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 |  | Диденко В. |
| Преподаватель |  | Татарникова Т.М. |

Санкт-Петербург

2020

ЗАДАНИЕ

**НА КУРСОВУЮ РАБОТУ**

|  |  |  |
| --- | --- | --- |
| Студент Диденко В. | | |
| Группа 8363 | | |
| Тема работы: Разработка и реализация базы данных | | |
| Исходные данные:  Спроектировать базу данных, построить программу, обеспечивающую взаимодействие с ней в режиме диалога, для обслуживания работы конференции. База данных должна содержать справочник персоналий участников конференции (фамилия, имя, отчество, ученая степень, ученое звание, научное направление, место работы, кафедра (отдел), должность, страна, город, почтовый индекс, адрес, рабочий телефон, домашний телефон, e-mail), и информацию, связанную с участием в конференции (докладчик или участник, дата рассылки 1-го приглашения, дата поступления заявки, тема доклада, отметка о поступлении тезисов, дата рассылки 2-го приглашения, дата поступления оргвзноса, размер поступившего оргвзноса, дата приезда, дата отъезда, потребность в гостинице).  Помимо SQL запросов для создания таблиц базы данных, разработать пакет, состоящий из процедур и функций, позволяющий:   * для указанной даты 1-ой рассылки вывести список приглашенных и посчитать их количество; * предоставить возможность добавления приглашенных на конференцию с указанием оргвзноса и даты его уплаты; * вывести список приглашенных, с указанием даты об уплате оргвзноса; * для указанной интервала дат, вывести список участников, уплативших оргвзнос в этом диапазоне; * для указанного города вывести название тезисов докладов, поступивших из этого города; * для указанного города, вывести список нуждающихся в гостинице.   Необходимо предусмотреть возможность выдачи приглашения всем участникам с указанием в нем необходимой информации. | | |
| 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 50 страниц. | | |
| Дата выдачи задания: 14.09.2020 | | |
| Дата сдачи курсовой работы: 21.12.2020 | | |
| Дата защиты курсовой работы: 21.12.2020 | | |
| Студент |  | Диденко В. |
| Преподаватель |  | Татарникова Т.М. |

Аннотация

Программа «Конференции» предназначена для обслуживания работы конференции и обеспечивает в режиме диалога доступ к базе данных, справочнику персоналий участников конференции и информации, связанную с участием в конференции. Предусмотрена возможность как внесения изменений, так и получения приглашений с выводом информации на документ MS Word. Также программа по взаимодействию с БД позволяет выводить сведения для обслуживания работы конференции, 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 "Conferences" program is designed to service the work of the conference and provides in the dialogue mode access to the database, the directory of the conference participants' personalities and information related to participation in the conference. The possibility of both making changes and receiving invitations with the output to the MS Word document is provided. Also, the program for interacting with the database allows you to display information for servicing the conference, 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](#_Toc59252923)

[1. Анализ предметной области 8](#_Toc59252924)

[1.1 Подробное описание объектов предметной области 8](#_Toc59252925)

[1.2 Определение сущностей (объектов) 9](#_Toc59252926)

[1.3 Информационные объекты (сущности) 9](#_Toc59252927)

[1.4 Схема данных 9](#_Toc59252928)

[1.5 Формулировка задачи 10](#_Toc59252929)

[1.6 Ограничение на информацию БД 11](#_Toc59252930)

[1.7 Краткое описание алгоритмов решения задач 11](#_Toc59252931)

[1.8 Определение групп пользователей 12](#_Toc59252932)

[1.9 Описание выходных документов, которые должны генерироваться в системе 13](#_Toc59252933)

[1.10 Описание входных документов для заполнения БД 13](#_Toc59252934)

[2. Обоснование модели данных 14](#_Toc59252935)

[3. Проектирование БД 16](#_Toc59252938)

[3.1 Нормализация, схема базы данных 16](#_Toc59252939)

[3.2 Физическое проектирование базы данных 17](#_Toc59252940)

[4. Обоснование выбора СУБД 22](#_Toc59252941)

[4.1 Тип модели данных и её адекватность потребностям рассматриваемой предметной области. 22](#_Toc59252945)

[4.2 Характеристики производительности и набор функциональных возможностей 22](#_Toc59252946)

[4.3 Стоимость СУБД и дополнительного ПО 23](#_Toc59252947)

[4.4 Удобство и надежность СУБД в эксплуатации 23](#_Toc59252948)

[5. Описание функций управления данными 24](#_Toc59252949)

[5.1 Хранение (создание информационных объектов) 24](#_Toc59252951)

[5.2 Манипулирование (добавление, изменение, удаление, поиск данных) 27](#_Toc59252952)

[5.3 Доступ к данным (назначение прав доступа) 28](#_Toc59252953)

[5.4 Предоставление запрашиваемых данных пользователю (генерация справок, отчетов, итогов) 28](#_Toc59252954)

[6. Организация защиты БД 30](#_Toc59252955)

[6.1 Описание ограничений целостности для каждого информационного объекта. 30](#_Toc59252961)

[6.2 Рекомендуемые средства физической защиты (виды резервного копирования и периодичность проведения резервного копирования) 31](#_Toc59252962)

[6.3 Применение протоколов шифрования для защиты базы данных 32](#_Toc59252963)

[Заключение 33](#_Toc59252964)

[Список использованных источников 35](#_Toc59252965)

[Приложение А. Руководство ползователя БД 36](#_Toc59252966)

[Приложение Б. Листинг программного кода SQL 45](#_Toc59252979)

[Листинг 1. Создание таблиц и установка зависимостей (внешних ключей) 45](#_Toc59252980)

[Приложение В. Листинг программного кода C# 48](#_Toc59252981)

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 справочник персоналий участников конференции и информацию, связанную с участием в конференции.

Для каждого участника конференции в БД должны храниться: фамилия, имя, отчество, ученая степень, ученое звание, научное направление, место работы, кафедра, должность, страна, город, почтовый индекс, адрес, рабочий телефон, домашний телефон, e-mail, докладчик или участник, дата рассылки 1-го приглашения, дата поступления заявки, тема доклада, отметка о поступлении тезисов, дата рассылки 2-го приглашения, дата поступления оргвзноса, размер поступившего оргвзноса, дата приезда, дата отъезда, потребность в гостинице.

Будет разработан пакет, состоящий из процедур и функций, позволяющий:

* для указанной даты 1-ой рассылки вывести список приглашенных и посчитать их количество;
* предоставить возможность добавления приглашенных на конференцию с указанием оргвзноса и даты его уплаты;
* вывести список приглашенных, с указанием даты об уплате оргвзноса;
* для указанной интервала дат, вывести список участников, уплативших оргвзнос в этом диапазоне;
* для указанного города вывести название тезисов докладов, поступивших из этого города;
* для указанного города, вывести список нуждающихся в гостинице.

Необходимо предусмотреть возможность выдачи приглашения всем участникам с указанием в нем необходимой информации.

* 1. Определение сущностей (объектов)

После анализа предметной области выделим следующие сущности:

* Автор;
* Доклад;
* Конференция;
* Участник;
* Гостиница.
  1. Информационные объекты (сущности)

Сущности имеют следующие атрибуты:

Автор: id\_автор, фамилия, имя, отчество, адрес, страна, город, почтовый индекс, рабочий телефон, домашний телефон, e-mail, ученая степень, ученое звание, научное направление, кафедра, место работы, должность.

Конференция: id\_конференция, название, организатор, статус, дата начала, оргвзнос.

Гостиница: id\_гостиница, название, адрес, город, страна, стоимость номера.

Участник: id\_участник, id\_автор, id\_конференция, id\_гостиница, дата поступления оргвзноса, размер оргвзноса, дата поступления заявки, дата рассылки 1-ого приглашения, дата рассылки 2-ого приглашения, дата приезда, дата отъезда, докладчик или участник, потребность в гостинице.

Доклад: тема доклада, отметка о поступлении тезисов, количество страниц, соавторы.

* 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).

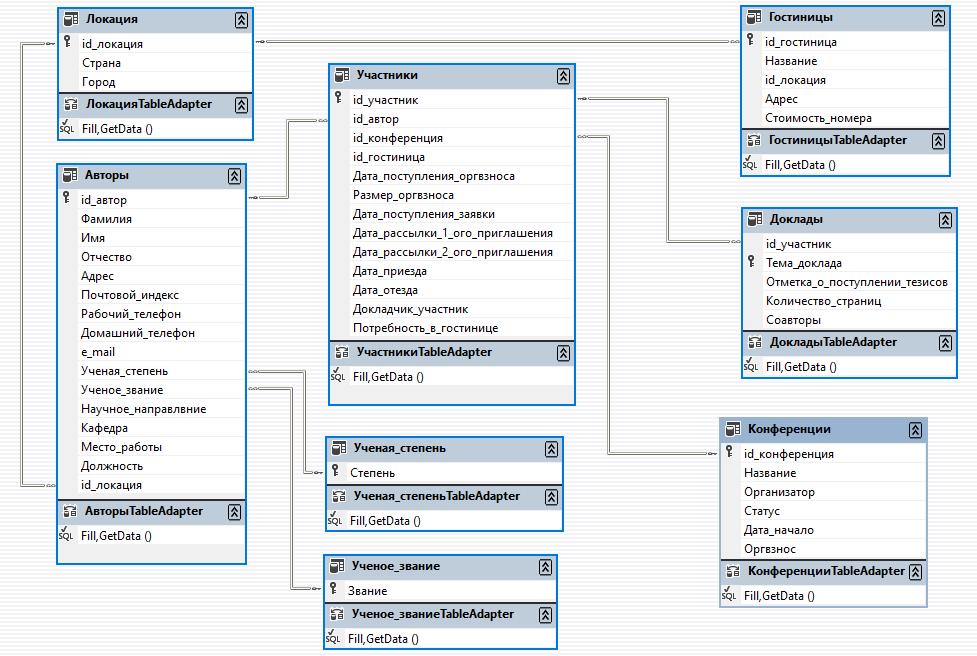


Рисунок 1 – Инфологическая модель проектированной БД «Конференции»

* 1. Формулировка задачи

Содержать справочник персоналий участников конференции и информацию, связанную с участием в конференции.

Выдачи приглашения всем участникам с указанием в нем необходимой информации.

* 1. Ограничение на информацию БД
* Каждый автор может быть зарегистрирован как участник только в одной конференции.
* Поля ученая степень и ученое звание могут иметь только определенные значения.
* Дата рассылки 1-ого приглашения не может опережать дату рассылки 2-ого приглашения.
* Дата приезда участника не может опережать дату отъезда участника.
* Размер оргвзноса участника не может быть меньше, чем требуемый оргвзнос для данной конференции.
  1. Краткое описание алгоритмов решения задач

1. Хранение информации персоналий участников конференции и информацию, связанную с участием в конференции.

Информация об авторах, участниках, их докладов, конференции и гостиниц хранятся в 8 связанных (по внешним ключам) таблицах, связи организованы таким образом, чтобы обеспечить корректную работу системы и целостность базы данных. Например, чтобы удалить данные конференции, надо сначала удалить всех участников этой конференции, а также их доклады. Такое ограничение не позволяет появления ситуации, когда секретарь случайно удалил конференцию, а в итоге – также информацию о всех участниках и их докладах.

1. Выдача приглашения всем участникам с указанием в нем необходимой информации.

Для выдачи приглашений используется запрос к базе данных, берутся данные участников для указанной конференции, а с полученными данными создается MS Word документ и заполняется (подробное описание приведено в 5-ом разделе).

* 1. Определение групп пользователей

Пользователь будет один – организатор конференции.

Организатору конференции доступны следующие сведения:

* информация об авторах;
* информация об участниках;
* информация о докладах участников;
* для указанной даты 1-ой рассылки вывести список приглашенных и посчитать их количество;
* предоставить возможность добавления приглашенных на конференцию с указанием оргвзноса и даты его уплаты;
* вывести список приглашенных, с указанием даты об уплате оргвзноса;
* для указанной интервала дат, вывести список участников, уплативших оргвзнос в этом диапазоне;
* для указанного города вывести название тезисов докладов, поступивших из этого города;
* для указанного города, вывести список нуждающихся в гостинице.

Организатор конференции может вносить следующие изменения:

* вносить информацию о новом конференции;
* удалить информацию о конференции;
* изменить информацию о конференции;
* вносить информацию о новом авторе;
* удалить информацию об авторе;
* изменить информацию об авторе;
* вносить информацию о новом участнике;
* удалить информацию об участнике;
* изменить информацию об участнике;
* вносить информацию о докладе участника;
* удалить информацию о докладе участника;
* изменить информацию о докладе участника;
* вносить информацию о гостинице;
* удалить информацию о гостинице;
* изменить информацию о гостинице.
  1. Описание выходных документов, которые должны генерироваться в системе
* приглашение для всех участников выбранной конференции, с указанием необходимой информации.
  1. Описание входных документов для заполнения БД
* анкета автора, с указанием всей нужной информации;
* данные конференции;
* справка с гостиниц, с указанием адреса, названия и стоимости номера в гостинице.

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.

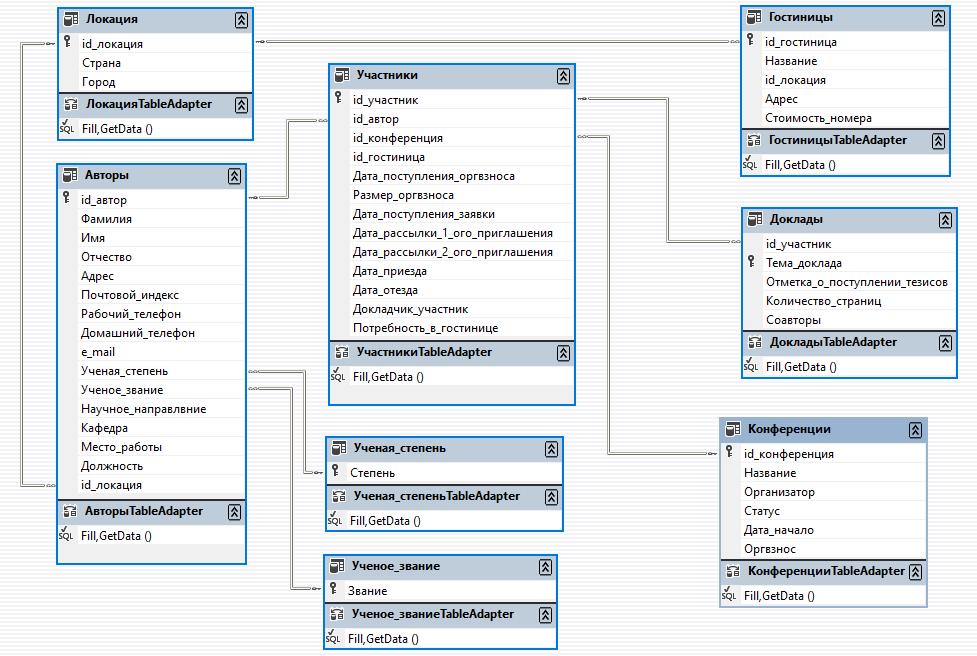


Рисунок 2. Реляционная модель данных.

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.

* 1. Физическое проектирование базы данных

В спроектированной базе данных школы созданы следующие таблицы:

* Авторы
* Конференции
* Доклады
* Гостиницы
* Участники
* Локация
* Ученая степень
* Ученое звание

В таблице Авторы хранится список авторов. Таблица состоит из следующих полей: id\_автор (автор, ключевое, счетчик), фамилия (текстовое), имя (текстовое), отчество (текстовое), адрес (текстовое), почтовый индекс (целое), рабочий телефон (текстовое), домашний телефон (текстовое), e-mail (текстовое), ученая степень ученое звание (текстовое), научное направление (текстовое), кафедра (текстовое), место работы (текстовое), должность (текстовое), id\_локация (целое, источник строк - таблица Локация, числовое)

*Таблица 1. Авторы*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_автор | INT | PRIMARY KEY, IDENTITY (1, 1),  NOT NULL |
| Фамилия | NVARCHAR (20) | NOT NULL |
| Имя | NVARCHAR (20) | NOT NULL |
| Отчество | NVARCHAR (20) | NOT NULL |
| Адрес | NVARCHAR (20) | NOT NULL |
| Почтовый\_индекс | NVARCHAR (20) | NOT NULL |
| Рабочий\_телефон | NVARCHAR (20) | NOT NULL |
| Домашний\_телефон | NVARCHAR (20) | NOT NULL |
| e\_mail | VARCHAR (30) | NOT NULL |
| Ученая\_степень | NVARCHAR (25) | FOREIGN KEY,  NOT NULL |
| Ученое\_звание | NVARCHAR (25) | FOREIGN KEY,  NOT NULL |
| Научное\_направление | NVARCHAR (50) | NOT NULL |
| Кафедра | NVARCHAR (30) | NOT NULL |
| Место\_работы | NVARCHAR (30) | NOT NULL |
| Должность | NVARCHAR (30) | NOT NULL |
| id\_локация | INT | FOREIGN KEY,  NOT NULL |

Таблица Конференции содержит список конференций. Таблица состоит из следующих полей: id\_конференция (ключевое, счетчик), название (текстовое), организатор (текстовое), статус (текстовое), дата начала (текстовое), оргвзнос (целое).

