СОДЕРЖАНИЕ

ВВЕДЕНИЕ

База данных – это совокупность данных, организованных по определенным правилам и связанных между собой логически. Приложение для работы с базой данных – это программное обеспечение, которое позволяет выполнять различные операции над данными, такие как ввод, хранение, обработка, поиск, анализ и представление. Проектирование базы данных и приложения для работы с базой данных в организации подразумевает под собой процесс анализа потребностей и задач организации, определения структуры и свойств данных, выбора подходящих технологий и инструментов, разработки логической и физической модели базы данных, реализации и тестирования приложения для работы с базой данных.

В современных организациях использование баз данных имеет ряд преимуществ, таких как:

- повышение эффективности и качества управления информацией;
- обеспечение целостности, безопасности и доступности данных;
- упрощение интеграции и обмена данными между различными подразделениями и системами;
- улучшение поддержки принятия решений на основе анализа данных;
- снижение затрат на хранение и обслуживание данных.

Возникновение избыточного количества данных провоцирует сложность ориентации в них, а также увеличивает количество затрачиваемых сил на обработку информации в организациях и предприятиях. Чтобы избежать этого, нужно внедрять работу с автоматизированными базами данных на всех уровнях технологических процессов и работ.

Сегодня многие компании применяют персональные компьютеры для хранения и обработки различных видов информации. Эта информация находится в базах данных. Основные принципы современной информационной методологии основаны на том, что информация должна быть организована в базы данных с целью отражения динамично меняющегося мира и удовлетворения всех информационных потребностей пользователей. Базы данных создаются и функционируют под контролем специальных программных сред, которые называются системами управления базами данных. База данных, которая представлена в объективной форме, это набор следующих материалов: статей, счетов, нормативных актов, судебных решений или других подобных материалов, собранных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронного вычислительного устройства.

База данных – это система технических, информационных, программных и других ресурсов, необходимых для решения различных информационных задач, интересующих пользователя [1].

Для того, чтобы информация в базах данных была достоверной и полезной, она должна соответствовать определенным требованиям. Одним из таких

требований является непротиворечивость данных, то есть отсутствие противоречащих друг другу сведений. Другим требованием является целостность данных, то есть сохранение связей между различными элементами информации. Еще одним требованием является не избыточность данных, то есть исключение повторения одних и тех же сведений в разных частях базы данных. База данных предназначена для того, чтобы хранить и обеспечивать прямой доступ к информации, которая отражает определенную предметную область, то есть сферу деятельности или интереса пользователя. Степень детализации данных зависит от ряда факторов. В первую очередь, от цели использования информации из баз данных и от сложности информационных процессов, которые происходят в рамках предметной области в конкретных условиях.

Система управления базами данных – это программный механизм, предназначенный для записи, поиска, сортировки, обработки и печати информации, содержащейся в базе данных [2].

В компьютере данные базы данных представляется в виде таблицы, схожей на электронную таблицу. Названия столбцов, представляющих заголовки таблицы, называют именами полей, а сами столбцы – полями. Данные, которые находятся в полях, называют значениями полей.

Создание современного программного обеспечения – это весьма трудоемкий процесс, требующий от специалиста представлений о методах анализа, проектирования, реализации и тестирования программного продукта.

В данной курсовой работе поставлена задача базы данных и создания информационной системы кафедры (преподаватели и предметы).

Для создания информационной базы данных будет использоваться СУДБ SQL Server. Для создания приложения – среда Visual Studio 2022.

1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

1.1 Анализ предметной области и выявление необходимого набора сущностей

В качестве идеи для проектирования и разработки была выбрана тема «Кафедра (преподаватели и предметы)», которая предназначена для хранения, обработки и предоставления данных о преподавателях, дисциплинах, группах студентов, нагрузке, обучающих материалах и мероприятиях, связанных с кафедрой. Система содержит следующее описание исходя из выданного задания к курсовой работе: в таблице «Person» находятся данные о фамилии, имени и отчестве преподавателя, сами преподаватели находятся в отдельной таблице и наследуют эти данные из прошлой таблицы. Каждый преподаватель имеет свою должность, может вести разные дисциплины, которые разделяются на типы занятий (лабораторные работы, практические занятия, лекции), курирует одну или несколько групп студентов. За преподавателем закрепляется определенное количество часов нагрузки. В нагрузку входит: количество часов на занятие, специальность, которую содержит в себе группа студентов, у которой ведется дисциплина, а также тип занятия. Пользователем системы является преподаватель, который может только просматривать информацию о закреплениях для своей нагрузки, а также проставлять средний балл для дисциплин у группы.

