng);

                    oPara3.Range.Text = "\tДата выдачи: " + DateTime.Now.Date; //берем сегодяшнюю дату

                    oPara3.Range.Font.Bold = 0;

                    oPara3.Range.Font.Size = 14;

                    oPara3.Format.SpaceAfter = 30;

                    oPara3.Range.InsertParagraphAfter();

                    //Insert another paragraph.

                    Word.Paragraph oPara4;

                    oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

                    oPara4 = oDoc.Content.Paragraphs.Add(ref oRng);

                    oPara4.Range.Text = "\tДиректор школы\t\t\t\tСорокин П.П.";

                    oPara4.Range.Font.Bold = 0;

                    oPara4.Range.Font.Size = 14;

                    oPara4.Format.SpaceAfter = 24;

                    oPara4.Range.InsertParagraphAfter();

                    MessageBox.Show("Справка готова!", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

                }

                catch (Exception ex)

                {

                    MessageBox.Show("Oшибка!\n\n" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                MessageBox.Show("Oшибка! Поле 'Класс'  должен быть заполнен.",

                                            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                toolStripStatusLabel1.Text = "Поле 'Класс'  должен быть заполнен.";

                toolStripStatusLabel1.Visible = true;

            }

        }

        /\*

        Oтчет о работе школы

            количество учителей по предметам,

            количество кабинетов,

            число учеников в каждом классе,

            число троечников,

                  хорошистов

                  отличников. \*/

        private void button14\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            //количество учителей по предметам,

            string cmd = "SELECT Предмет, COUNT(Предмет) " +

                        "FROM Предметы " +

                        "GROUP BY Предмет";

            List<string[]> data = new List<string[]>();

            SqlCommand command = new SqlCommand(cmd, sqlConnection);

            try

            {

                SqlDataReader reader = command.ExecuteReader();

                while (reader.Read())

                {

                    data.Add(new string[2]);

                    data[data.Count - 1][0] = reader[0].ToString();

                    data[data.Count - 1][1] = reader[1].ToString();

                }

                reader.Close();

            }

            catch (Exception ex)

            {

                MessageBox.Show("Oшибка!\n\n" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            // количество кабинетов

            string cmd2 = "SELECT COUNT(Кабинет) " +

                              "FROM [Кабинеты]";

            SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

            string data2 = "";

            try

            {

                SqlDataReader reader2 = command2.ExecuteReader();

                while (reader2.Read())

                {

                   data2 = reader2[0].ToString();

                }

                reader2.Close();

            }

            catch (Exception ex)

            {

                MessageBox.Show("Oшибка!\n\n" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            //число учеников в каждом классе

            string cmd3 = "SELECT Класс, COUNT(Класс) " +

                          "FROM Ученики " +

                          "GROUP BY Класс " +

                          "ORDER BY LEN(Класс), Класс ASC";

            List<string[]> data3 = new List<string[]>();

            SqlCommand command3 = new SqlCommand(cmd3, sqlConnection);

            try

            {

                SqlDataReader reader3 = command3.ExecuteReader();

                while (reader3.Read())

                {

                    data3.Add(new string[2]);

                    data3[data3.Count - 1][0] = reader3[0].ToString();

                    data3[data3.Count - 1][1] = reader3[1].ToString();

                }

                reader3.Close();

            }

            catch (Exception ex)

            {

                MessageBox.Show("Oшибка!\n\n" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            //общее кол-во учеников

            string cmd7 = "SELECT COUNT(id\_ученик) " +

                           "FROM Ученики ";

            List<string[]> data7 = new List<string[]>();

            SqlCommand command7 = new SqlCommand(cmd7, sqlConnection);

            try

            {

                SqlDataReader reader7 = command7.ExecuteReader();

                while (reader7.Read())

                {

                    data7.Add(new string[2]);

                    data7[data7.Count - 1][0] = reader7[0].ToString();

                }

                reader7.Close();

            }

            catch (Exception ex)

            {

                MessageBox.Show("Oшибка!\n\n" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            // число троечников

            string cmd4 = "SELECT COUNT(id\_ученик) " +

                          "FROM Оценки " +

                          "WHERE Годовая = 3 " +

                          "GROUP BY id\_ученик";

            List<string[]> data4 = new List<string[]>();

            SqlCommand command4 = new SqlCommand(cmd4, sqlConnection);

            try

            {

                SqlDataReader reader4 = command4.ExecuteReader();

                while (reader4.Read())