*Таблица 2. Конференции*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_конференция | INT | PRIMARY KEY, IDENTITY (1, 1),  NOT NULL |
| Название | NVARCHAR (50) | NOT NULL |
| Организатор | NVARCHAR (50) | NOT NULL |
| Статус | NVARCHAR (50) | NOT NULL |
| Дата\_начало | DATE | NOT NULL |
| Оргвзнос | INT | NOT NULL |

Таблица Доклады состоит из следующих полей: id\_участник (ключевое, счетчик), тема доклада (текстовое), отметка о поступлении тезисов (булево) количество страниц (целое), Соавторы (текстовое).

*Таблица 3. Доклады*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_участник | INT | FOREIGN KEY,  NOT NULL |
| Тема\_доклада | NVARCHAR (200) | NOT NULL |
| Отметка\_о\_поступлении\_тезисов | BIT | NOT NULL |
| Количество\_страниц | INT | NOT NULL |
| Соавторы | NVARCHAR (50) | NULL |

В таблице Гостиницы хранится список номеров гостиниц. Таблица состоит из следующих полей: id\_гостиница (ключевое, счетчик), название (текстовое), id\_локация (источник строк - таблица Локации, числовое), адрес (текстовое), стоимость номера (целое).

*Таблица 4. Гостиницы*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_гостиница | INT | PRIMARY KEY, IDENTITY (1, 1),  NOT NULL |
| Название | NVARCHAR (25) | NOT NULL |
| id\_локация | INT | FOREIGN KEY,  NOT NULL |
| Адрес | NVARCHAR (50) | NOT NULL |
| Стоимость\_номера | INT | NOT NULL |

В таблице Участники хранится информация об авторах, связанная с участием в конференции. Таблица состоит из следующих полей: id\_участник (числовое, ключевое), id\_автор (числовое, источник строк - таблица Авторы), id\_конференция (числовое, источник строк - таблица Конференции), id\_гостиница (числовое, источник строк - таблица Гостиницы), дата поступления оргвзноса (дата), размер оргвзноса (числовое), дата поступления заявки (дата), дата рассылки 1-ого приглашения (дата), дата рассылки 2-ого приглашения (дата), дата приезда (дата), дата отъезда (дата), Докладчик/участник (булево), потребность в гостинице (булево).

*Таблица 5. Участники*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_участник | INT | PRIMARY KEY,  IDENTITY (1, 1)  NOT NULL |
| id\_автор | INT | FOREIGN KEY,  NULL |
| id\_конференция | INT | FOREIGN KEY,  NOT NULL |
| id\_гостиница | INT | FOREIGN KEY,  NULL |
| Дата\_поступления\_оргвзноса | DATE | NOT NULL |
| Размер\_оргвзноса | INT | NOT NULL |
| Дата\_поступления\_заявки | DATE | NOT NULL |
| Дата\_рассылки\_1\_ого\_приглашения | DATE | NOT NULL |
| Дата\_рассылки\_2\_ого\_приглашения | DATE | NOT NULL |
| Дата\_приезда | DATE | NOT NULL |
| Дата\_отезда | DATE | NOT NULL |
| Докладчик\_участник | BIT | NOT NULL |
| Потребность\_в\_гостинице | BIT | NOT NULL |

В таблице Локация хранится список пар страна-город. Таблица состоит из следующих полей: id\_локация (числовое, ключевое), страна (текстовое), город (текстовое).

*Таблица 6. Локация*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| id\_локация | INT | PRIMARY KEY,  IDENTITY (1, 1)  NOT NULL |
| Страна | NVARCHAR (25) | NOT NULL |
| Город | NVARCHAR (25) | NOT NULL |

В таблице Ученая степень хранится всевозможные ученые степени. Таблица состоит из одного поля: Степень (текстовое поле, ключевое).

*Таблица 7. Ученая степень*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| Степень | NVARCHAR (25) | PRIMARY KEY,  NOT NULL |

В таблице Ученое звание хранится всевозможные ученые звания. Таблица состоит из одного поля: Звание (текстовое поле, ключевое).

*Таблица 7. Ученая степень*

|  |  |  |
| --- | --- | --- |
| **Название столбца** | **Тип данных** | **Атрибуты** |
| Звание | NVARCHAR (25) | PRIMARY KEY,  NOT NULL |

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

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

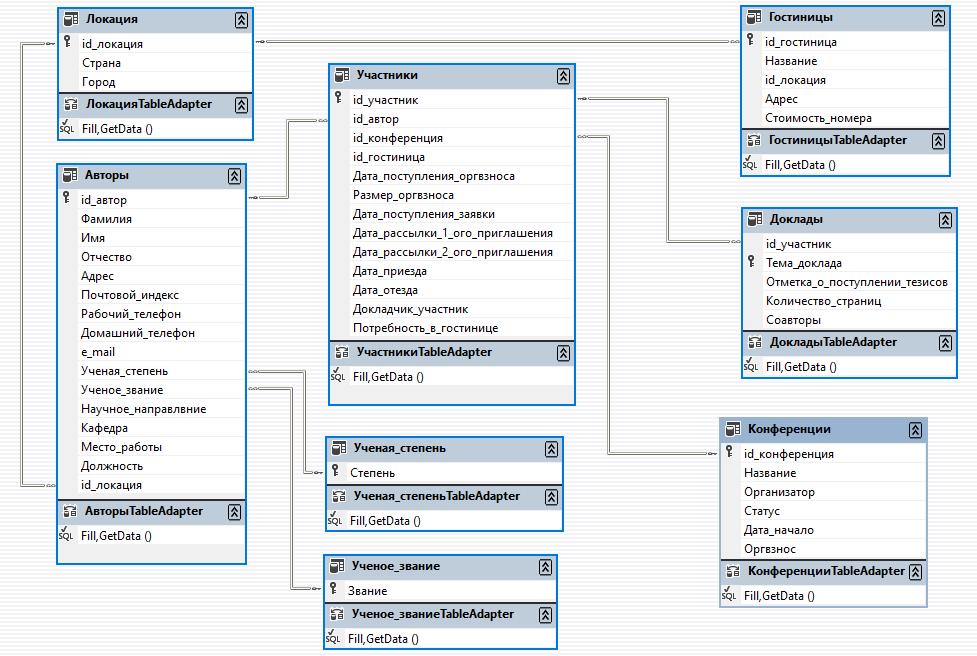


Рисунок 3. Схема реляционной модели БД «Конференции».

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].[Локация] (

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

    [Страна]     NVARCHAR (25) NOT NULL,

    [Город]      NVARCHAR (25) NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_локация] ASC)

)

GO

CREATE TABLE [dbo].[Ученая\_степень] (

    [Степень] NVARCHAR (25) NOT NULL,

    PRIMARY KEY CLUSTERED ([Степень] ASC)

)

GO

CREATE TABLE [dbo].[Ученое\_звание] (

    [Звание] NVARCHAR (25) NOT NULL,

    PRIMARY KEY CLUSTERED ([Звание] ASC)

)

GO

CREATE TABLE [dbo].[Авторы] (

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

    [Фамилия]             NVARCHAR (20) NOT NULL,

    [Имя]                 NVARCHAR (20) NOT NULL,

    [Отчество]            NVARCHAR (20) NOT NULL,

    [Адрес]               NVARCHAR (50) NOT NULL,

    [Почтовой\_индекс]     NVARCHAR (10) NOT NULL,

    [Рабочий\_телефон]     NVARCHAR (20) NOT NULL,

    [Домашний\_телефон]    NVARCHAR (20) NOT NULL,

    [e\_mail]              VARCHAR (30)  NOT NULL,

    [Ученая\_степень]      NVARCHAR (25) NOT NULL,

    [Ученое\_звание]       NVARCHAR (25) NOT NULL,

    [Научное\_направлвние] NVARCHAR (50) NOT NULL,

    [Кафедра]             NVARCHAR (30) NOT NULL,

    [Место\_работы]        NVARCHAR (30) NOT NULL,

    [Должность]           NVARCHAR (50) NOT NULL,

    [id\_локация]          INT           NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_автор] ASC),

    CONSTRAINT [FK\_Авторы\_Локация]

FOREIGN KEY ([id\_локация])

REFERENCES [dbo].[Локация] ([id\_локация]),

    CONSTRAINT [FK\_Авторы\_Ученая\_степень]

FOREIGN KEY ([Ученая\_степень])

REFERENCES [dbo].[Ученая\_степень] ([Степень]),

    CONSTRAINT [FK\_Авторы\_Ученое\_звание]

FOREIGN KEY ([Ученое\_звание])

REFERENCES [dbo].[Ученое\_звание] ([Звание])

)

GO

CREATE TABLE [dbo].[Конференции] (

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

    [Название]       NVARCHAR (50) NOT NULL,

    [Организатор]    NVARCHAR (50) NOT NULL,

    [Статус]         NVARCHAR (50) NOT NULL,

    [Дата\_начало]    DATE          NOT NULL,

    [Оргвзнос]       INT           NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_конференция] ASC)

)

GO

CREATE TABLE [dbo].[Гостиницы] (

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

    [Название]         NVARCHAR (25) NOT NULL,

    [id\_локация]       INT           NOT NULL,

    [Адрес]            NVARCHAR (50) NOT NULL,

    [Стоимость\_номера] INT           NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_гостиница] ASC),

    CONSTRAINT [FK\_Гостиницы\_Локация]

FOREIGN KEY ([id\_локация])

REFERENCES [dbo].[Локация] ([id\_локация])

)

GO

CREATE TABLE [dbo].[Участники] (

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

    [id\_автор]                        INT  NULL,

    [id\_конференция]                  INT  NOT NULL,

    [id\_гостиница]                    INT  NULL,

    [Дата\_поступления\_оргвзноса]      DATE NOT NULL,

    [Размер\_оргвзноса]                INT  NOT NULL,

    [Дата\_поступления\_заявки]         DATE NOT NULL,

    [Дата\_рассылки\_1\_ого\_приглашения] DATE NOT NULL,

    [Дата\_рассылки\_2\_ого\_приглашения] DATE NOT NULL,

    [Дата\_приезда]                    DATE NOT NULL,

    [Дата\_отезда]                     DATE NOT NULL,

    [Докладчик\_участник]              BIT  NOT NULL,

    [Потребность\_в\_гостинице]         BIT  NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_участник] ASC),

    UNIQUE NONCLUSTERED ([id\_автор] ASC, [id\_конференция] ASC),

    CONSTRAINT [FK\_Участники\_Авторы]

FOREIGN KEY ([id\_автор])

REFERENCES [dbo].[Авторы] ([id\_автор]),

    CONSTRAINT [FK\_Участники\_Конференции]

FOREIGN KEY ([id\_конференция])

REFERENCES [dbo].[Конференции] ([id\_конференция])

)

GO

CREATE TABLE [dbo].[Доклады] (

    [id\_участник]                   INT            NOT NULL,

    [Тема\_доклада]                  NVARCHAR (200) NOT NULL,

    [Отметка\_о\_поступлении\_тезисов] BIT            NOT NULL,

    [Количество\_страниц]            INT            NOT NULL,

    [Соавторы]                      NVARCHAR (50)  NULL,

    PRIMARY KEY CLUSTERED ([Тема\_доклада] ASC),

    CONSTRAINT [FK\_Доклады\_Участники]

FOREIGN KEY ([id\_участник])

REFERENCES [dbo].[Участники] ([id\_участник])

)

GO

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

Для добавления данных из интерфейса программы используются параметризированное запросы. Например, для добавления конференции, данные заносятся в базу данных следующим запросом:

INSERT INTO Конференции(Название, Организатор, Статус, Дата\_начала, Оргвзнос)

VALUES(@Название, @Организатор, @Статус, @Дата\_начало, @Оргвзнос)

С помощью встроенных средств языка C# вместо параметров @Название, @Организатор, @Статус, @Дата\_начало, @Оргвзнос 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 адекватно реагирует 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.

* 1. Доступ к данным (назначение прав доступа)

Программа предназначена для организатора конференций, организатор имеет все права на все операции по добавлению, обновлению, удалению и поиску данных.

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

В программе все справки и отчеты можно получить во вкладке «Справки отчеты запросы» (см. рис. 4).

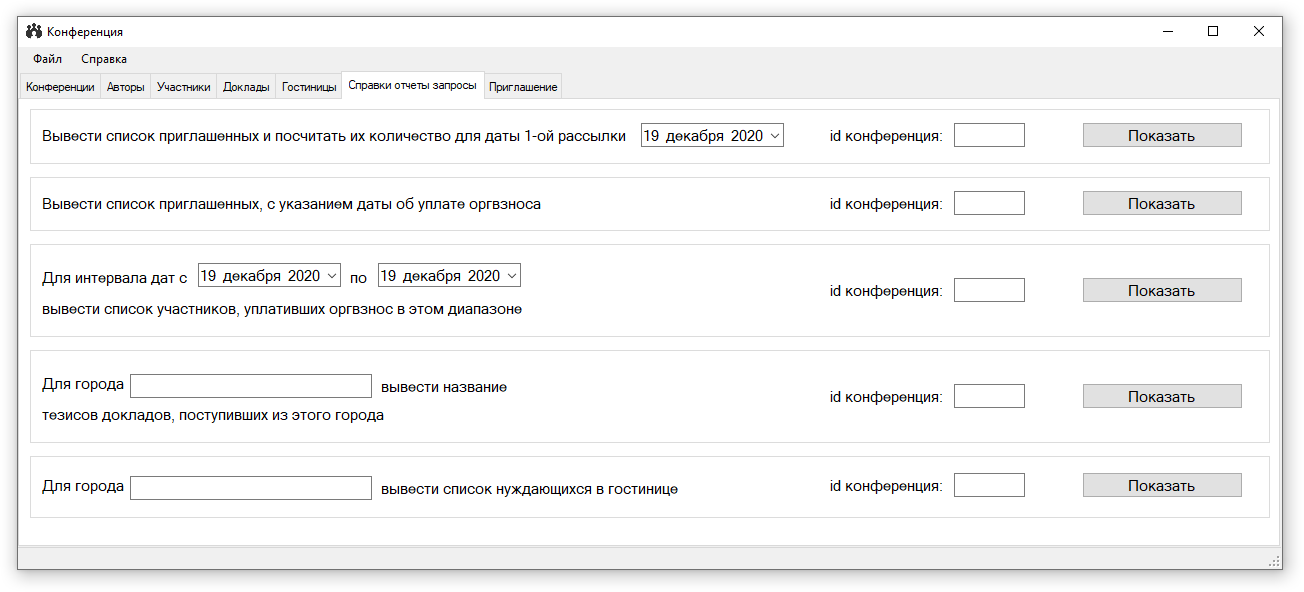


Рисунок 4 – Вкладка «Справки отчеты запросы»

Во вкладке «Приглашение» можно генерировать приглашение сразу для всех участников выбранной конференции (см. рис. 5).