Также у информационной системы присутствует администратор, который может редактировать, добавлять и удалять данные для всех существующих таблиц. Кроме того, администратор добавляет в информационную систему обучающие материалы, содержащие в себе информацию о дате публикации, авторе и их название. Также присутствует таблица мероприятий, проходящих на факультете, в которой администратор заносит информацию о гостях и участниках данного мероприятия.

Каждая таблица содержит в себе уникальные атрибуты, которые диктуются выбранной темой.

1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов

Определим нужный набор атрибутов для каждой сущности с целью построения инфологической концептуальной модели. Концептуальная модель данных (КМД) – это общая информационная модель предметной области,

охватывающая вопросы классификации, структуризации и семантической целостности (достоверности и согласованности данных).

Атрибутом является поименованная характеристика сущности. Наименования должны быть уникальными для конкретного типа сущности и не совпадать с системными словами, такими как ключевые слова запросов (select, from), но могут быть одинаковыми для различного типа сущностей. Атрибуты используются для определения того, какая информация должна быть собрана о сущности.

В таблице 1.1 представлены сущности, определенные для них атрибуты, описание атрибутов и ключи.

Таблица 1.1 – Описание сущностей

Таблица	Поле	Ключ	Описание
Person	PersonID	PK	Идентификационный номер таблицы TipiPoezda
	FirstName		Имя человека
	LastName		Фамилия человека
	Patronymic		Отчество человека
Admin	FKPersonID	FK	Идентификационный номер таблицы Person
	AdminID	PK	Идентификационный номер таблицы Admin
	Login		Логин для входа в систему в качестве админа
	Password		Пароль для входа в систему в качестве админа
Teacher	FKPersonID	FK	Идентификационный номер таблицы Person
	FKPostID	FK	Идентификационный номер таблицы Post
	TeacherID	PK	Идентификационный номер таблицы Teacher
	Login		Логин для входа в систему в качестве преподавателя
	Password		Пароль для входа в систему в качестве преподавателя

Продолжение таблицы 1.1

Таблица	Поле	Ключ	Описание
Post	PostID	PK	Идентификационный номер таблицы Post
	PostName		Название должности преподавателя
EducationalMaterials	FKAdminID	FK	Идентификационный номер таблицы Admin
	EducationalMaterialsID	PK	Идентификационный номер таблицы EducationalMaterials
	MaterialType		Тип материала
	MaterialName		Название обучающего материала
	MaterialAutor		Фамилия и инициалы автора
	PublicationYear		Дата публикации обучающего материала
Projects	FKAdminID	FK	Идентификационный номер таблицы Admin
	ProjectsID	PK	Идентификационный номер таблицы Projects
	TypeOfProject		Тип проекта
	Status		Существующий статус проекта (завершен, в разработке)
Events	FKAdminID	FK	Идентификационный номер таблицы Admin
	EventsID	PK	Идентификационный номер таблицы Events
	EventType		Название типа мероприятия
	EventName		Название мероприятия
	EventDate		Дата проведения мероприятия
Participants	ParticipantsID	PK	Идентификационный номер таблицы Participants

Продолжение таблицы 1.1

Таблица	Поле	Ключ	Описание
	FirstName		Имя участника мероприятия
	LastName		Фамилия участника мероприятия
	Patronymic		Отчество участника мероприятия
Events_Participants	Events_ParticipantsID	PK	Идентификационный номер таблицы Events_Participants
	FKParticipantsID	FK	Идентификационный номер таблицы Participants
	FKEventsID	FK	Идентификационный номер таблицы Events
Guests	GuestsID	PK	Идентификационный номер таблицы Events
	FirstName		Имя гостя мероприятия
	LastName		Фамилия гостя мероприятия
	Patronymic		Отчество гостя мероприятия
Events_Guests	Events_GuestsID	PK	Идентификационный номер таблицы Events_Guests
	FKGuestsID	FK	Идентификационный номер таблицы Guests
	FKEventsID	FK	Идентификационный номер таблицы Events
Specialization	SpecializationID	PK	Идентификационный номер таблицы Specialization
	SpecializationName		Название специальности
Discipline	DisciplineID	PK	Идентификационный номер таблицы Discipline
	DisciplineName		Название дисциплины