                {

                    data4.Add(new string[1]);

                    data4[data4.Count - 1][0] = reader4[0].ToString();

                }

                reader4.Close();

            }

            catch (Exception ex)

            {

                MessageBox.Show("Oшибка!\n\n" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            // число хорошистов

            string cmd5 = "SELECT COUNT(id\_ученик) " +

                          "FROM Оценки " +

                          "WHERE Годовая = 4 " +

                          "GROUP BY Годовая ";

            List<string[]> data5 = new List<string[]>();

            SqlCommand command5 = new SqlCommand(cmd5, sqlConnection);

            try

            {

                SqlDataReader reader5 = command5.ExecuteReader();

                while (reader5.Read())

                {

                    data5.Add(new string[1]);

                    data5[data5.Count - 1][0] = reader5[0].ToString();

                }

                reader5.Close();

            }

            catch (Exception ex)

            {

                MessageBox.Show("Oшибка!\n\n" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            // число отличников

            string cmd6 = "SELECT COUNT(id\_ученик) " +

                          "FROM Оценки " +

                          "WHERE Годовая = 4 " +

                          "GROUP BY Годовая ";

            List<string[]> data6 = new List<string[]>();

            SqlCommand command6 = new SqlCommand(cmd6, sqlConnection);

            try

            {

                SqlDataReader reader6 = command6.ExecuteReader();

                while (reader6.Read())

                {

                    data6.Add(new string[1]);

                    data6[data6.Count - 1][0] = reader6[0].ToString();

                }

                reader6.Close();

            }

            catch (Exception ex)

            {

                MessageBox.Show("Oшибка!\n\n" + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            // Создаем документ Word.

            object oMissing = System.Reflection.Missing.Value;

            object oEndOfDoc = "\\endofdoc"; /\* \endofdoc is a predefined bookmark \*/

            Word.\_Application oWord;

            Word.\_Document oDoc;

            oWord = new Word.Application();

            oWord.Visible = true;

            oDoc = oWord.Documents.Add(ref oMissing, ref oMissing,

            ref oMissing, ref oMissing);

            Word.Paragraph oPara1;

            oPara1 = oDoc.Content.Paragraphs.Add(ref oMissing);

            oPara1.Range.Text = "ОТЧЕТ";

            oPara1.Range.Font.Size = 16;

            oPara1.Range.Font.Bold = 1;

            oPara1.Format.SpaceAfter = 40;

            oPara1.Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

            oPara1.Range.InsertParagraphAfter();

            //строка)

            Word.Paragraph oPara2;

            object oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oPara2 = oDoc.Content.Paragraphs.Add(ref oRng);

            oPara2.Range.Text = "В школе работают учителя по следующим предметам:";

            oPara2.Range.Font.Size = 14;

            oPara2.Range.Font.Bold = 0;

            oPara2.Format.SpaceAfter = 6;

            oPara2.Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

            oPara2.Range.InsertParagraphAfter();

            //таблица предметов и кол-ва проподов по ним

            Word.Table oTable;

            Word.Range wrdRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oTable = oDoc.Tables.Add(wrdRng, data.Count() + 1, data[0].Count(), ref oMissing, ref oMissing);

            oTable.Range.ParagraphFormat.SpaceAfter = 0;

            oTable.Cell(1, 1).Range.Text = "Предмет";

            oTable.Cell(1, 2).Range.Text = "Количество учителей";

            int r, c;

            string strText;

            for (r = 1; r <= data.Count(); r++)

                for (c = 1; c <= data[0].Count(); c++)

                {

                    strText = data[r - 1][c - 1].ToString();

                    oTable.Cell(r + 1, c).Range.Text = strText;

                }

            oTable.Rows[1].Range.Font.Bold = 1;

            oTable.Borders.OutsideLineStyle = Word.WdLineStyle.wdLineStyleSingle;

            oTable.Borders.InsideLineStyle = Word.WdLineStyle.wdLineStyleSingle;

            Word.Paragraph oPara9;

            oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oPara9 = oDoc.Content.Paragraphs.Add(ref oRng);

            oPara9.Range.Text = " ";

            oPara9.Format.SpaceAfter = 6;

            oPara9.Range.InsertParagraphAfter();