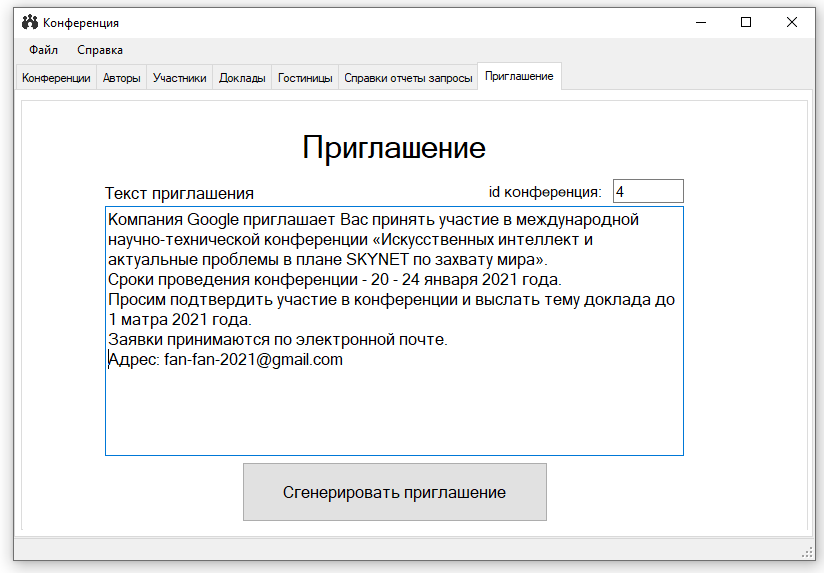


Рисунок 5 – Вкладка «Приглашение»

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.

Аналогичными средствами строится остальная часть документа.

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 участника в базу в конференцию, 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.
* Ограничение CHECK ограничивает данные, которые пользователь может ввести в определенную колонку указанными значениями.
  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.

* 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 обслуживания работы конференции. В БД хранится справочник персоналий участников конференции и информацию, связанную с участием в конференции.

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 справочника персоналий участников конференции и информации, связанную с участием в конференции;
* выдача приглашения всем участникам с указанием в нем необходимой информации;
* предусмотрена возможность корректировки БД (добавление, удаление, изменение данных о конференции, об авторах, участниках и других информационных объектах);
* разработана реляционная СУБД, содержащая элементы автоматизации и обработки данных.

Разработан пакет, состоящий из процедур и функций, позволяющий:

* для указанной даты 1-ой рассылки вывести список приглашенных и посчитать их количество;
* предоставить возможность добавления приглашенных на конференцию с указанием оргвзноса и даты его уплаты;
* вывести список приглашенных, с указанием даты об уплате оргвзноса;
* для указанной интервала дат, вывести список участников, уплативших оргвзнос в этом диапазоне;
* для указанного города вывести название тезисов докладов, поступивших из этого города;
* для указанного города, вывести список нуждающихся в гостинице.

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.
4. Техническая документация по SQL Server. // Техническая документация Майкрософт для разработчиков. URL: https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15 (дата обращения: 04.12.2020).
5. Документация по .NET. // Техническая документация Майкрософт для разработчиков. URL: https://docs.microsoft.com/ru-ru/dotnet/ (дата обращения: 04.12.2020).
6. Документация по C# // Техническая документация Майкрософт для разработчиков. URL: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (дата обращения: 04.12.2020).

Приложение А. Руководство ползователя БД

1. Краткое описание программы

Программа «Конференции» предназначена для обслуживания работы конференции, с обеспечением взаимодействия с базой данных в режиме диалога. В БД хранится справочник персоналий участников конференции и информацию, связанную с участием в конференции.

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 «Conference.zip». Необходимо распаковать архив и запустить установочный файл «setup.exe» (см. рис. 6).

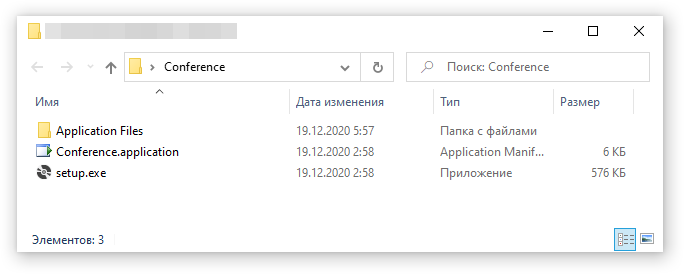


Рисунок 6– файл «setup.exe»

После двойного нажатия левой кнопки мыши происходит запуск установочного файла (см. рис. 7).

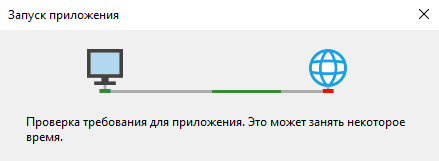


Рисунок 7 – запуск файла «setup.exe»

На экран выводится диалоговое окно «Установка приложения – Предупреждение о безопасности». Для подтверждения установки нажмите кнопку «Установить» (см. рис. 8).

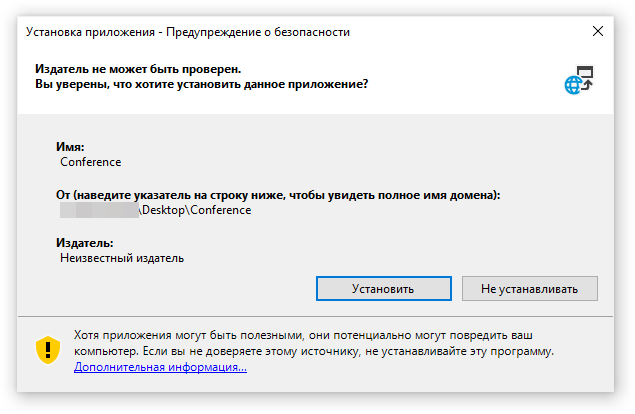


Рисунок 8 – Диалоговое окно «Установка приложения – Предупреждение о безопасности»

Начнется установка приложения, дождитесь его окончания.

После окончании процесса установки программа запуститься и выведет на экран основное окно (см. рис. 9). Кроме того, программа добавится к списку приложений и его можно будет найти в меню Пуск.

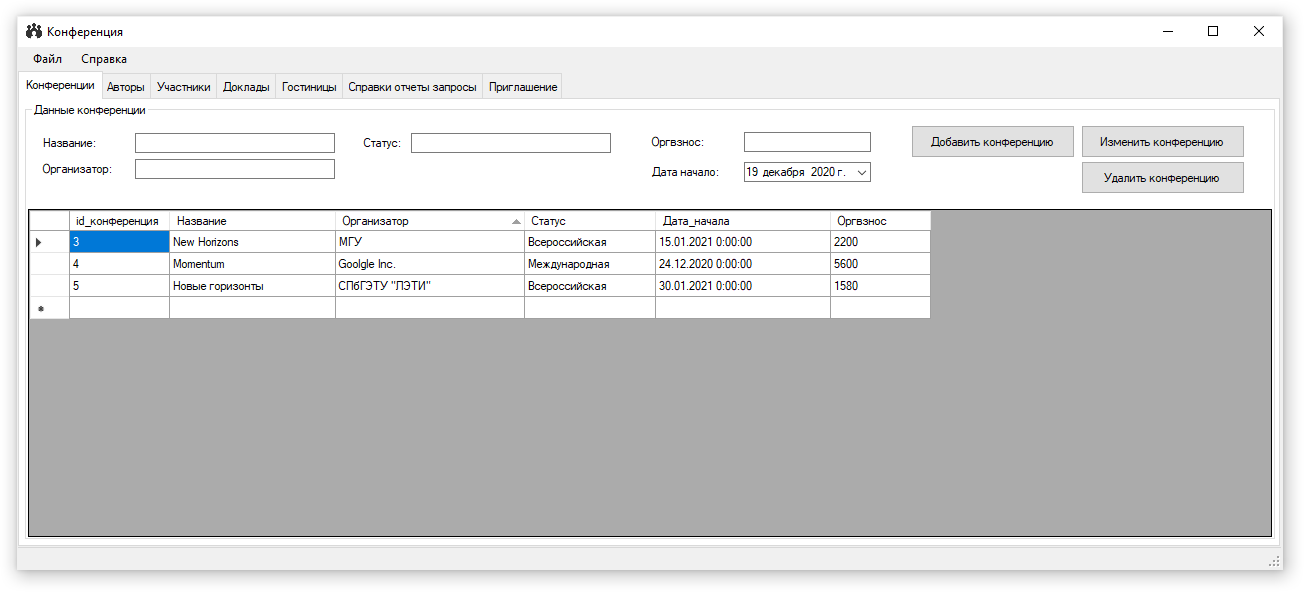


Рисунок 9 – Основное окно приложения «Конференции»

1. Работа с программой

При двойном нажатии на иконку программы в меню Пуск откроется основное окно программы (см. рис. 9).

Главное окно программы «Конференции» имеет 7 вкладок: «Конференции», «Авторы», «Участники», «Доклады», «Гостиницы», «Справки отчеты запросы» и «Приглашение». Подробно рассмотрим функционал каждой из них.

### 4.1 Вкладка «Конференции»

Вкладка «Конференции» предназначена для работы с данными конференций: добавления, изменения и удаления данных о конференции (см. рис. 9)

Для добавления данных о конференции в базу, нужно просто заполнить соответствующие поля и нажать кнопку «Добавить конференцию».

Для изменения данных конференции нужно нажать на кнопку «Изменить конференцию», после чего откроется диалоговое окно (см. рис. 10), где сначала нужно ввести id\_конференции, а после заполнить те поля, которые требуется изменить, а также поставить галочки перед заполненными полями.

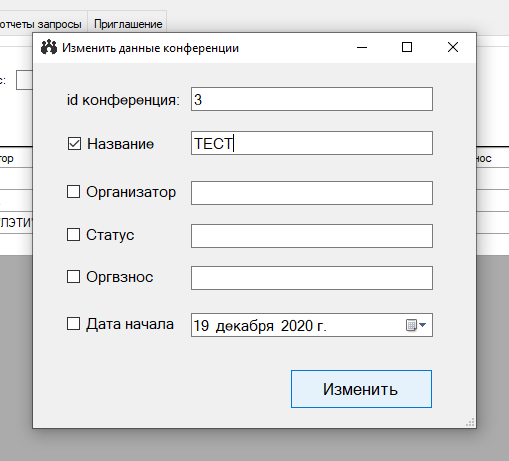


Рисунок 10 – Диалоговое окно изменения конференции

Для удаления данных конференции с базы нужно нажать на кнопку «Удалить конференцию». Откроется диалоговое окно, где нужно будет ввести id\_конференции и нажать кнопку удалить (см. рис. 11).

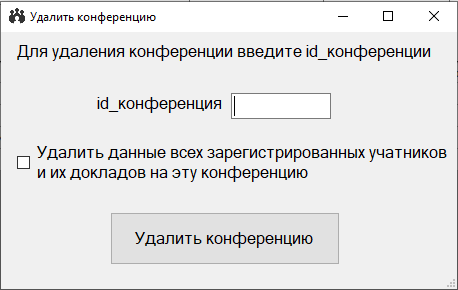


Рисунок 11 – Диалоговое окно удаления конференции

### 4.2 Вкладка «Авторы»

Вкладка «Авторы» предназначена для работы с данными авторов: добавления нового, изменения и удаления зарегистрированного автора (см. рис. 12).

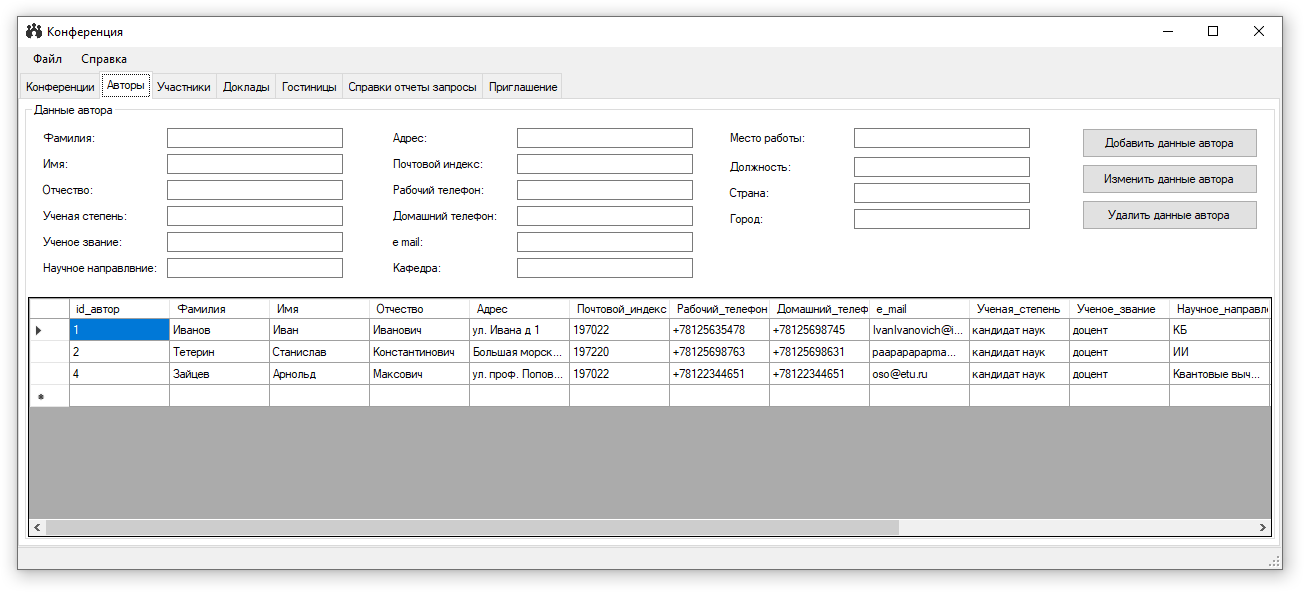


Рисунок 12 – Вкладка «Авторы»

Процесс добавления, изменения и удаления данных авторов аналогичен процессу добавления и удаления данных конференций.

### 4.3 Вкладка «Участники»

Вкладка «Участники» предназначена для работы с данными участников конференций: добавления нового, изменения и удаления зарегистрированного участника (см. рис. 13). Процесс аналогичен процессу добавления и удаления данных конференций.

Галочки перед пунктами «Докладчик» и «Гостиница» определяют является участник докладчиком или просто участником и есть ли потребность в гостинице соответственно.

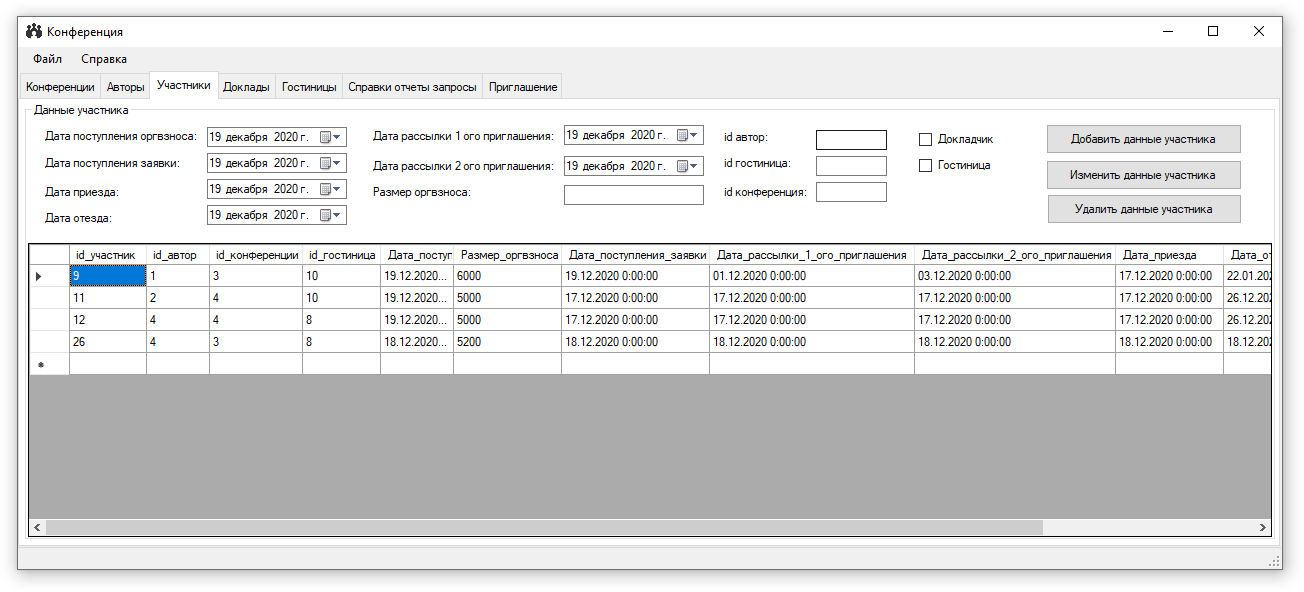


Рисунок 13 – Вкладка «Участники»

### 4.4 Вкладка «Доклады»

Вкладка «Расписание» предназначена для регистрации докладов участников конференции в базу данных. (см. рис. 14).