Продолжение таблицы 1.1

Таблица	Поле	Ключ	Описание
TypeWork	TypeWorkID	PK	Идентификационный номер таблицы TypeWork
	TypeWorkName		Тип дисциплины (лекция, практическое занятие, лабораторная работа)
Specialization_Discipline	FKSpecializationID	FK	Идентификационный номер таблицы Specialization
	FKDisciplineID	FK	Идентификационный номер таблицы Discipline
	Specialization_DisciplineID	PK	Идентификационный номер таблицы Specialization_Discipline
TypeWork_Specialization_Discipline	TypeWork_Specialization_DisciplineID	PK	Идентификационный номер таблицы TypeWork_Specialization_Discipline
	FKTypeWorkID	FK	Идентификационный номер таблицы TypeWork
	FKSpecialization_DisciplineID	FK	Идентификационный номер таблицы Specialization_Discipline
TimeManage	FKTeacherID	FK	Идентификационный номер таблицы Teacher
	FKTypeWork_Specialization_DisciplineID	FK	Идентификационный номер таблицы TypeWork_Specialization_Discipline
	TimeManageID	PK	Идентификационный номер таблицы TimeManage

Продолжение таблицы 1.1

Таблица	Поле	Ключ	Описание
	AverageTime		Количество часов, закрепленных за преподавателем на дисциплину
	FKGroupID	FK	Идентификационный номер таблицы Group
SupervisedGroup	FKTeacherID	FK	Идентификационный номер таблицы Teacher
	FKSpecializationID	FK	Идентификационный номер таблицы Specialization
	SupervisedGroupID	PK	Идентификационный номер таблицы SupervisedGroup
	GroupName		Название группы
	StudentsCount		Количество студентов в группе
Grade	FKSupervisedGroupID	FK	Идентификационный номер таблицы SupervisedGroup
	FKDisciplineID	FK	Идентификационный номер таблицы Discipline
	FKTypeWorkID	FK	Идентификационный номер таблицы TypeWork
	GradeID	PK	Идентификационный номер таблицы Grade
	AverageRating		Средний балл

1.3 Определение связей между объектами

Между двумя типами сущностей может быть установлена связь, которая характеризуется глаголом, применяемым для взаимодействия между ними. Связь создается с помощью внешних ключей, которые являются атрибутами или наборами атрибутов, ссылающихся на первичный ключ или уникальный ключ другой таблицы.

Связи позволяют находить другие сущности, связанные с одной сущностью. Каждая связь имеет два конца и одно или два наименования, которые обычно выражаются в неопределенной глагольной форме, например, «иметь» или «принадлежать». Наименования относятся к своему концу связи.

Связи делятся на многие ко многим, один ко многим и один к одному. В связи многие ко многим нам нужен посредник между двумя таблицами, который должен хранить два внешних ключа, первый из которых ссылается на первую таблицу, а второй – на вторую. В связи один ко многим сущность является общей для всех остальных, но может связываться с каждой сущностью не более одного раза. Примером может служить наличие у пользователя двух мобильных устройств, каждое из которых имеет поле владельца, а в нем одного и того же пользователя. В связи один к одному отличительной особенностью является уникальность значения в столбце, который относится ко второй сущности. Примером может служить голосование, где голос можно отдать только один раз, так как вас отмечают по уникальному полю: ваше удостоверение. Мы можем наложить на атрибут ограничение «primary key», которое отличается от ограничения «unique» лишь тем, что не может принимать значения «null».

Для реализации информационной системы станции необходимо установить все связи между объектами. Для этого нужно рассмотреть всю информационную систему в совокупности и определить отношения объектов, составляющих систему.

Проследить отношения, в которых состоят таблицы базы данных можно по схеме, изображенной на рисунке А.1 приложения А.

1.4 Описание полученной модели на языке инфологического проектирования

Проектирование инфологической модели предметной области – это процесс создания схемы базы данных, описывающей объекты предметной области в терминах некоторой семантической модели, например, в терминах ER-модели.

Концептуальное проектирование – это построение семантической модели предметной области, которая создается без ориентации на какую-либо конкретную СУБД и модель данных.

Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контексте равноправно могут использоваться слова «модель базы данных» и «модель предметной области», поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности. Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для

этого формальным аппаратом. Обычно используются графические нотации, подобные ER-диаграммам.

Как пример, представим, по условию требуется создать базу данных для учета студентов и курсов в университете. Можно создать таблицы для студентов и курсов, а затем связать их между собой. В этом случае, студенты будут иметь уникальный идентификатор, который будет использоваться для связи с курсами, которые они посещают. Это и есть пример инфологической модели предметной области.

Концептуальная модель базы данных включает в себя описание информационных объектов или понятий предметной области и связей между ними, а также описание ограничений целостности, то есть требований к допустимым значениям данных и к связям между ними.

То есть термины «семантическая модель», «концептуальная модель» и «инфологическая модель» используются для описания процесса создания схемы базы данных, описывающей объекты предметной области в терминах некоторой семантической модели. Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для этого формальным аппаратом.

Концептуальная модель проектируемой базы данных представлена на рисунке А.1 приложения А.