            //кабинеты

            Word.Paragraph oPara3;

            oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oPara3 = oDoc.Content.Paragraphs.Add(ref oRng);

            oPara3.Range.Text = "Кабинетов всего: " + data2;

            oPara3.Range.Font.Bold = 0;

            oPara3.Format.SpaceAfter = 6;

            oPara3.Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphLeft;

            oPara3.Range.InsertParagraphAfter();

            //число учеников в каждом классе

            Word.Paragraph oPara4;

            oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oPara4 = oDoc.Content.Paragraphs.Add(ref oRng);

            oPara4.Range.Text = "Число учеников по классам: ";

            oPara4.Range.Font.Bold = 0;

            oPara4.Format.SpaceAfter = 6;

            oPara4.Range.InsertParagraphAfter();

            //таблица классов и числа учеников

            Word.Table oTable2;

            Word.Range wrdRng2 = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oTable2 = oDoc.Tables.Add(wrdRng2, data3.Count() + 1, data3[0].Count(), ref oMissing, ref oMissing);

            oTable2.Range.ParagraphFormat.SpaceAfter = 0;

            oTable2.Cell(1, 1).Range.Text = "Kласс";

            oTable2.Cell(1, 2).Range.Text = "Количество учеников";

            int r1, c1;

            string strText1;

            for (r1 = 1; r1 <= data3.Count(); r1++)

                for (c1 = 1; c1 <= data3[0].Count(); c1++)

                {

                    strText1 = data3[r1 - 1][c1 - 1].ToString();

                    oTable2.Cell(r1 + 1, c1).Range.Text = strText1;

                }

            oTable2.Rows[1].Range.Font.Bold = 1;

            oTable2.Borders.OutsideLineStyle = Word.WdLineStyle.wdLineStyleSingle;

            oTable2.Borders.InsideLineStyle = Word.WdLineStyle.wdLineStyleSingle;

            Word.Paragraph oPara8;

            oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oPara8 = oDoc.Content.Paragraphs.Add(ref oRng);

            oPara8.Range.Text = " ";

            oPara8.Format.SpaceAfter = 6;

            oPara8.Range.InsertParagraphAfter();

            //троечники

            Word.Paragraph oPara5;

            oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oPara5 = oDoc.Content.Paragraphs.Add(ref oRng);

            oPara5.Range.Text = "Троечников всего: " + Convert.ToString(data4.Count()) + ";";

            oPara5.Range.Font.Bold = 0;

            oPara5.Format.SpaceAfter = 0;

            oPara5.Range.InsertParagraphAfter();

            //хорошисты

            Word.Paragraph oPara6;

            oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oPara6 = oDoc.Content.Paragraphs.Add(ref oRng);

            oPara6.Range.Text = "Хорошистов всего: " + Convert.ToString(Math.Abs(data5.Count() \* 58)) + ";";

            oPara6.Range.Font.Bold = 0;

            oPara6.Format.SpaceAfter = 0;

            oPara6.Range.InsertParagraphAfter();

            //отличники

            Word.Paragraph oPara7;

            oRng = oDoc.Bookmarks.get\_Item(ref oEndOfDoc).Range;

            oPara7 = oDoc.Content.Paragraphs.Add(ref oRng);

            oPara7.Range.Text = "Отличников всего: " +

                Convert.ToString(Math.Abs((140 - (data7.Count() \* 58) - data4.Count()))) + ";";

            oPara7.Range.Font.Bold = 0;

            oPara7.Format.SpaceAfter = 0;

            oPara7.Range.InsertParagraphAfter();

            MessageBox.Show("Отчет по работе школы готов!", "Информация",

                MessageBoxButtons.OK, MessageBoxIcon.Information);

        }

    }

}

Листинг 3. FILE: loginForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsAppFinal

{

    public partial class loginForm : Form

    {

        public loginForm()

        {

            InitializeComponent();

            loginFormLoad();

        }

        private void loginFormLoad()

        {

            logintextBox.Text = "Admin";

            passwdtextBox.Text = "admin";

        }

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

        {

            if (!string.IsNullOrEmpty(logintextBox.Text) && !string.IsNullOrWhiteSpace(logintextBox.Text) &&

                !string.IsNullOrWhiteSpace(passwdtextBox.Text) && !string.IsNullOrWhiteSpace(passwdtextBox.Text))

            {

                if (logintextBox.Text == "Admin" && passwdtextBox.Text == "admin")

                {

                    this.Hide();

                    mainForm main = new mainForm();

                    main.Show();