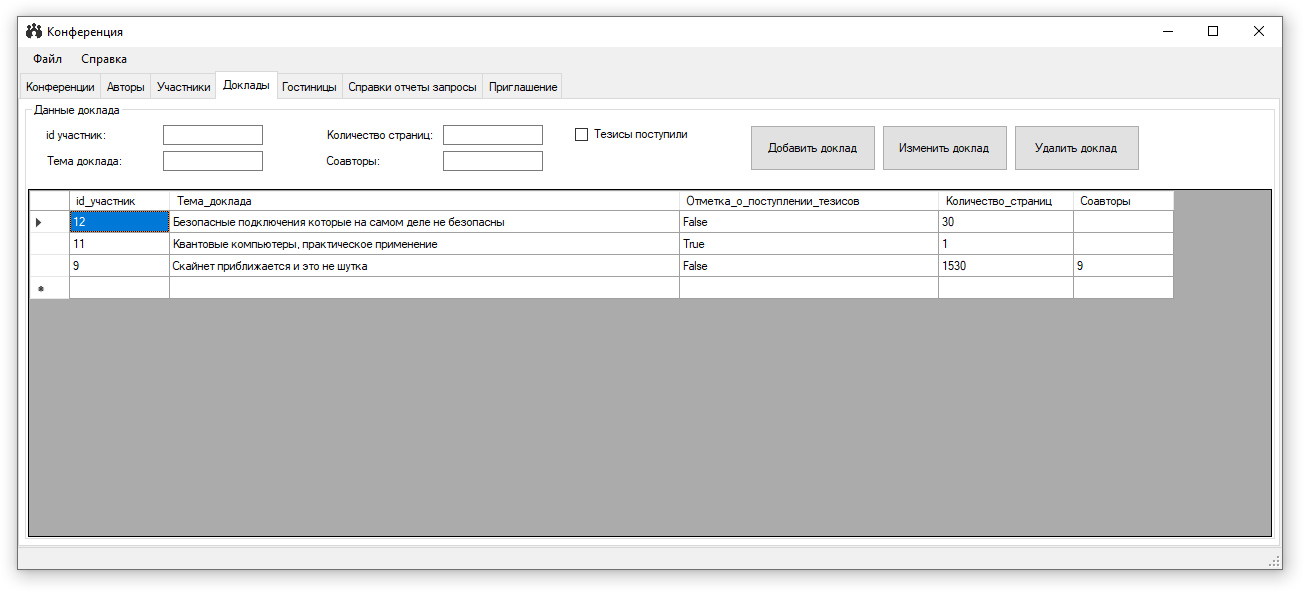


Рисунок 14 – Вкладка «Доклады»

### 4.5 Вкладка «Гостиницы»

Вкладка «Гостиницы» предназначена для работы с данными о гостиничных номерах (см. рис. 15).

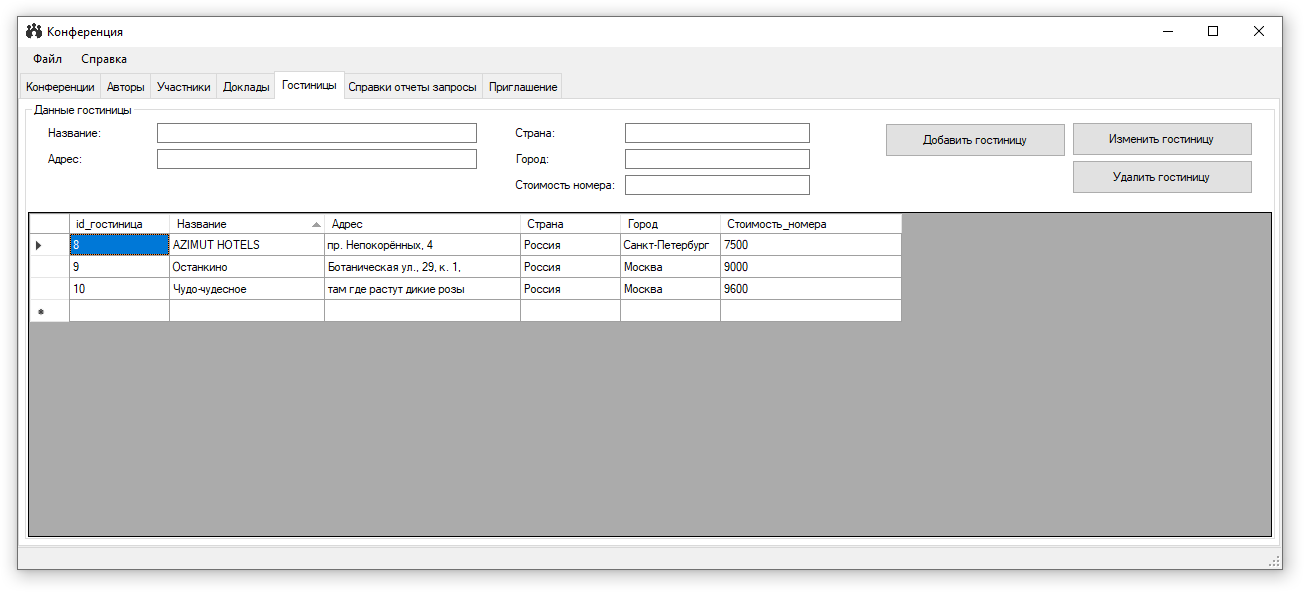


Рисунок 15 – Вкладка «Гостиницы»

### 4.6 Вкладка «Справки отчеты запросы»

Вкладка «Справки отчеты запросы» предназначена для получения справочной информации (см. рис. 16).

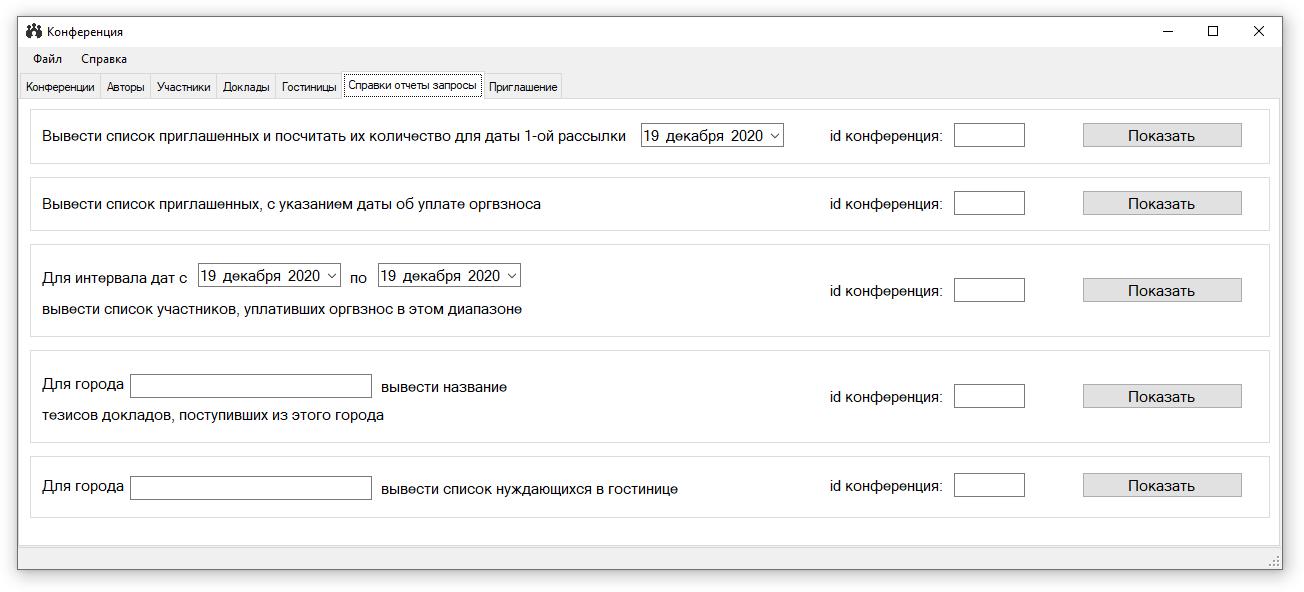


Рисунок 16 – Вкладка «Справки отчеты запросы»

Результаты запросов выводятся в диалоговых окнах в виде таблиц. Например, 3-ий запрос: «Для интервала дат с …» с параметрами дат с 16-12-2020 до 31-12-2020 и id\_конференции равной 4. Результат запроса будет следующим (см. рис. 17):

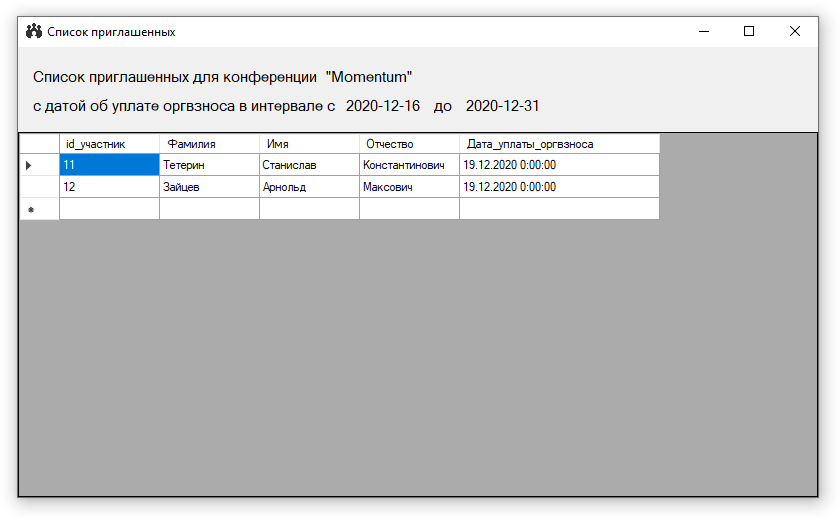


Рисунок 17 – Результат запроса

### 4.7 Вкладка «Приглашение»

Вкладка «Приглашение» предназначена для генерации приглашений на указанную конференцию по списку зарегистрированных участников (см. рис. 18).

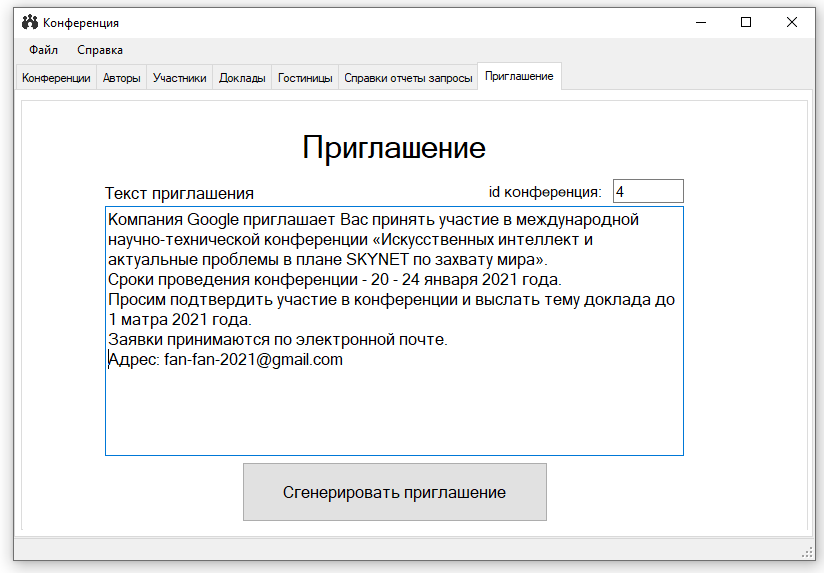


Рисунок 18 – Вкладка «Приглашение»

Для генерации приглашений необходимо заполнить поле id\_конференция, а также текст приглашения и нажать кнопку «Сгенерировать приглашение».

### 4.8 Контекстное меню

Контекстное меню «Файл» содержит следующие элементы (см. рис. 19):

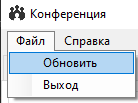


Рисунок 19 – Контекстное меню «Файл»

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

Контекстное меню «Справка» содержит следующие элементы (см. рис. 20):

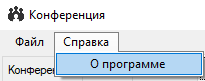


Рисунок 20 – Контекстное меню «Справка»

* О программе – выводит на экран форму «О программе» (см. рис. 21).

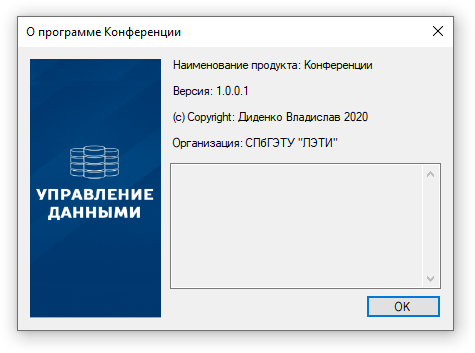


Рисунок 21 – Окно «О программе Конференции»

Приложение Б. Листинг программного кода SQL

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

CREATE TABLE [dbo].[Локация] (

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

    [Страна]     NVARCHAR (25) NOT NULL,

    [Город]      NVARCHAR (25) NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_локация] ASC)

)

GO

CREATE TABLE [dbo].[Ученая\_степень] (

    [Степень] NVARCHAR (25) NOT NULL,

    PRIMARY KEY CLUSTERED ([Степень] ASC)

)

GO

CREATE TABLE [dbo].[Ученое\_звание] (

    [Звание] NVARCHAR (25) NOT NULL,

    PRIMARY KEY CLUSTERED ([Звание] ASC)

)

GO

CREATE TABLE [dbo].[Авторы] (

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

    [Фамилия]             NVARCHAR (20) NOT NULL,

    [Имя]                 NVARCHAR (20) NOT NULL,

    [Отчество]            NVARCHAR (20) NOT NULL,

    [Адрес]               NVARCHAR (50) NOT NULL,

    [Почтовой\_индекс]     NVARCHAR (10) NOT NULL,

    [Рабочий\_телефон]     NVARCHAR (20) NOT NULL,

    [Домашний\_телефон]    NVARCHAR (20) NOT NULL,

    [e\_mail]              VARCHAR (30)  NOT NULL,

    [Ученая\_степень]      NVARCHAR (25) NOT NULL,

    [Ученое\_звание]       NVARCHAR (25) NOT NULL,

    [Научное\_направлвние] NVARCHAR (50) NOT NULL,

    [Кафедра]             NVARCHAR (30) NOT NULL,

    [Место\_работы]        NVARCHAR (30) NOT NULL,

    [Должность]           NVARCHAR (50) NOT NULL,

    [id\_локация]          INT           NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_автор] ASC),

    CONSTRAINT [FK\_Авторы\_Локация] FOREIGN KEY ([id\_локация]) REFERENCES [dbo].[Локация] ([id\_локация]),

    CONSTRAINT [FK\_Авторы\_Ученая\_степень] FOREIGN KEY ([Ученая\_степень]) REFERENCES [dbo].[Ученая\_степень] ([Степень]),

    CONSTRAINT [FK\_Авторы\_Ученое\_звание] FOREIGN KEY ([Ученое\_звание]) REFERENCES [dbo].[Ученое\_звание] ([Звание])

)

GO

CREATE TABLE [dbo].[Конференции] (

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

    [Название]       NVARCHAR (50) NOT NULL,

    [Организатор]    NVARCHAR (50) NOT NULL,

    [Статус]         NVARCHAR (50) DEFAULT (N'Международная') NOT NULL,

    [Дата\_начало]    DATE          NOT NULL,

    [Оргвзнос]       INT           NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_конференция] ASC)

)

GO

CREATE TABLE [dbo].[Гостиницы] (

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

    [Название]         NVARCHAR (25) NOT NULL,

    [id\_локация]       INT           NOT NULL,

    [Адрес]            NVARCHAR (50) NOT NULL,

    [Стоимость\_номера] INT           NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_гостиница] ASC),

    CONSTRAINT [FK\_Гостиницы\_Локация] FOREIGN KEY ([id\_локация]) REFERENCES [dbo].[Локация] ([id\_локация])

)

GO

CREATE TABLE [dbo].[Участники] (

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

    [id\_автор]                        INT  NULL,

    [id\_конференция]                  INT  NOT NULL,

    [id\_гостиница]                    INT  NULL,

    [Дата\_поступления\_оргвзноса]      DATE NOT NULL,

    [Размер\_оргвзноса]                INT  NOT NULL,

    [Дата\_поступления\_заявки]         DATE NOT NULL,

    [Дата\_рассылки\_1\_ого\_приглашения] DATE NOT NULL,

    [Дата\_рассылки\_2\_ого\_приглашения] DATE NOT NULL,

    [Дата\_приезда]                    DATE NOT NULL,

    [Дата\_отезда]                     DATE NOT NULL,

    [Докладчик\_участник]              BIT  NOT NULL,

    [Потребность\_в\_гостинице]         BIT  NOT NULL,

    PRIMARY KEY CLUSTERED ([id\_участник] ASC),

    UNIQUE NONCLUSTERED ([id\_автор] ASC, [id\_конференция] ASC),

    CONSTRAINT [FK\_Участники\_Авторы] FOREIGN KEY ([id\_автор]) REFERENCES [dbo].[Авторы] ([id\_автор]),

    CONSTRAINT [FK\_Участники\_Конференции] FOREIGN KEY ([id\_конференция]) REFERENCES [dbo].[Конференции] ([id\_конференция])

)

GO

CREATE TABLE [dbo].[Доклады] (

    [id\_участник]                   INT            NOT NULL,

    [Тема\_доклада]                  NVARCHAR (200) NOT NULL,

    [Отметка\_о\_поступлении\_тезисов] BIT            NOT NULL,

    [Количество\_страниц]            INT            NOT NULL,

    [Соавторы]                      NVARCHAR (50)  NULL,

    PRIMARY KEY CLUSTERED ([Тема\_доклада] ASC),

    CONSTRAINT [FK\_Доклады\_Участники] FOREIGN KEY ([id\_участник]) REFERENCES [dbo].[Участники] ([id\_участник])

)

GO

INSERT INTO Конференции (Название, Организатор, Статус, Дата\_начало, Оргвзнос)

VALUES(@Название, @Организатор, @Статус, @Дата\_начало, @Оргвзнос)

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 Conference

{

    static class Program

    {

        /// <summary>

        /// Главная точка входа для приложения.