2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

2.1 Построение набора необходимых отношений базы данных

Для проектирования схемы реляционной базы данных требуется определить совокупность отношений, составляющих базу данных. Данная совокупность отношений должна содержать всю информацию, хранящуюся в базе данных.

На основе полученной концептуальной модели следует определить набор необходимых отношений базы данных. На рисунке 2.1 представлены отношения базы данных. Конец связи, отмеченный ключом, указывает на то, что сущность, связанная с этим концом, является главной по отношению к сущности, связанной с противоположным концом. В случае, если все отношения между сущностями установлены в виде «один ко многим», конец связи с ключом означает, что эта сущность используется несколько раз в зависимой сущности.

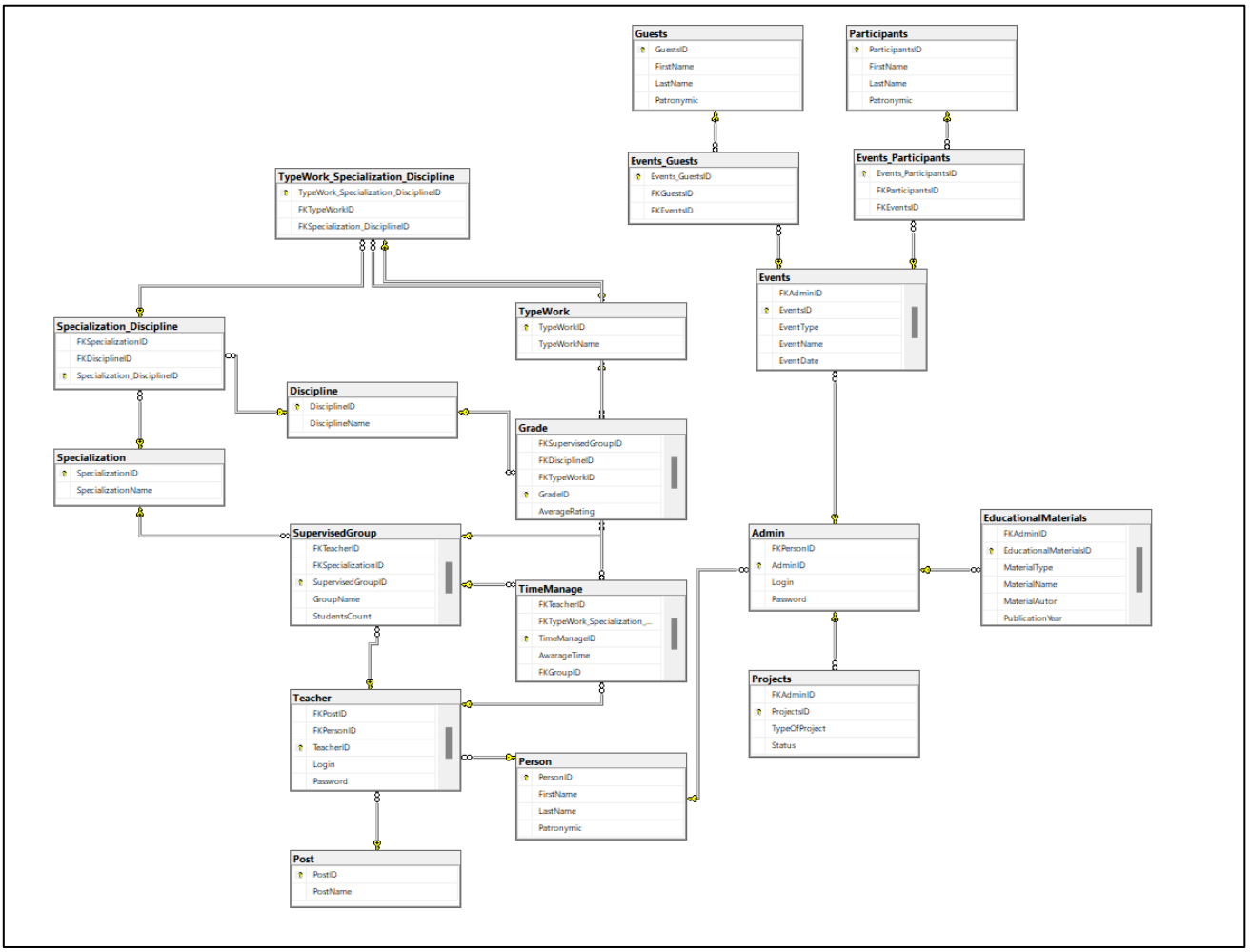

Рисунок 2.1 – Набор необходимых отношений базы данных

2.2 Задание первичных и внешних ключей определенных отношений

В каждой таблице базы данных может существовать первичный ключ. Основными требования к первичным ключам являются уникальность и минимальность [3].