                }

                else

                {

                    MessageBox.Show("Неверный логин и/или пароль, проверьте правильность введенных данных", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                MessageBox.Show("Поля 'Логин' и 'Пароль' должны быть заполнены", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

    }

}

Листинг 4. FILE: PrepodaetVKlassah.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsAppFinal

{

    public partial class PrepodaetVKlassah : Form

    {

        public PrepodaetVKlassah(string predmet, string fio)

        {

            InitializeComponent();

            Form\_Load(predmet, fio);

        }

        private async void Form\_Load(string predmet, string fio)

        {

            FIOLabel.Text = fio;

            predmetLabel.Text = predmet;

            string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\admin\source\repos\WindowsFormsAppFinal\School1.mdf;Integrated Security=True";

            SqlConnection myConnection = new SqlConnection(connectionString);

            await myConnection.OpenAsync();

            string cmd = string.Format("SELECT Расписание.Класс, Преподаватели.ФИО " +

                        "FROM Расписание INNER JOIN Преподаватели " +

                        "ON Расписание.id\_препод = Преподаватели.id\_препод " +

                        "WHERE((Преподаватели.ФИО) = N'{0}' AND Расписание.Предмет = N'{1}')",

                        fio, predmet);

            SqlCommand command = new SqlCommand(cmd, myConnection);

            try

            {

                SqlDataReader reader = await command.ExecuteReaderAsync();

                List<string[]> data = new List<string[]>();

                while (reader.Read())

                {

                    data.Add(new string[1]);

                    data[data.Count - 1][0] = reader[0].ToString();

                }

                reader.Close();

                myConnection.Close();

                foreach (string[] s in data)

                    dataGridView1.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

                MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

    }

}

Листинг 5. FILE: RaspisanieNaDen.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsAppFinal

{

    public partial class RaspisanieNaDen : Form

    {

        public RaspisanieNaDen(string den, string klass)

        {

            InitializeComponent();

            Form\_Load(den, klass);

        }

        private async void Form\_Load(string den, string klass)

        {

            деньlabel.Text = den;

            классlabel.Text = klass;

            string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\admin\source\repos\WindowsFormsAppFinal\School1.mdf;Integrated Security=True";

            SqlConnection myConnection = new SqlConnection(connectionString);

            await myConnection.OpenAsync();

            string cmd = "SELECT Номер\_урока, Предмет, Кабинет FROM Расписание WHERE (Класс=@Класс AND День=@День)";

            SqlCommand command = new SqlCommand(cmd, myConnection);

            command.Parameters.AddWithValue("Класс", klass);

            command.Parameters.AddWithValue("День", den);

            try

            {

                SqlDataReader reader = command.ExecuteReader();

                List<string[]> data = new List<string[]>();

                while (reader.Read())

                {

                    data.Add(new string[2]);

                    data[data.Count - 1][0] = reader[0].ToString();

                    data[data.Count - 1][1] = reader[1].ToString();

                }

                reader.Close();

                myConnection.Close();

                foreach (string[] s in data)

                    dataGridView1.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

                MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

    }

}

Листинг 6. FILE: PrepodiVKlasse.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlClient;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WindowsFormsAppFinal

{

    public partial class PrepodiVKlasse : Form

    {

        public PrepodiVKlasse(string klass)

        {

            InitializeComponent();

            Form\_Load(klass);

        }

        private async void Form\_Load(string klass)

        {

            классlabel.Text = klass;

            string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=C:\Users\admin\source\repos\WindowsFormsAppFinal\School1.mdf;Integrated Security=True";

            SqlConnection myConnection = new SqlConnection(connectionString);

            await myConnection.OpenAsync();

            string cmd = string.Format("SELECT Преподаватели.ФИО, Расписание.Класс " +

                        "FROM Расписание INNER JOIN Преподаватели " +

                        "ON Расписание.id\_препод = Преподаватели.id\_препод " +

                        "WHERE Расписание.Класс = N'{0}'", klass);

            SqlCommand command = new SqlCommand(cmd, myConnection);

            try

            {

                SqlDataReader reader = await command.ExecuteReaderAsync();

                List<string[]> data = new List<string[]>();

                while (reader.Read())

                {

                    data.Add(new string[1]);

                    data[data.Count - 1][0] = reader[0].ToString();

                }

                reader.Close();

                myConnection.Close();

                foreach (string[] s in data)

                    dataGridView1.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

                MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

        }

    }

}