        /// </summary>

        [STAThread]

        static void Main()

        {

            Application.EnableVisualStyles();

            Application.SetCompatibleTextRenderingDefault(false);

            Application.Run(new mainForm());

        }

    }

}

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 Conference

{

    public partial class mainForm : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public mainForm()

        {

            InitializeComponent();

        }

        //при загрузке программы

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

        {

            toolStripStatusLabel1.Visible = false;

            UpdateAll();

            textBox5.Text = "Компания Google приглашает " +

                "Вас принять участие в международной научно-технической конференции " +

                "«Искусственных интеллект и актуальные проблемы в плане SKYNET по захвату мира». " +

                "\n\tСроки проведения конференции - 20 - 24 января 2021 года. " +

                "\n\tПросим подтвердить участие в конференции и выслать тему доклада " +

                "до 1 матра 2021 года. \n\tЗаявки принимаются по электронной почте. " +

                "\n\tАдрес: fan-fan-2021@gmail.com";

        }

        //кнопка Файл->Обновить

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

        //при закрытии программы (всеми способами), сначало закрыть связь с БД, потом выйти из программы

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

        {

            sqlConnection.Close();

            Application.Exit();

        }

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

        {

            sqlConnection.Close();

            Application.Exit();

        }

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

        {

            sqlConnection.Close();

            Application.Exit();

        }

        //загрузка и обновление данных в таблицах программы

        private async void UpdateAll()

        {

            //если есть хотя бы один запись в любом из таблиц то очистка

            if (dataGridView1.Rows.Count != 0 || dataGridView3.Rows.Count != 0 ||

                dataGridView3.Rows.Count != 0 || dataGridView4.Rows.Count != 0 ||

                dataGridView5.Rows.Count != 0)

            {

                dataGridView1.Rows.Clear();

                dataGridView2.Rows.Clear();

                dataGridView3.Rows.Clear();

                dataGridView4.Rows.Clear();

                dataGridView5.Rows.Clear();

            }

            //установка связи с БД и заполнение таблиц

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            //заполнение таблицы конференции

            string cmd = "SELECT \* FROM Конференции";

            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]);

                    for (int i = 0; i < 6; i++)

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

                }

                reader.Close();

                foreach (string[] s in data)

                    dataGridView2.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

            //заполнение таблицы Авторы

            string cmd2 = "SELECT Авторы.id\_автор, Авторы.Фамилия, Авторы.Имя, Авторы.Отчество, Авторы.Адрес, " +

                "Авторы.Почтовой\_индекс, Авторы.Рабочий\_телефон, Авторы.Домашний\_телефон, Авторы.e\_mail, " +

                "Авторы.Ученая\_степень, Авторы.Ученое\_звание, Авторы.Научное\_направлвние, Авторы.Кафедра, " +

                "Авторы.Место\_работы, Авторы.Должность, Локация.Страна, Локация.Город " +

                "FROM Авторы INNER JOIN Локация ON Авторы.id\_локация = Локация.id\_локация";

            SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

            try

            {

                SqlDataReader reader2 = command2.ExecuteReader();

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

                while (reader2.Read())

                {

                    data2.Add(new string[17]);

                    for (int i = 0; i < 17; i++)

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

                }

                reader2.Close();

                foreach (string[] s in data2)

                    dataGridView1.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

            //заполнение таблицы участники

            string cmd3 = "SELECT \* FROM Участники";

            SqlCommand command3 = new SqlCommand(cmd3, sqlConnection);

            try

            {

                SqlDataReader reader3 = command3.ExecuteReader();

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

                while (reader3.Read())

                {

                    data3.Add(new string[13]);

                    for (int i = 0; i < 13; i++)

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

                }

                reader3.Close();

                foreach (string[] s in data3)

                    dataGridView3.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

            //Заполнение таблицы Доклады

            string cmd4 = "SELECT \* FROM Доклады";

            SqlCommand command4 = new SqlCommand(cmd4, sqlConnection);

            try

            {

                SqlDataReader reader4 = command4.ExecuteReader();

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

                while (reader4.Read())

                {

                    data4.Add(new string[5]);

                    for (int i = 0; i < 5; i++)

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

                }

                reader4.Close();

                foreach (string[] s in data4)

                    dataGridView4.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

            //заполнение таблицы Гостиницы

            string cmd5 = "SELECT Гостиницы.id\_гостиница, Гостиницы.Название, Гостиницы.Адрес, " +

                "Локация.Страна, Локация.Город, Гостиницы.Стоимость\_номера " +

                "FROM Гостиницы INNER JOIN Локация ON Гостиницы.id\_локация = Локация.id\_локация";

            SqlCommand command5 = new SqlCommand(cmd5, sqlConnection);

            try

            {

                SqlDataReader reader5 = command5.ExecuteReader();

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

                while (reader5.Read())

                {

                    data5.Add(new string[6]);

                    for (int i = 0; i < 6; i++)

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

                }

                reader5.Close();

                foreach (string[] s in data5)

                    dataGridView5.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

        }

        //добавить конференцию

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

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

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

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

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

            {

                string cmd = "INSERT INTO Конференции(Название, Организатор, Статус, Дата\_начало, Оргвзнос) " +

                    "VALUES(@Название, @Организатор, @Статус, @Дата\_начало, @Оргвзнос)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("Название", названиеTextBox.Text);

                command.Parameters.AddWithValue("Организатор", организаторTextBox.Text);

                command.Parameters.AddWithValue("Статус", статусTextBox.Text);

                command.Parameters.AddWithValue("Оргвзнос", оргвзносTextBox.Text);

                command.Parameters.AddWithValue("Дата\_начало", дата\_началоDateTimePicker.Value);

                try

                {

                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Данные конференции успешно добавлени в базу данных.";

                    toolStripStatusLabel1.Visible = true;

                }

                catch (Exception ex)

                {

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

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоля 'Название', 'Организатор', 'Статус', 'Дата\_начало' и" +

                    " 'Оргвзнос' должны быть заполнены.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                toolStripStatusLabel1.Text = "Поля 'Название', 'Организатор', 'Статус', 'Дата\_начало' и" +

                    " 'Оргвзнос' должны быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

            UpdateAll();

        }

        //удалить конференцию

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

        {

            DeleteConference del = new DeleteConference();

            del.ShowDialog();

            UpdateAll();

        }

        //добавить автора

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            //если все поля заполнены

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

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

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

               !string.IsNullOrEmpty(ученая\_степеньTextBox.Text) && !string.IsNullOrWhiteSpace(ученая\_степеньTextBox.Text) &&

               !string.IsNullOrEmpty(ученое\_званиеTextBox.Text) && !string.IsNullOrWhiteSpace(ученое\_званиеTextBox.Text) &&

               !string.IsNullOrEmpty(научное\_направлвниеTextBox.Text) && !string.IsNullOrWhiteSpace(научное\_направлвниеTextBox.Text) &&

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

               !string.IsNullOrEmpty(почтовой\_индексTextBox.Text) && !string.IsNullOrWhiteSpace(почтовой\_индексTextBox.Text) &&

               !string.IsNullOrEmpty(рабочий\_телефонTextBox.Text) && !string.IsNullOrWhiteSpace(рабочий\_телефонTextBox.Text) &&

               !string.IsNullOrEmpty(домашний\_телефонTextBox.Text) && !string.IsNullOrWhiteSpace(домашний\_телефонTextBox.Text) &&

               !string.IsNullOrEmpty(e\_mailTextBox.Text) && !string.IsNullOrWhiteSpace(e\_mailTextBox.Text) &&

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

               !string.IsNullOrEmpty(место\_работыTextBox.Text) && !string.IsNullOrWhiteSpace(место\_работыTextBox.Text) &&

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

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

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

            {

                int id\_location = 0;

                while (id\_location == 0)

                {

                    string searchCity = "SELECT id\_локация, COUNT(id\_локация) " +

                                        "FROM Локация " +

                                        "WHERE Город = @Город " +

                                        "GROUP BY id\_локация";

                    SqlCommand commandCity = new SqlCommand(searchCity, sqlConnection);

                    commandCity.Parameters.AddWithValue("Город", городTextBox.Text);

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

                    try

                    {

                        SqlDataReader sqlDataReader = await commandCity.ExecuteReaderAsync();

                        while (sqlDataReader.Read())

                        {

                            data.Add(new string[2]);

                            for (int i = 0; i < 2; i++)

                            {

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

                            }

                        }

                        sqlDataReader.Close();

                        if (data.Count != 0)

                        {

                            id\_location = Convert.ToInt32(data[0][0].ToString());

                            break;

                        }

                        else

                        {

                            string cityInsert = "INSERT INTO Локация(Страна, Город) VALUES(@Страна, @Город)";

                            SqlCommand CityInsertCmd = new SqlCommand(cityInsert, sqlConnection);

                            CityInsertCmd.Parameters.AddWithValue("Страна", странаTextBox.Text);

                            CityInsertCmd.Parameters.AddWithValue("Город", городTextBox.Text);

                            try

                            {

                                await CityInsertCmd.ExecuteNonQueryAsync();

                            }

                            catch (Exception ex)

                            {

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

                            }

                        }

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                string cmd = "INSERT INTO Авторы(Фамилия, Имя, Отчество, Адрес, Почтовой\_индекс, " +

                    "Рабочий\_телефон, Домашний\_телефон, e\_mail, Ученая\_степень, Ученое\_звание, " +

                    "Научное\_направлвние, Кафедра, Место\_работы, Должность, id\_локация) " +

                    "VALUES(@Фамилия, @Имя, @Отчество, @Адрес, @Почтовой\_индекс, " +

                    "@Рабочий\_телефон, @Домашний\_телефон, @e\_mail, @Ученая\_степень, @Ученое\_звание, " +

                    "@Научное\_направлвние, @Кафедра, @Место\_работы, @Должность, @id\_локация)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("Фамилия", фамилияTextBox.Text);

                command.Parameters.AddWithValue("Имя", имяTextBox.Text);

                command.Parameters.AddWithValue("Отчество", отчествоTextBox.Text);

                command.Parameters.AddWithValue("Адрес", адресTextBox.Text);

                command.Parameters.AddWithValue("Почтовой\_индекс", почтовой\_индексTextBox.Text);

                command.Parameters.AddWithValue("Рабочий\_телефон", рабочий\_телефонTextBox.Text);

                command.Parameters.AddWithValue("Домашний\_телефон", домашний\_телефонTextBox.Text);

                command.Parameters.AddWithValue("e\_mail", e\_mailTextBox.Text);

                command.Parameters.AddWithValue("Ученая\_степень", ученая\_степеньTextBox.Text);

                command.Parameters.AddWithValue("Ученое\_звание", ученое\_званиеTextBox.Text);

                command.Parameters.AddWithValue("Научное\_направлвние", научное\_направлвниеTextBox.Text);

                command.Parameters.AddWithValue("Кафедра", кафедраTextBox.Text);

                command.Parameters.AddWithValue("Место\_работы", место\_работыTextBox.Text);

                command.Parameters.AddWithValue("Должность", должностьTextBox.Text);

                command.Parameters.AddWithValue("id\_локация", id\_location);

                try

                {

                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Данные автора успешно добавлени в базу данных.";

                    toolStripStatusLabel1.Visible = true;

                }

                catch (Exception ex)

                {

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

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоля 'Фамилия', 'Имя', 'Отчество', 'Адрес', 'Почтовой\_индекс', " +

                    "'Рабочий\_телефон', 'Домашний\_телефон', 'e\_mail', 'Ученая\_степень', 'Ученое\_звание', " +

                    "'Научное\_направлвние', 'Кафедра', 'Место\_работы', 'Должность', 'Город' и 'Страна' " +

                    "должны быть заполнены.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                toolStripStatusLabel1.Text = "Ошибка! Все поля должны быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

            UpdateAll();

        }

        //удалить автора

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

        {

            DeleteAutor del = new DeleteAutor();

            del.ShowDialog();

            UpdateAll();

        }

        //добавить участника

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            //если все нужные поля заполнены

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

                !string.IsNullOrEmpty(размер\_оргвзносаTextBox.Text) && !string.IsNullOrWhiteSpace(размер\_оргвзносаTextBox.Text) &&

                (потребность\_в\_гостиницеCheckBox.Checked && !string.IsNullOrEmpty(textBox2.Text) ||

                    (!потребность\_в\_гостиницеCheckBox.Checked && string.IsNullOrEmpty(textBox2.Text))))

            {

                string cmd = "INSERT INTO Участники(id\_автор, id\_конференция, id\_гостиница, " +

                    "Дата\_поступления\_оргвзноса, Дата\_поступления\_заявки, " +

                    "Дата\_рассылки\_1\_ого\_приглашения, Дата\_рассылки\_2\_ого\_приглашения, " +

                    "Дата\_приезда, Дата\_отезда, Докладчик\_участник, Потребность\_в\_гостинице, Размер\_оргвзноса) " +

                    "VALUES(@id\_автор, @id\_конференция, @id\_гостиница, " +

                    "@Дата\_поступления\_оргвзноса, @Дата\_поступления\_заявки, " +

                    "@Дата\_рассылки\_1\_ого\_приглашения, @Дата\_рассылки\_2\_ого\_приглашения, " +

                    "@Дата\_приезда, @Дата\_отезда, @Докладчик\_участник, @Потребность\_в\_гостинице, @Размер\_оргвзноса)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("id\_автор", textBox1.Text);

                command.Parameters.AddWithValue("id\_конференция", id\_конференцияTextBox.Text);

                command.Parameters.AddWithValue("Дата\_поступления\_оргвзноса", дата\_поступления\_оргвзносаDateTimePicker.Value);

                command.Parameters.AddWithValue("Дата\_поступления\_заявки", дата\_поступления\_заявкиDateTimePicker.Value);

                command.Parameters.AddWithValue("Дата\_рассылки\_1\_ого\_приглашения", дата\_рассылки\_1\_ого\_приглашенияDateTimePicker.Value);

                command.Parameters.AddWithValue("Дата\_рассылки\_2\_ого\_приглашения", дата\_рассылки\_2\_ого\_приглашенияDateTimePicker.Value);

                command.Parameters.AddWithValue("Дата\_приезда", дата\_приездаDateTimePicker.Value);

                command.Parameters.AddWithValue("Дата\_отезда", дата\_отездаDateTimePicker.Value);

                command.Parameters.AddWithValue("Докладчик\_участник", докладчик\_участникCheckBox.Checked);

                command.Parameters.AddWithValue("Потребность\_в\_гостинице", потребность\_в\_гостиницеCheckBox.Checked);

                command.Parameters.AddWithValue("Размер\_оргвзноса", размер\_оргвзносаTextBox.Text);

                string id\_hotel = "";

                if (потребность\_в\_гостиницеCheckBox.Checked)