Минимальность рассматривается в двух аспектах: первым является объем памяти, который предназначен для хранения значений атрибутов, входящих в первичный ключ. Во втором случае под минимальностью первичного ключа подразумевается отсутствие в составе ключа атрибута, значения которого уникальны.

В связанных таблицах первичный ключ родительской таблицы становится внешним ключом в дочерней таблице. Внешний ключ дочерней таблицы отсылает к сведениям родительской таблицы.

Первичный ключ – это уникальный идентификатор записи в реляционной модели данных. Он также используется для связи таблиц. В связанных таблицах первичный ключ родительской таблицы становится внешним ключом в дочерней таблице. Внешний ключ дочерней таблицы ссылается на сведения родительской таблицы.

Если в отношении имеется единственный потенциальный ключ, он является первичным ключом. Если потенциальных ключей несколько, один из них выбирается в качестве первичного, а другие называются «альтернативными».

Чтобы правильно выбрать первичный ключ, следует учитывать его основные характеристики. Во-первых, он однозначно определяет каждую строку. Во-вторых, в нем нет пустых или отсутствующих значений – он всегда содержит значение. В-третьих, он никогда не меняется или меняется, но крайне редко.

В качестве первичного ключа не рекомендуется выбирать фамилию или адрес, поскольку такие данные со временем могут измениться.

Всегда следует определять для таблицы первичный ключ. Для первичного ключа автоматически создается индекс, ускоряющий выполнение запросов и операций.

Внешний ключ представляет собой подмножество атрибутов некоторой переменной отношения «R2», значения которых должны совпадать со значениями некоторого потенциального ключа некоторой переменной отношения «R1 Внешний ключ, также как и потенциальный, может быть простым и составным. Поддержка внешних ключей также называется соблюдением ссылочной целостности. Реляционные СУБД поддерживают автоматический контроль ссылочной целостности.

Хотя каждое значение внешнего ключа обязано совпадать со значениями потенциального ключа в некотором кортеже родительского отношения, то обратное, вообще говоря, неверно. Например, могут существовать поставщики, не поставляющие никаких деталей. Для внешнего ключа не требуется, чтобы он был компонентом некоторого потенциального ключа (как получилось в примере с поставщиками и деталями).

Null-значения для атрибутов внешнего ключа допустимы только в том случае, когда атрибуты внешнего ключа не входят в состав никакого потенциального ключа. Т.к. внешние ключи фактически служат ссылками на кортежи в другом (или в том же самом) отношении, то эти ссылки не должны указывать на несуществующие объекты. Это определяет следующее правило целостности внешних ключей:

Правило целостности внешних ключей. Внешние ключи не должны быть несогласованными, т.е. для каждого значения внешнего ключа должно существовать соответствующее значение первичного ключа в родительском отношении [4].

Первичные ключи имеют постфикс PK, вторичные – FK. Следует упомянуть, что в контексте данной работы первичные ключи будут являться единственным полем в таблице.

Первичные и вторичные ключи представлены в таблице 1.1.

В дальнейшем построении схемы реляционной базы данных ключи будут служить для организации связей между отношениями.

2.3 Третья нормальная форма

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц) [5].

Третья нормальная форма (3NF) – это метод проектирования баз данных, который используется для уменьшения дублирования, избежания аномалий данных, обеспечения целостности данных и упрощения управления данными. Она требует, чтобы все атрибуты были функционально зависимы от первичного ключа или кандидатских ключей.

Для того, чтобы таблица соответствовала третьей нормальной форме, она должна удовлетворять следующим условиям:

– Она должна соответствовать второй нормальной форме;

– никакой не первичный атрибут не должен транзитивно зависеть от первичного ключа.

Третья нормальная форма является важным шагом в проектировании баз данных, поскольку она помогает уменьшить дублирование данных, избежать аномалий данных и обеспечить целостность данных. Это также упрощает управление данными и повышает производительность базы данных.

Первая нормальная форма: отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Повторяющимися являются поля, содержащие одинаковые по смыслу значения. Например, если требуется получить статистику сдачи экзаменов по

предметам, можно создать поля для хранения данных об оценке по каждому предмету. Однако в этом случае мы имеем дело с повторяющимися группами.

Вторая нормальная форма: отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа (ПК). Третья нормальная форма требует, чтобы значение любого поля таблицы, не входящего в первичный ключ, не зависело от значения другого поля, не входящего в первичный ключ.

Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.

Третья нормальная форма: отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Берем во внимание то, что все атрибуты наших отношений атомарны, что каждое отношение имеет первичный ключ, что в отношениях базы отсутствуют зависимости неключевых атрибутов от других неключевых атрибутов, а также отсутствуют зависимости неключевых атрибутов от части составного ключа, и делаем вывод, что отношения базы находятся в третьей нормальной форме.