                {

                    string cmd2 = "SELECT id\_гостиница FROM Гостиницы WHERE id\_гостиница=@id\_гостиница";

                    SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

                    command2.Parameters.AddWithValue("id\_гостиница", textBox2.Text);

                    SqlDataReader sdaReader;

                    try

                    {

                        sdaReader = await command2.ExecuteReaderAsync();

                        while (sdaReader.Read())

                            id\_hotel = sdaReader[0].ToString();

                        sdaReader.Close();

                    }

                    catch (Exception ex)

                    {

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

                    }

                    //если гостиница есть в базе, то продолжаем

                    if (id\_hotel != "")

                    {

                        command.Parameters.AddWithValue("id\_гостиница", textBox2.Text);

                        try

                        {

                            await command.ExecuteNonQueryAsync();

                            toolStripStatusLabel1.Text = "Данные участника успешно добавлени в базу данных.";

                            toolStripStatusLabel1.Visible = true;

                        }

                        catch (Exception ex)

                        {

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

                        }

                    }

                    else

                    {

                        MessageBox.Show("Ошибка!\n\nТакой гостиницы нет в базе данных", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                        toolStripStatusLabel1.Text = "Ошибка! Такой гостиницы нет в базе данных.";

                        toolStripStatusLabel1.Visible = true;

                    }

                }

                if (!потребность\_в\_гостиницеCheckBox.Checked)

                {

                    textBox2.Text = "";

                    command.Parameters.AddWithValue("id\_гостиница", textBox2.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                        toolStripStatusLabel1.Text = "Данные участника успешно добавлени в базу данных.";

                        toolStripStatusLabel1.Visible = true;

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоля 'id\_конференция' и 'Размер\_оргвзноса' " +

                    "должны быть заполнены." +

                    "\n\nИли поле 'id\_гостиница' должен быть заполнен если " +

                    "установлен checkBox 'Гостиница'" +

                    "\n\nИли такой автор уже есть среди участников той же конференции.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                toolStripStatusLabel1.Text = "Ошибка! Поля 'id\_конференция' и 'Размер\_оргвзноса' должны быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

            UpdateAll();

        }

        //удалить участника

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

        {

            DeleteUchastnik del = new DeleteUchastnik();

            del.ShowDialog();

            UpdateAll();

        }

        //добавить доклад

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            //если все нужные поля заполнены

            if (!string.IsNullOrEmpty(id\_участникTextBox1.Text) && !string.IsNullOrWhiteSpace(id\_участникTextBox1.Text) &&

                !string.IsNullOrEmpty(тема\_докладаTextBox.Text) && !string.IsNullOrWhiteSpace(тема\_докладаTextBox.Text) &&

                !string.IsNullOrEmpty(количество\_страницTextBox.Text) && !string.IsNullOrWhiteSpace(количество\_страницTextBox.Text))

            {

                string cmd = "INSERT INTO Доклады(id\_участник, Тема\_доклада, Отметка\_о\_поступлении\_тезисов, Количество\_страниц, Соавторы) " +

                    "VALUES(@id\_участник, @Тема\_доклада, @Отметка\_о\_поступлении\_тезисов, @Количество\_страниц, @Соавторы)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("id\_участник", id\_участникTextBox1.Text);

                command.Parameters.AddWithValue("Тема\_доклада", тема\_докладаTextBox.Text);

                command.Parameters.AddWithValue("Отметка\_о\_поступлении\_тезисов", тезисcheckBox.Checked);

                command.Parameters.AddWithValue("Количество\_страниц", количество\_страницTextBox.Text);

                command.Parameters.AddWithValue("Соавторы", соавторыTextBox.Text);

                try

                {

                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Данные доклада успешно добавлени в базу данных.";

                    toolStripStatusLabel1.Visible = true;

                }

                catch (Exception ex)

                {

                    MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка",

                        MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоля 'id\_участник', 'Тема\_доклада' и 'Количество\_страниц' " +

                    "должны быть заполнены.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                toolStripStatusLabel1.Text = "Ошибка! Поля 'id\_участник', 'Тема\_доклада' " +

                    "и 'Количество\_страниц' должны быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

            UpdateAll();

        }

        //удалить доклад

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

        {

            DeleteDoklad del = new DeleteDoklad();

            del.ShowDialog();

            UpdateAll();

        }

        //добавить гостиницу

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            //если все нужные поля заполнены

            if (!string.IsNullOrEmpty(названиеTextBox1.Text) && !string.IsNullOrWhiteSpace(названиеTextBox1.Text) &&

                !string.IsNullOrEmpty(адресTextBox1.Text) && !string.IsNullOrWhiteSpace(адресTextBox1.Text) &&

                !string.IsNullOrEmpty(стоимость\_номераTextBox.Text) && !string.IsNullOrWhiteSpace(стоимость\_номераTextBox.Text) &&

                !string.IsNullOrEmpty(городTextBox1.Text) && !string.IsNullOrWhiteSpace(городTextBox1.Text) &&

                !string.IsNullOrEmpty(странаTextBox1.Text) && !string.IsNullOrWhiteSpace(странаTextBox1.Text))

            {

                int id\_location = 0;

                while (id\_location == 0)

                {

                    string searchCity = "SELECT id\_локация, COUNT(id\_локация) " +

                                        "FROM Локация " +

                                        "WHERE Город = @Город " +

                                        "GROUP BY id\_локация";

                    SqlCommand commandCity = new SqlCommand(searchCity, sqlConnection);

                    commandCity.Parameters.AddWithValue("Город", городTextBox1.Text);

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

                    try

                    {

                        SqlDataReader sqlDataReader = await commandCity.ExecuteReaderAsync();

                        while (sqlDataReader.Read())

                        {

                            data.Add(new string[2]);

                            for (int i = 0; i < 2; i++)

                            {

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

                            }

                        }

                        sqlDataReader.Close();

                        if (data.Count != 0)

                        {

                            id\_location = Convert.ToInt32(data[0][0].ToString());

                            break;

                        }

                        else

                        {

                            string cityInsert = "INSERT INTO Локация(Страна, Город) VALUES(@Страна, @Город)";

                            SqlCommand CityInsertCmd = new SqlCommand(cityInsert, sqlConnection);

                            CityInsertCmd.Parameters.AddWithValue("Страна", странаTextBox1.Text);

                            CityInsertCmd.Parameters.AddWithValue("Город", городTextBox1.Text);

                            try

                            {

                                await CityInsertCmd.ExecuteNonQueryAsync();

                            }

                            catch (Exception ex)

                            {

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

                            }

                        }

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                string cmd = "INSERT INTO Гостиницы(Название, id\_локация, Адрес, Стоимость\_номера) " +

                    "VALUES(@Название, @id\_локация, @Адрес, @Стоимость\_номера)";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("Название", названиеTextBox1.Text);

                command.Parameters.AddWithValue("id\_локация", id\_location);

                command.Parameters.AddWithValue("Адрес", адресTextBox1.Text);

                command.Parameters.AddWithValue("Стоимость\_номера", стоимость\_номераTextBox.Text);

                try

                {

                    await command.ExecuteNonQueryAsync();

                    toolStripStatusLabel1.Text = "Данные гостиницы успешно добавлени в базу данных.";

                    toolStripStatusLabel1.Visible = true;

                }

                catch (Exception ex)

                {

                    MessageBox.Show("Ошибка!\n\n" + ex.Message, "Ошибка",

                        MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоля 'Название', 'Адрес', 'Стоимость\_номера', 'Страна' и " +

                    "'Город' должны быть заполнены.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                toolStripStatusLabel1.Text = "Ошибка! Поля 'Название', 'Адрес', 'Стоимость\_номера', " +

                    "'Страна' и 'Город' должны быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

            UpdateAll();

        }

        //удалить гостиницу

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

        {

            DeleteHotel del = new DeleteHotel();

            del.ShowDialog();

            UpdateAll();

        }

        //о программе

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

        {

            About about = new About();

            about.Show();

        }

        //для указанной даты 1-ой рассылки вывести список приглашенных и посчитать их количество

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

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

            {

                PervoePriglashenie f = new PervoePriglashenie(dateTimePicker3.Value, idConference.Text);

                f.Show();

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПолe 'id\_конференция' должен быть заполнен.",

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

                toolStripStatusLabel1.Text = "Ошибка! Полe 'id\_конференция' должен быть заполнен.";

                toolStripStatusLabel1.Visible = true;

            }

        }

        //вывести список приглашенных, с указанием даты об уплате оргвзноса

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

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

            {

                OrgVznosDateSpisok f = new OrgVznosDateSpisok(textBox3.Text);

                f.Show();

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПолe 'id\_конференция' должен быть заполнен.",

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

                toolStripStatusLabel1.Text = "Ошибка! Полe 'id\_конференция' должен быть заполнен.";

                toolStripStatusLabel1.Visible = true;

            }

        }

        //для указанной интервала дат, вывести список участников, уплативших оргвзнос в этом диапазоне

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

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

            {

                //если дата "с" не позже даты "до", то все ок, продолжаем

                if (dateTimePicker1.Value <= dateTimePicker2.Value)

                {

                    OrgVznosDateInterval f = new OrgVznosDateInterval(textBox4.Text,

                        dateTimePicker1.Value, dateTimePicker2.Value);

                    f.Show();

                }

                else

                {

                    MessageBox.Show("Ошибка!\n\nНеправильно выставлены даты \"с\" и \"до\".",

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

                    toolStripStatusLabel1.Text = "Ошибка! Неправильно выставлены даты \"с\" и \"до\".";

                    toolStripStatusLabel1.Visible = true;

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПолe 'id\_конференция' должен быть заполнен.",

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

                toolStripStatusLabel1.Text = "Ошибка! Полe 'id\_конференция' должен быть заполнен.";

                toolStripStatusLabel1.Visible = true;

            }

        }

        //для указанного города вывести название тезисов докладов, поступивших из этого города

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

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

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

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

            {

                TeseusDokladus f = new TeseusDokladus(gorodtextbox.Text, idconfertextbox.Text);

                f.Show();

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоля 'id\_конференция' и 'Город' должены быть заполнены.",

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

                toolStripStatusLabel1.Text = "Ошибка! Поля 'id\_конференция' и 'Город' должены быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

        }

        //для указанного города, вывести список нуждающихся в гостинице

        private void button15\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

            if (!string.IsNullOrEmpty(город2textbox.Text) && !string.IsNullOrWhiteSpace(город2textbox.Text) &&

                !string.IsNullOrEmpty(idconference12.Text) && !string.IsNullOrWhiteSpace(idconference12.Text))

            {

                TrueHotel f = new TrueHotel(город2textbox.Text, idconference12.Text);

                f.Show();

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоля 'id\_конференция' и 'Город' должены быть заполнены.",

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

                toolStripStatusLabel1.Text = "Ошибка! Поля 'id\_конференция' и 'Город' должены быть заполнены.";

                toolStripStatusLabel1.Visible = true;

            }

        }

        //приглашение

        private void button16\_Click(object sender, EventArgs e)

        {

            if (toolStripStatusLabel1.Visible)

                toolStripStatusLabel1.Visible = false;

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

            {

                string cmd2 = "SELECT Авторы.Фамилия, Авторы.Имя, Авторы.Отчество " +

                          "FROM Участники INNER JOIN Авторы ON Участники.id\_автор = Авторы.id\_автор " +

                          "WHERE Участники.id\_конференция = @id\_конференция";

                SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

                command2.Parameters.AddWithValue("id\_конференция", textBox6.Text);

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

                try

                {

                    SqlDataReader reader2 = command2.ExecuteReader();

                    while (reader2.Read())

                    {

                        data2.Add(new string[3]);

                        for (int i = 0; i < 3; i++)

                        {

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

                        }

                    }

                    reader2.Close();

                }

                catch (System.Exception ex)

                {

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

                }

                //название конференции

                string cmd = "SELECT Название FROM Конференции " +

                    "WHERE id\_конференция = @id\_конференция";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("id\_конференция", textBox6.Text);

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

                try

                {

                    SqlDataReader reader = command.ExecuteReader();

                    while (reader.Read())

                    {

                        data.Add(new string[1]);

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

                    }

                    reader.Close();

                }

                catch (System.Exception ex)

                {

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

                }

                // Создаем документ Word.

                object oMissing = System.Reflection.Missing.Value;

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

                //Start Word and create a new document.

                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);

                foreach (string[] fio in data2)

                {

                    //Insert a paragraph at the beginning of the document.

                    Word.Paragraph oPara1;

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

                    oPara1.Range.Text = "П Р И Г Л А Ш Е Н И Е";

                    oPara1.Range.Font.Size = 24;

                    oPara1.Range.Font.Bold = 1;

                    oPara1.Format.SpaceAfter = 6;    //6 pt spacing after paragraph.

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

                    oPara1.Range.InsertParagraphAfter();

                    //Insert a paragraph at the end of the document.

                    Word.Paragraph oPara2;

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

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

                    oPara2.Range.Text = "на конфернцию " + data[0][0].ToString();

                    oPara1.Range.Font.Size = 16;

                    oPara2.Range.Font.Bold = 1;

                    oPara2.Format.SpaceAfter = 20;

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

                    oPara2.Range.InsertParagraphAfter();

                    //Insert another paragraph.

                    Word.Paragraph oPara3;

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

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

                    if (fio[2][fio[2].Count() - 1].ToString() == "ч")

                    {

                        oPara3.Range.Text = "Уважаемый " + fio[0].ToString() + " " + fio[1].ToString() + " " + fio[2].ToString();

                    }

                    if (fio[2][fio[2].Count() - 1].ToString() == "а")

                    {

                        oPara3.Range.Text = "Уважаемая " + fio[0].ToString() + " " + fio[1].ToString() + " " + fio[2].ToString();

                    }

                    oPara3.Range.Font.Bold = 1;

                    oPara3.Format.SpaceAfter = 12;

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

                    oPara3.Range.InsertParagraphAfter();

                    Word.Paragraph oPara7;

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

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

                    oPara7.Range.Text = " ";

                    oPara7.Range.Font.Bold = 0;

                    oPara7.Format.SpaceAfter = 0;

                    oPara7.Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphJustify;

                    oPara7.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 = "\t" + textBox5.Text;

                    oPara4.Range.Font.Bold = 0;

                    oPara4.Format.SpaceAfter = 0;

                    oPara4.Range.ParagraphFormat.Alignment = Word.WdParagraphAlignment.wdAlignParagraphJustify;

                    //разрыв страницы

                    oDoc.Words.Last.InsertBreak(Word.WdBreakType.wdPageBreak);

                }

                MessageBox.Show("Приглашения созданы!", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПолe 'id\_конференция' должен быть заполнен.",

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

                toolStripStatusLabel1.Text = "Ошибка! Полe 'id\_конференция' должен быть заполнен.";

                toolStripStatusLabel1.Visible = true;

            }

        }

        //изменить конференцию

        private void button17\_Click(object sender, EventArgs e)

        {

            UpdateConference f = new UpdateConference();

            f.ShowDialog();

            UpdateAll();

        }

        //изменить автора

        private void button18\_Click(object sender, EventArgs e)

        {

            UpdateAutor f = new UpdateAutor();

            f.ShowDialog();

            UpdateAll();

        }

        //изменить участника

        private void button19\_Click(object sender, EventArgs e)

        {

            UpdateUchastnik f = new UpdateUchastnik();

            f.ShowDialog();

            UpdateAll();

        }

        //изменить доклад

        private void button20\_Click(object sender, EventArgs e)

        {

            UpdateDoklad f = new UpdateDoklad();

            f.ShowDialog();

            UpdateAll();

        }

        //изменить гостиницу

        private void button21\_Click(object sender, EventArgs e)

        {

            UpdateHotel f = new UpdateHotel();

            f.ShowDialog();

            UpdateAll();

        }

    }

}

Листинг 3. FILE: About.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Linq;

using System.Reflection;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Conference

{

    partial class About : Form

    {

        public About()