2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом

Ограничения целостности можно определить как специальные средства в базах данных, главное назначение которых – не дать попасть в базу недопустимым данным.

Ограничения целостности в базах данных – это специальные средства, которые предназначены для предотвращения попадания в базу данных недопустимых данных. Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. То есть любая запись отношения должна быть отлична от любой другой записи этого же отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

Для удовлетворения требования ограничения целостности для внешних ключей отношений и для отношений в целом необходимо, чтобы выполнялось соответствие между типами вводимых данных и типами столбцов в таблицах, чтобы были заполнены все обязательные поля в таблицах, то есть те поля, которые не могут содержать значения NULL. Для автоматического обновления связанных полей (удаления записей) при обновлении (удалении) в главной таблице, следует

устанавливать обеспечение целостности данных и каскадное обновление связанных полей (каскадное удаление связанных записей).

Ограничения целостности – это правила, которые ограничивают все возможные состояния базы данных, а также переходы с одного стана в другой. Ограничения целостности для внешних ключей отношений и для отношений в целом определяют, какие значения могут быть вставлены в таблицы базы данных. Ограничения целостности для внешних ключей гарантируют, что значения внешнего ключа всегда существуют в таблице, на которую он ссылается. Ограничения целостности для отношений в целом гарантируют, что значения в таблице уникальны и не дублируются.

Для того, чтобы автоматически обновлять связанные поля (удалять записи) при обновлении (удалении) в главной таблице, необходимо установить обеспечение целостности данных и каскадное обновление связанных полей (каскадное удаление связанных записей).

Для удовлетворения требования ограничения целостности для внешних ключей отношений и для отношений в целом необходимо, чтобы выполнялось соответствие между типами вводимых данных и типами столбцов в таблицах, а также чтобы были заполнены все обязательные поля в таблицах, то есть те поля, которые не могут содержать значения NULL. Система управления базами данных не может контролировать правильность каждого отдельного значения, вводимого в базу данных. Для этого существует ряд средств, помогающих разработчику минимизировать возможность нарушения целостности данных базы: триггеры, проверки, уникальность и другое.

2.5 Графическое представление связей между внешними первичными ключами

В ходе нормализации и определении первичных ключей, внешних ключей и связей между сущностями, была разработана схема реляционной базы данных, которая представлена на рисунке Б.1 приложения Б. На данной схеме изображаются все отношения базы данных, а также связи между внешними и первичными ключами.

3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Благодаря системе управления базами данных SQL Server 2018 была реализована спроектированная ранее база данных станции. Данная система была выбрана по ряду весомых причин: широкое распространение, наличие свободно распространяемых сборок, наличие высококачественных программных средств разработки, позволяющих создавать разного вида приложения, которые, в свою очередь, смогут использовать базы данных Microsoft SQL Server.

Описание структур каждой из таблиц базы данных с описанием типа полей представлено в таблицах ниже.

Таблица 3.1 – Характеристики атрибутов таблицы Person

Поле	Тип данных	Описание
PersonID	INT	Идентификационный номер таблицы Person
FirstName	NVARCHAR(50)	Имя человека
LastName	NVARCHAR(50)	Фамилия человека
Patronymic	NVARCHAR(50)	Отчество человека

Таблица 3.2 – Характеристики атрибутов таблицы Admin

Поле	Тип данных	Описание
FKPersonID	INT	Идентификационный номер таблицы Person
AdminID	INT	Идентификационный номер таблицы Admin
Login	NVARChar(50)	Логин для входа в систему в качестве админа
Password	NVARChar(50)	Пароль для входа в систему в качестве админа

Таблица 3.3 – Характеристики атрибутов таблицы Teacher

Поле	Тип данных	Описание
FKPersonID	INT	Идентификационный номер таблицы Person
FKPostID	INT	Идентификационный номер таблицы Post
TeacherID	INT	Идентификационный номер таблицы Teacher
Login	NVARChar(50)	Логин для входа в систему в качестве преподавателя
Password	NVARChar(50)	Пароль для входа в систему в качестве преподавателя

Таблица 3.4 – Характеристики атрибутов таблицы Post

Поле	Тип данных	Описание
PostID	INT	Идентификационный номер таблицы Post
PostName	NVARCHAR(50)	Название должности преподавателя

Таблица 3.5 – Характеристики атрибутов таблицы EducationalMaterials

Поле	Тип данных	Описание
FKAdminID	INT	Идентификационный номер таблицы Admin
EducationalMaterialsID	INT	Идентификационный номер таблицы EducationalMaterials
MaterialType	NVARChar(MAX)	Тип материала
MaterialName	NVARChar(MAX)	Название обучающего материала
MaterialAutor	NVARChar(MAX)	Фамилия и инициалы автора
PublicationYear	DATE	Дата публикации обучающего материала