        {

            InitializeComponent();

            this.Text = String.Format("О программе {0}", "Конференции");

            this.labelProductName.Text = "Наименование продукта: Конференции";

            this.labelVersion.Text = String.Format("Версия: {0}","1.0.0.1");

            this.labelCopyright.Text = "(c) Copyright: Диденко Владислав 2020";

            this.labelCompanyName.Text = "Организация: СПбГЭТУ \"ЛЭТИ\"";

            this.textBoxDescription.Text = AssemblyDescription;

        }

        #region Методы доступа к атрибутам сборки

        public string AssemblyTitle

        {

            get

            {

                object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyTitleAttribute), false);

                if (attributes.Length > 0)

                {

                    AssemblyTitleAttribute titleAttribute = (AssemblyTitleAttribute)attributes[0];

                    if (titleAttribute.Title != "")

                    {

                        return titleAttribute.Title;

                    }

                }

                return System.IO.Path.GetFileNameWithoutExtension(Assembly.GetExecutingAssembly().CodeBase);

            }

        }

        public string AssemblyVersion

        {

            get

            {

                return Assembly.GetExecutingAssembly().GetName().Version.ToString();

            }

        }

        public string AssemblyDescription

        {

            get

            {

                object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyDescriptionAttribute), false);

                if (attributes.Length == 0)

                {

                    return "";

                }

                return ((AssemblyDescriptionAttribute)attributes[0]).Description;

            }

        }

        public string AssemblyProduct

        {

            get

            {

                object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyProductAttribute), false);

                if (attributes.Length == 0)

                {

                    return "";

                }

                return ((AssemblyProductAttribute)attributes[0]).Product;

            }

        }

        public string AssemblyCopyright

        {

            get

            {

                object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCopyrightAttribute), false);

                if (attributes.Length == 0)

                {

                    return "";

                }

                return ((AssemblyCopyrightAttribute)attributes[0]).Copyright;

            }

        }

        public string AssemblyCompany

        {

            get

            {

                object[] attributes = Assembly.GetExecutingAssembly().GetCustomAttributes(typeof(AssemblyCompanyAttribute), false);

                if (attributes.Length == 0)

                {

                    return "";

                }

                return ((AssemblyCompanyAttribute)attributes[0]).Company;

            }

        }

        #endregion

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

        {

            this.Close();

        }

    }

}

Листинг 4. FILE: OrgVznosDateInterval.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 Conference

{

    public partial class OrgVznosDateInterval : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public OrgVznosDateInterval(string idConference, DateTime from, DateTime to)

        {

            InitializeComponent();

            FormLoad(idConference, from, to);

        }

        private async void FormLoad(string idConference, DateTime from, DateTime to)

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            //даты "с" и "до" //вывод

            fromlabel.Text= from.ToString("yyyy-MM-dd");

            tolabel.Text = to.ToString("yyyy-MM-dd");

            //название конференции

            string cmd = "SELECT Название FROM Конференции " +

                "WHERE id\_конференция = @id\_конференция";

            SqlCommand command = new SqlCommand(cmd, sqlConnection);

            command.Parameters.AddWithValue("id\_конференция", idConference);

            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();

                foreach (string[] s in data)

                    конференция.Text = "\"" + s[0].ToString() + "\"";

            }

            catch (System.Exception ex)

            {

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

            }

            //заполнение таблицы

            string cmd2 = "SELECT Участники.id\_участник, Авторы.Фамилия, Авторы.Имя, " +

                "Авторы.Отчество, Участники.Дата\_поступления\_оргвзноса " +

                "FROM  Участники INNER JOIN Авторы ON Участники.id\_автор = Авторы.id\_автор " +

                "WHERE (Участники.id\_конференция = @id\_конференция " +

                        "AND (Дата\_поступления\_оргвзноса BETWEEN @fromdate AND @todate))";

            SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

            command2.Parameters.AddWithValue("id\_конференция", idConference);

            command2.Parameters.AddWithValue("fromdate", from.ToString("yyyy-MM-dd"));

            command2.Parameters.AddWithValue("todate", to.ToString("yyyy-MM-dd"));

            try

            {

                SqlDataReader reader2 = command2.ExecuteReader();

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

                while (reader2.Read())

                {

                    data2.Add(new string[5]);

                    for (int i = 0; i < 5; i++)

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

                }

                reader2.Close();

                foreach (string[] s in data2)

                    dataGridView1.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

            sqlConnection.Close();

        }

    }

}

Листинг 5. FILE: OrgVznosDateSpisok.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 Conference

{

    public partial class OrgVznosDateSpisok : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public OrgVznosDateSpisok(string idConference)

        {

            InitializeComponent();

            Form\_load(idConference);

        }

        private async void Form\_load(string idConference)

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            //название конференции //выводим

            string cmd = "SELECT Название FROM Конференции " +

                "WHERE id\_конференция = @id\_конференция";

            SqlCommand command = new SqlCommand(cmd, sqlConnection);

            command.Parameters.AddWithValue("id\_конференция", idConference);

            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();

                foreach (string[] s in data)

                    конференция.Text = "\"" + s[0].ToString() + "\"";

            }

            catch (System.Exception ex)

            {

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

            }

            //заполнение таблицы

            string cmd2 = "SELECT Участники.id\_участник, Авторы.Фамилия, Авторы.Имя, " +

                "Авторы.Отчество, Участники.Дата\_поступления\_оргвзноса " +

                "FROM  Участники INNER JOIN Авторы ON Участники.id\_автор = Авторы.id\_автор " +

                "WHERE Участники.id\_конференция = @id\_конференция";

            SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

            command2.Parameters.AddWithValue("id\_конференция", idConference);

            try

            {

                SqlDataReader reader2 = command2.ExecuteReader();

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

                while (reader2.Read())

                {

                    data2.Add(new string[5]);

                    for (int i = 0; i < 5; i++)

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

                }

                reader2.Close();

                foreach (string[] s in data2)

                    dataGridView1.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

            sqlConnection.Close();

        }

    }

}

Листинг 6. FILE: TeseusDokladus.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 Conference

{

    public partial class TeseusDokladus : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public TeseusDokladus(string gorod, string idConference)

        {

            InitializeComponent();

            Form\_Load(gorod, idConference);

        }

        private async void Form\_Load(string gorod, string idConference)

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            //город название //вывод

            городlabel.Text = gorod;

            //название конференции

            string cmd = "SELECT Название FROM Конференции " +

                "WHERE id\_конференция = @id\_конференция";

            SqlCommand command = new SqlCommand(cmd, sqlConnection);

            command.Parameters.AddWithValue("id\_конференция", idConference);

            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();

                foreach (string[] s in data)

                    конференция.Text = "\"" + s[0].ToString() + "\"";

            }

            catch (System.Exception ex)

            {

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

            }

            //заполнение таблицы

            string cmd2 = "SELECT Доклады.Тема\_доклада " +

                            "FROM Доклады " +

                                "INNER JOIN (Участники INNER JOIN " +

                                                "(Авторы INNER JOIN Локация " +

                                                            "ON Авторы.id\_локация = Локация.id\_локация) " +

                                            "ON Участники.id\_автор = Авторы.id\_автор) " +

                                "ON Участники.id\_участник = Доклады.id\_участник " +

                            "WHERE(Локация.Город = @город AND Участники.id\_конференция = @id\_конференция)";

            SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

            command2.Parameters.AddWithValue("id\_конференция", idConference);

            command2.Parameters.AddWithValue("город", gorod);

            try

            {

                SqlDataReader reader2 = command2.ExecuteReader();

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

                while (reader2.Read())

                {

                    data2.Add(new string[1]);

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

                }

                reader2.Close();

                foreach (string[] s in data2)

                    dataGridView1.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

        }

    }

}

Листинг 7. FILE: TrueHotel.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 Conference

{

    public partial class TrueHotel : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public TrueHotel(string gorod, string idConference)

        {

            InitializeComponent();

            Form\_Load(gorod, idConference);

        }

        private async void Form\_Load(string gorod, string idConference)

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            //город название //вывод

            городlabel.Text = gorod;

            //название конференции

            string cmd = "SELECT Название FROM Конференции " +

                "WHERE id\_конференция = @id\_конференция";

            SqlCommand command = new SqlCommand(cmd, sqlConnection);

            command.Parameters.AddWithValue("id\_конференция", idConference);

            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();

                foreach (string[] s in data)

                    конференция.Text = "\"" + s[0].ToString() + "\"";

            }

            catch (System.Exception ex)

            {

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

            }

            //заполнение таблицы

            string cmd2 = "SELECT Участники.id\_участник, Авторы.Фамилия, Авторы.Имя, Авторы.Отчество " +

       "FROM Участники INNER JOIN (Авторы INNER JOIN Локация " +

"ON Авторы.id\_локация = Локация.id\_локация) " +

"ON Участники.id\_автор = Авторы.id\_автор " +

       "WHERE(Локация.Город = @город " +

             "AND Участники.Потребность\_в\_гостинице = 1 " +

             "AND Участники.id\_конференция = @id\_конференция)";

            SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

            command2.Parameters.AddWithValue("id\_конференция", idConference);

            command2.Parameters.AddWithValue("город", gorod);

            try

            {

                SqlDataReader reader2 = command2.ExecuteReader();

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

                while (reader2.Read())

                {

                    data2.Add(new string[4]);

                    for (int i = 0; i < 4; i++)

                    {

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

                    }

                }

                reader2.Close();

                foreach (string[] s in data2)

                    dataGridView1.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

            sqlConnection.Close();

        }

    }

}

Листинг 8. FILE: PervoePriglashenie.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 Conference

{

    public partial class PervoePriglashenie : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public PervoePriglashenie(DateTime dateTime, string idConference)

        {

            InitializeComponent();

            Form\_Load(dateTime, idConference);

        }

        private async void Form\_Load(DateTime dateTime, string idConference)

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            //для какой даты инфа о приглашенных //выводим

            датаLabel.Text = dateTime.ToString("yyyy-MM-dd");

            //заполнение таблицы Приглашенных

            string cmd2 = "SELECT Участники.id\_участник, Авторы.Фамилия, Авторы.Имя, Авторы.Отчество " +

                "FROM Участники INNER JOIN Авторы ON Участники.id\_автор = Авторы.id\_автор " +

                "WHERE (Участники.Дата\_рассылки\_1\_ого\_приглашения = @Дата " +

                            "AND Участники.id\_конференция = @id\_конференция)";

            SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

            command2.Parameters.AddWithValue("Дата", dateTime.ToString("yyyy-MM-dd"));

            command2.Parameters.AddWithValue("id\_конференция", idConference);

            try

            {

                SqlDataReader reader2 = command2.ExecuteReader();

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

                while (reader2.Read())

                {

                    data2.Add(new string[4]);

                    for (int i = 0; i < 4; i++)

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

                }

                reader2.Close();

                foreach (string[] s in data2)

                    dataGridView1.Rows.Add(s);

            }

            catch (System.Exception ex)

            {

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

            }

            //подсчет количества приглашенных

            //заполнение таблицы Приглашенных

            string cmd = "SELECT COUNT(id\_участник) FROM Участники " +

                "WHERE (id\_конференция = @id\_конференция " +

                        "AND Дата\_рассылки\_1\_ого\_приглашения = @Дата) " +

                "GROUP BY id\_конференция";

            SqlCommand command = new SqlCommand(cmd, sqlConnection);

            command.Parameters.AddWithValue("Дата", dateTime.ToString("yyyy-MM-dd"));

            command.Parameters.AddWithValue("id\_конференция", idConference);

            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();

                foreach (string[] s in data)

                    количество.Text = s[0].ToString();

            }

            catch (System.Exception ex)

            {

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

            }

            sqlConnection.Close();

        }

    }

}

Листинг 9. FILE: UpdateUchastnik.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 Conference

{

    public partial class UpdateUchastnik : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public UpdateUchastnik()

        {

            InitializeComponent();

        }

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            if (!string.IsNullOrEmpty(id\_участникTextBox.Text) && (checkBox1.Checked ||

                checkBox2.Checked  || checkBox4.Checked  || checkBox9.Checked || checkBox10.Checked ||

                checkBox8.Checked  || checkBox5.Checked  || checkBox3.Checked || checkBox7.Checked  ||

                checkBox13.Checked || checkBox6.Checked  || checkBox11.Checked))

            {

                //id автор

                if (checkBox1.Checked)

                {

                    string cmd = "UPDATE Участники SET id\_автор = @id\_автор WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("id\_автор", textBox1.Text);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //id конференции

                if (checkBox2.Checked)

                {

                    string cmd = "UPDATE Участники SET id\_конференция = @id\_конференция WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("id\_конференция", textBox2.Text);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //id гостиница

                if (checkBox4.Checked)

                {

                    if (checkBox13.Checked)

                    {

                        string cmd = "UPDATE Участники SET id\_гостиница = @id\_гостиница, " +

                            "Потребность\_в\_гостинице = @Потребность WHERE id\_участник = @id\_участник";

                        SqlCommand command = new SqlCommand(cmd, sqlConnection);

                        command.Parameters.AddWithValue("id\_гостиница", textBox4.Text);

                        command.Parameters.AddWithValue("Потребность", checkBox14.Checked);

                        command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                        try

                        {

                            await command.ExecuteNonQueryAsync();

                        }

                        catch (Exception ex)

                        {

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

                        }

                    }

                    else

                    {

                        string cmd2 = "UPDATE Участники SET id\_гостиница = @id\_гостиница WHERE id\_участник = @id\_участник";

                        SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

                        command2.Parameters.AddWithValue("id\_гостиница", textBox4.Text);

                        command2.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                        try

                        {

                            await command2.ExecuteNonQueryAsync();

                        }

                        catch (Exception ex)

                        {

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

                        }

                    }

                }

                //размер орвзноса

                if (checkBox10.Checked)

                {

                    string cmd = "UPDATE Участники SET Размер\_оргвзноса = @Размер\_оргвзноса WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Размер\_оргвзноса", textBox3.Text);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //потрбность в гостинице

                if (checkBox13.Checked)

                {

                    string cmd = "UPDATE Участники SET Потребность\_в\_гостинице = @Потребность WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Потребность", checkBox14.Checked);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                // докладчик/участник

                if (checkBox9.Checked)

                {

                    string cmd = "UPDATE Участники SET Докладчик\_участник = @ДокладчикУчастник WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("ДокладчикУчастник", checkBox12.Checked);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //дата поступления оргвзноса

                if (checkBox5.Checked)

                {

                    string cmd = "UPDATE Участники SET Дата\_поступления\_оргвзноса = @Датавзноса WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Датавзноса", дата\_началоDateTimePicker.Value);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //дата поступления заявки

                if (checkBox3.Checked)

                {

                    string cmd = "UPDATE Участники SET Дата\_поступления\_заявки = @Датазаявки WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Датазаявки", dateTimePicker1.Value);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //дата приезда

                if (checkBox7.Checked)

                {

                    string cmd = "UPDATE Участники SET Дата\_приезда = @ДатазПриезда WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("ДатазПриезда", dateTimePicker3.Value);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //дата отъезда

                if (checkBox8.Checked)

                {

                    string cmd = "UPDATE Участники SET Дата\_отезда = @ДатаОтъезда WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("ДатаОтъезда", dateTimePicker4.Value);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //дата 1ого приглашения

                if (checkBox6.Checked)

                {

                    string cmd = "UPDATE Участники SET Дата\_рассылки\_1\_ого\_приглашения = @Дата1огоПриглаш WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Дата1огоПриглаш", dateTimePicker2.Value);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //дата 2ого приглашения

                if (checkBox11.Checked)

                {

                    string cmd = "UPDATE Участники SET Дата\_рассылки\_2\_ого\_приглашения = @Дата2огоПриглаш WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Дата2огоПриглаш", dateTimePicker6.Value);

                    command.Parameters.AddWithValue("id\_участник", id\_участникTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

            }

            sqlConnection.Close();

            MessageBox.Show("Данные успешно изменени!\n\n", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

            this.Close();

        }

    }

}

Листинг 10. FILE: UpdateDoklad.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 Conference

{

    public partial class UpdateDoklad : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public UpdateDoklad()

        {

            InitializeComponent();

        }

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            if (!string.IsNullOrEmpty(id\_TextBox.Text) && (checkBox1.Checked || checkBox2.Checked || checkBox3.Checked || checkBox4.Checked))

            {

                if (checkBox2.Checked)

                {

                    string cmd = "UPDATE Доклады SET Тема\_доклада = @Тема\_доклада WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Тема\_доклада", textBox2.Text);

                    command.Parameters.AddWithValue("id\_участник", id\_TextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (checkBox1.Checked)