Таблица 3.6 – Характеристики атрибутов таблицы Projects

Поле	Тип данных	Описание
FKAdminID	INT	Идентификационный номер таблицы Admin
ProjectsID	INT	Идентификационный номер таблицы Projects
TypeOfProject	NVARChar(MAX)	Тип проекта
Status	NVARCHAR(50)	Существующий статус проекта (завершен, в разработке)

Таблица 3.7 – Характеристики атрибутов таблицы Events

Поле	Тип данных	Описание
FKAdminID	INT	Идентификационный номер таблицы Admin
EventsID	INT	Идентификационный номер таблицы Events
EventType	NVARCHAR(50)	Название типа мероприятия
EventName	NVARChar(MAX)	Название мероприятия
EventDate	Date	Дата проведения мероприятия

Таблица 3.8 – Характеристики атрибутов таблицы Participants

Поле	Тип данных	Описание
ParticipantsID	INT	Идентификационный номер таблицы Participants
FirstName	NVARChar(50)	Имя участника мероприятия
LastName	NVARChar(50)	Фамилия участника мероприятия
Patronymic	NVARChar(50)	Отчество участника мероприятия

Таблица 3.9 – Характеристики атрибутов таблицы Events_Participants

Поле	Тип данных	Описание
Events_ParticipantsID	INT	Идентификационный номер таблицы Events_Participants

Продолжение таблицы 3.9

Поле	Тип данных	Описание
FKParticipantsID	INT	Идентификационный номер таблицы Participants
FKEventsID	INT	Идентификационный номер таблицы Events

Таблица 3.10 – Характеристики атрибутов таблицы Guests

Поле	Тип данных	Описание
GuestsID	INT	Идентификационный номер таблицы Events
FirstName	NVARChar(50)	Имя гостя мероприятия
LastName	NVARChar(50)	Фамилия гостя мероприятия
Patronymic	NVARChar(50)	Отчество гостя мероприятия

Таблица 3.11 – Характеристики атрибутов таблицы Events_Guests

Поле	Тип данных	Описание
Events_GuestsID	INT	Идентификационный номер таблицы Events_Guests
FKGuestsID	INT	Идентификационный номер таблицы Guests
FKEventsID	INT	Идентификационный номер таблицы Events

Таблица 3.12 – Характеристики атрибутов таблицы Specialization

Поле	Тип данных	Описание
SpecializationID	INT	Идентификационный номер таблицы Specialization
SpecializationName	NVARChar(50)	Название специальности

Таблица 3.13 – Характеристики атрибутов таблицы Discipline

Поле	Тип данных	Описание
DisciplineID	INT	Идентификационный номер таблицы Discipline
DisciplineName	NVARChar(MAX)	Название дисциплины

Таблица 3.14 – Характеристики атрибутов таблицы TypeWork

Поле	Тип данных	Описание
TypeWorkID	INT	Идентификационный номер таблицы TypeWork
TypeWorkName	NVARChar(50)	Тип дисциплины (лекция, практическое занятие, лабораторная работа)

Таблица 3.15 – Характеристики атрибутов таблицы Specialization_Discipline

Поле	Тип данных	Описание
FKSpecializationID	INT	Идентификационный номер таблицы Specialization
FKDisciplineID	INT	Идентификационный номер таблицы Discipline
Specialization_DisciplineID	INT	Идентификационный номер таблицы Specialization_Discipline

Таблица 3.16 – Характеристики атрибутов таблицы TypeWork_Specialization_Discipline

Поле	Тип данных	Описание
TypeWork_Specialization_DisciplineID	INT	Идентификационный номер таблицы TypeWork_Specialization_Discipline
FKTypeWorkID	INT	Идентификационный номер таблицы TypeWork
FKSpecialization_DisciplineID	INT	Идентификационный номер таблицы Specialization_Discipline

Таблица 3.17 – Характеристики атрибутов таблицы TimeManage

Поле	Тип данных	Описание
FKTeacherID	INT	Идентификационный номер таблицы Teacher
FKTypeWork_Specialization_DisciplineID	INT	Идентификационный номер таблицы TypeWork_Specialization_Discipline
TimeManageID	INT	Идентификационный номер таблицы TimeManage
AverageTime	INT	Количество часов, закрепленных за преподавателем на дисциплину
FKGroupID	INT	Идентификационный номер таблицы Group