                {

                    string cmd = "UPDATE Доклады SET Количество\_страниц = @Количество\_страниц WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Количество\_страниц", textBox1.Text);

                    command.Parameters.AddWithValue("id\_участник", id\_TextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (checkBox4.Checked)

                {

                    string cmd = "UPDATE Доклады SET Отметка\_о\_поступлении\_тезисов = @Отметка WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Отметка", checkBox5.Checked);

                    command.Parameters.AddWithValue("id\_участник", id\_TextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (checkBox3.Checked)

                {

                    string cmd = "UPDATE Доклады SET Соавторы = @Соавторы WHERE id\_участник = @id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Соавторы", textBox3.Text);

                    command.Parameters.AddWithValue("id\_участник", id\_TextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                sqlConnection.Close();

                MessageBox.Show("Данные успешно изменени!\n\n", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

                this.Close();

            }

            sqlConnection.Close();

        }

    }

}

Листинг 11. FILE: UpdateHotel.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 Conference

{

    public partial class UpdateHotel : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public UpdateHotel()

        {

            InitializeComponent();

        }

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            if (checkBox1.Checked || checkBox4.Checked || checkBox5.Checked)

            {

                if (checkBox1.Checked)

                {

                    string cmd = "UPDATE Гостиницы SET Название = @Название WHERE id\_гостиница = @id\_гостиница";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Название", textBox4.Text);

                    command.Parameters.AddWithValue("id\_гостиница", id\_hotelTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (checkBox4.Checked)

                {

                    string cmd = "UPDATE Гостиницы SET Адрес = @Адрес WHERE id\_гостиница = @id\_гостиница";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Адрес", textBox3.Text);

                    command.Parameters.AddWithValue("id\_гостиница", id\_hotelTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (checkBox5.Checked)

                {

                    string cmd = "UPDATE Гостиницы SET Стоимость\_номера = @Стоимость\_номера WHERE id\_гостиница = @id\_гостиница";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Стоимость\_номера", textBox5.Text);

                    command.Parameters.AddWithValue("id\_гостиница", id\_hotelTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                sqlConnection.Close();

                MessageBox.Show("Данные успешно изменени!\n\n", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

                this.Close();

            }

        }

    }

}

Листинг 12. FILE: UpdateConference.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 Conference

{

    public partial class UpdateConference : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public UpdateConference()

        {

            InitializeComponent();

        }

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            if (checkBox1.Checked || checkBox2.Checked || checkBox3.Checked || checkBox4.Checked || checkBox5.Checked)

            {

                if (checkBox1.Checked)

                {

                    string cmd = "UPDATE Конференции SET Название = @Название WHERE id\_конференция = @id\_конференция";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

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

                    command.Parameters.AddWithValue("id\_конференция", id\_конференцияTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (checkBox2.Checked)

                {

                    string cmd = "UPDATE Конференции SET Организатор = @Организатор WHERE id\_конференция = @id\_конференция";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

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

                    command.Parameters.AddWithValue("id\_конференция", id\_конференцияTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (checkBox3.Checked)

                {

                    string cmd = "UPDATE Конференции SET Статус = @Статус WHERE id\_конференция = @id\_конференция";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Статус", textBox3.Text);

                    command.Parameters.AddWithValue("id\_конференция", id\_конференцияTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (checkBox4.Checked)

                {

                    string cmd = "UPDATE Конференции SET Оргвзнос = @Оргвзнос WHERE id\_конференция = @id\_конференция";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Оргвзнос", textBox4.Text);

                    command.Parameters.AddWithValue("id\_конференция", id\_конференцияTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (checkBox5.Checked)

                {

                    string cmd = "UPDATE Конференции SET Дата\_начало = @Дата\_начало WHERE id\_конференция = @id\_конференция";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Дата\_начало", дата\_началоDateTimePicker.Value);

                    command.Parameters.AddWithValue("id\_конференция", id\_конференцияTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                sqlConnection.Close();

                MessageBox.Show("Данные успешно изменени!\n\n", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

                this.Close();

            }

        }

    }

}

Листинг 13. FILE: UpdateAutor.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 Conference

{

    public partial class UpdateAutor : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public UpdateAutor()

        {

            InitializeComponent();

        }

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

            if (!string.IsNullOrEmpty(id\_авторTextBox.Text) && (checkBox1.Checked ||

                checkBox2.Checked || checkBox3.Checked || checkBox4.Checked || checkBox15.Checked ||

                checkBox6.Checked || checkBox7.Checked || checkBox8.Checked || checkBox9.Checked ||

                checkBox10.Checked || checkBox11.Checked || checkBox12.Checked || checkBox13.Checked ||

                checkBox14.Checked))

            {

                //Фамилия

                if (checkBox1.Checked)

                {

                    string cmd = "UPDATE Авторы SET Фамилия=@Фамилия WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

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

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //Имя

                if (checkBox2.Checked)

                {

                    string cmd = "UPDATE Авторы SET Имя=@Имя WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

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

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //Отчество

                if (checkBox3.Checked)

                {

                    string cmd = "UPDATE Авторы SET Отчество=@Отчество WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Отчество", textBox3.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //Адрес

                if (checkBox4.Checked)

                {

                    string cmd = "UPDATE Авторы SET Адрес=@Адрес WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Адрес", textBox4.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //Почтовой индекс

                if (checkBox6.Checked)

                {

                    string cmd = "UPDATE Авторы SET Почтовой\_индекс=@Почтовой\_индекс WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Почтовой\_индекс", textBox5.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //рабочий телефон

                if (checkBox7.Checked)

                {

                    string cmd = "UPDATE Авторы SET Рабочий\_телефон=@Рабочий\_телефон WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Рабочий\_телефон", textBox6.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //Домашний телефон

                if (checkBox8.Checked)

                {

                    string cmd = "UPDATE Авторы SET Домашний\_телефон=@Домашний\_телефон WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Домашний\_телефон", textBox7.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //е-mail

                if (checkBox9.Checked)

                {

                    string cmd = "UPDATE Авторы SET e\_mail=@e\_mail WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("e\_mail", textBox8.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //ученая степень

                if (checkBox10.Checked)

                {

                    string cmd = "UPDATE Авторы SET Ученая\_степень=@Ученая\_степень WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Ученая\_степень", textBox9.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //ученое звание

                if (checkBox11.Checked)

                {

                    string cmd = "UPDATE Авторы SET Ученое\_звание=@Ученое\_звание WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Ученое\_звание", textBox10.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //научное направление

                if (checkBox12.Checked)

                {

                    string cmd = "UPDATE Авторы SET Научное\_направлвние=@Научное\_направлвние WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Научное\_направлвние", textBox11.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //кафедра

                if (checkBox13.Checked)

                {

                    string cmd = "UPDATE Авторы SET Кафедра=@Кафедра WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Кафедра", textBox12.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //место работы

                if (checkBox14.Checked)

                {

                    string cmd = "UPDATE Авторы SET Место\_работы=@Место\_работы WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Место\_работы", textBox13.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                //должность

                if (checkBox15.Checked)

                {

                    string cmd = "UPDATE Авторы SET Должность=@Должность WHERE id\_автор = @id\_автор";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("Должность", textBox14.Text);

                    command.Parameters.AddWithValue("id\_автор", id\_авторTextBox.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

            }

            sqlConnection.Close();

            MessageBox.Show("Данные успешно изменени!\n\n", "Информация", MessageBoxButtons.OK, MessageBoxIcon.Information);

            this.Close();

        }

    }

}

Листинг 14. FILE: DeleteDoklad.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 Conference

{

    public partial class DeleteDoklad : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public DeleteDoklad()

        {

            InitializeComponent();

        }

        //кнопка удаления доклада

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

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

            {

                string cmd = "DELETE FROM Доклады WHERE id\_участник=@id\_участник";

                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                command.Parameters.AddWithValue("id\_участник", textBox1.Text);

                try

                {

                    await command.ExecuteNonQueryAsync();

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

                    this.Close();

                }

                catch (Exception ex)

                {

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

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоле 'id\_участника' должен быть заполнен.", "Ошибка",

                    MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            sqlConnection.Close();

        }

    }

}

Листинг 15. FILE: DeleteAutor.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 Conference

{

    public partial class DeleteAutor : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public DeleteAutor()

        {

            InitializeComponent();

        }

        //кнопка удалить

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

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

            {

                if (checkBox2.Checked)

                {

                    int id\_uchast = 0;

                    string searchId = "SELECT id\_участник, COUNT(id\_участник) " +

                                        "FROM Участники " +

                                        "WHERE  id\_автор= @id\_автор " +

                                        "GROUP BY id\_участник";

                    SqlCommand commandID = new SqlCommand(searchId, sqlConnection);

                    commandID.Parameters.AddWithValue("id\_автор", textBox1.Text);

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

                    try

                    {

                        SqlDataReader sqlDataReader = await commandID.ExecuteReaderAsync();

                        while (sqlDataReader.Read())

                        {

                            data.Add(new string[2]);

                            for (int i = 0; i < 2; i++)

                            {

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

                            }

                        }

                        sqlDataReader.Close();

                        if (data.Count != 0)

                        {

                            id\_uchast = Convert.ToInt32(data[0][0].ToString());

                            string cmd = "DELETE FROM Доклады WHERE id\_участник=@id\_участник";

                            SqlCommand command = new SqlCommand(cmd, sqlConnection);

                            command.Parameters.AddWithValue("id\_участник", id\_uchast);

                            try

                            {

                                await command.ExecuteNonQueryAsync();

                            }

                            catch (Exception ex)

                            {

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

                            }

                            string cmd3 = "DELETE FROM Участники WHERE id\_участник=@id\_участник";

                            SqlCommand command3 = new SqlCommand(cmd3, sqlConnection);

                            command3.Parameters.AddWithValue("id\_участник", id\_uchast);

                            try

                            {

                                await command3.ExecuteNonQueryAsync();

                            }

                            catch (Exception ex)

                            {

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

                            }

                        }

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                string cmd2 = "DELETE FROM Авторы WHERE id\_автор=@id\_автор";

                SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

                command2.Parameters.AddWithValue("id\_автор", textBox1.Text);

                try

                {

                    await command2.ExecuteNonQueryAsync();

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

                    this.Close();

                }

                catch (Exception)

                {

                    MessageBox.Show("Ошибка!\n\nПрежде чем удалить данные участника удалите данные" +

                        "участника, связанного с ним, а также данные его " +

                        "доклада во вкладках 'Участник' и 'Доклады' соответственно " +

                        "или поставьте галочку 'Удалить все данные в базе связанные с этим автором, " +

                        "включая участников и доклады'", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоле 'id\_автор' должен быть заполнен.", "Ошибка",

                    MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            sqlConnection.Close();

        }

    }

}

Листинг 16. FILE: DeleteUchastnik.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 Conference

{

    public partial class DeleteUchastnik : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public DeleteUchastnik()

        {

            InitializeComponent();

        }

        //кнопка удаления участника

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

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

            {

                if (checkBox1.Checked)

                {

                    string cmd = "DELETE FROM Доклады WHERE id\_участник=@id\_участник";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("id\_участник", textBox1.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                string cmd2 = "DELETE FROM Участники WHERE id\_участник=@id\_участник";

                SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

                command2.Parameters.AddWithValue("id\_участник", textBox1.Text);

                try

                {

                    await command2.ExecuteNonQueryAsync();

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

                    this.Close();

                }

                catch (Exception)

                {

                    MessageBox.Show("Ошибка!\n\nПрежде чем удалить данные участника удалите данные его " +

                        "доклада во вкладке 'Доклады' или поставьте галочку 'Также удалить " +

                        "все данные о докладах данного участника'", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоле 'id\_участника' должен быть заполнен.", "Ошибка",

                    MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            sqlConnection.Close();

        }

    }

}

Листинг 17. FILE: DeleteHotel.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 Conference

{

    public partial class DeleteHotel : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public DeleteHotel()

        {

            InitializeComponent();

        }

        //кнопка удаления гостиницы

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

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

            {

                if (updChkBox.Checked)

                {

                    string searchId = "UPDATE Участники SET id\_гостиница = null " +

                                       "WHERE  id\_гостиница= @id\_гостиница";

                    SqlCommand commandID = new SqlCommand(searchId, sqlConnection);

                    commandID.Parameters.AddWithValue("id\_гостиница", textBox1.Text);

                    try

                    {

                        await commandID.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                if (forceDelChkBox.Checked)

                {

                    string cmd = "DELETE FROM Участники WHERE id\_гостиница=@id\_гостиница";

                    SqlCommand command = new SqlCommand(cmd, sqlConnection);

                    command.Parameters.AddWithValue("id\_гостиница", textBox1.Text);

                    try

                    {

                        await command.ExecuteNonQueryAsync();

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                string cmd3 = "DELETE FROM Гостиницы WHERE id\_гостиница=@id\_гостиница";

                SqlCommand command3 = new SqlCommand(cmd3, sqlConnection);

                command3.Parameters.AddWithValue("id\_гостиница", textBox1.Text);

                try

                {

                    await command3.ExecuteNonQueryAsync();

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

                    this.Close();

                }

                catch (Exception ex)

                {

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

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоле 'id\_гостиница' должен быть заполнен.", "Ошибка",

                    MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            sqlConnection.Close();

        }

    }

}

Листинг 18. FILE: DeleteConference.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 Conference

{

    public partial class DeleteConference : Form

    {

        SqlConnection sqlConnection;

        const string connectionString = @"Data Source=(LocalDB)\MSSQLLocalDB;AttachDbFilename=" +

                    @"C:\Users\USER\source\repos\Conference\Conference.mdf;Integrated Security=True";

        public DeleteConference()

        {

            InitializeComponent();

        }

        //кнопка удалить

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

        {

            sqlConnection = new SqlConnection(connectionString);

            await sqlConnection.OpenAsync();

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

            {

                if (checkBox1.Checked)

                {

                    string searchId = "SELECT id\_участник " +

                                        "FROM Участники " +

                                        "WHERE  id\_конференция= @id\_конференция";

                    SqlCommand commandID = new SqlCommand(searchId, sqlConnection);

                    commandID.Parameters.AddWithValue("id\_конференция", textBox1.Text);

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

                    try

                    {

                        SqlDataReader sqlDataReader = await commandID.ExecuteReaderAsync();

                        while (sqlDataReader.Read())

                        {

                            data.Add(new string[1]);

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

                        }

                        sqlDataReader.Close();

                        if (data.Count != 0)

                        {

                            foreach (string[] item in data)

                            {

                                string cmd = "DELETE FROM Доклады WHERE id\_участник=@id\_участник";

                                SqlCommand command = new SqlCommand(cmd, sqlConnection);

                                command.Parameters.AddWithValue("id\_участник", item[0].ToString());

                                try

                                {

                                    await command.ExecuteNonQueryAsync();

                                }

                                catch (Exception ex)

                                {

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

                                }

                                string cmd3 = "DELETE FROM Участники WHERE id\_участник=@id\_участник";

                                SqlCommand command3 = new SqlCommand(cmd3, sqlConnection);

                                command3.Parameters.AddWithValue("id\_участник", item[0].ToString());

                                try

                                {

                                    await command3.ExecuteNonQueryAsync();

                                }

                                catch (Exception ex)

                                {

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

                                }

                            }

                        }

                    }

                    catch (Exception ex)

                    {

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

                    }

                }

                string cmd2 = "DELETE FROM Конференции WHERE id\_конференция=@id\_конференция";

                SqlCommand command2 = new SqlCommand(cmd2, sqlConnection);

                command2.Parameters.AddWithValue("id\_конференция", textBox1.Text);

                try

                {

                    await command2.ExecuteNonQueryAsync();

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

                    this.Close();

                }

                catch (Exception)

                {

                    MessageBox.Show("Ошибка!\n\nПрежде чем удалить данные участника удалите данные" +

                        "участника, связанного с ним, а также данные его " +

                        "доклада во вкладках 'Участник' и 'Доклады' соответственно " +

                        "или поставьте галочку 'Удалить данные всех зарегистрированных учатников и их " +

                        "докладов на эту конференцию'", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

                }

            }

            else

            {

                MessageBox.Show("Ошибка!\n\nПоле 'id\_автор' должен быть заполнен.", "Ошибка",

                    MessageBoxButtons.OK, MessageBoxIcon.Error);

            }

            sqlConnection.Close();

        }

    }

}