Таблица 3.18 – Характеристики атрибутов таблицы SupervisedGroup

Поле	Тип данных	Описание
FKTeacherID	INT	Идентификационный номер таблицы Teacher
FKSpecializationID	INT	Идентификационный номер таблицы Specialization
SupervisedGroupID	INT	Идентификационный номер таблицы SupervisedGroup
GroupName	NVARChar(50)	Название группы
StudentsCount	INT	Количество студентов в группе

Таблица 3.19 – Характеристики атрибутов таблицы Grade

Поле	Тип данных	Описание
FKSupervisedGroupID	INT	Идентификационный номер таблицы SupervisedGroup
FKDisciplineID	INT	Идентификационный номер таблицы Discipline
FKTypeWorkID	INT	Идентификационный номер таблицы TypeWork
GradeID	INT	Идентификационный номер таблицы Grade
AverageRating	float	Средний балл

4 ЗАПИСЬ ВЫРАЖЕНИЙ, УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL

Описание запросов и их реализация указаны в названии листингов и их содержании ниже.

Листинг 4.1 – Запрос, возвращающий значение общего количества студентов на факультете

```
SELECT SUM(StudentsCount) as TotalStudents
FROM SupervisedGroup;
```

Листинг 4.2 – Запрос, возвращающий информацию о дисциплинах и связанных с ними группах, которые преподает учитель с заданным идентификатором

```
CREATE PROCEDURE GetTeacherDisciplines @TeacherID INT AS
BEGIN
    SELECT
        TWSD.TypeWork_Specialization_DisciplineID,
        D.DisciplineName,
        TW.TypeWorkName,
        S.SpecializationName,
        TM.AwarageTime,
        SG.GroupName,
        TM.FKGroupID AS GroupID
    FROM TimeManage TM
    INNER JOIN Teacher T ON TM.FKTeacherID = T.TeacherID
    INNER JOIN TypeWork_Specialization_Discipline TWSD ON
TM.FKTypeWork_Specialization_DisciplineID =
TWSD.TypeWork_Specialization_DisciplineID
    INNER JOIN TypeWork TW ON TWSD.FKTypeWorkID = TW.TypeWorkID
    INNER JOIN Specialization_Discipline SD ON
TWSD.FKSpecialization_DisciplineID = SD.Specialization_DisciplineID
    INNER JOIN Discipline D ON SD.FKDisciplineID = D.DisciplineID
    INNER JOIN Specialization S ON SD.FKSpecializationID =
S.SpecializationID
    INNER JOIN SupervisedGroup SG ON TM.FKGroupID =
SG.SupervisedGroupID
    WHERE T.TeacherID = @TeacherID
END
```

Листинг 4.3 – Запрос, возвращающий полную информацию о типах работ, дисциплинах и специализациях, связанных с ними

```
CREATE PROCEDURE GetFullInfoForTypeWork_Specialization_Discipline
AS
BEGIN
    SELECT
        TWSD.TypeWork_Specialization_DisciplineID,
        TW.TypeWorkName,
        D.DisciplineName,
        S.SpecializationName,
```

Продолжение листинга 4.3

```
        CONCAT(S.SpecializationName, ', ',TW.TypeWorkName , ', ',
D.DisciplineName ) AS FullInfo
    FROM TypeWork_Specialization_Discipline TWSD
    INNER JOIN TypeWork TW ON TWSD.FKTypeWorkID = TW.TypeWorkID
    INNER JOIN Specialization_Discipline SD ON
TWSD.FKSpecialization_DisciplineID = SD.Specialization_DisciplineID
    INNER JOIN Discipline D ON SD.FKDisciplineID = D.DisciplineID
    INNER JOIN Specialization S ON SD.FKSpecializationID =
S.SpecializationID
END
```

Листинг 4.4 – Запрос, возвращающий полную информацию о группах, которые курируются определенным преподавателем

```
CREATE PROCEDURE GetGroupsByTeacher
    @TeacherID INT
AS
BEGIN
    SELECT SG.SupervisedGroupID, SG.GroupName, SG.StudentsCount
    FROM SupervisedGroup SG
    WHERE SG.FKTeacherID = @TeacherID;
END;
```

Листинг 4.5 – Запрос, возвращающий названия дисциплин и типов работ, связанных с заданной специализацией

```
CREATE PROCEDURE GetDisciplinesAndTypeWorksForSpecialization
    @FKSpecializationID INT
AS
BEGIN
    SELECT
        SD.FKSpecializationID,
        SD.FKDisciplineID,
        TWSD.FKTypeWorkID,
        D.DisciplineName,
        TW.TypeWorkName
    FROM
        Specialization_Discipline SD
        JOIN Discipline D ON SD.FKDisciplineID = D.DisciplineID
        JOIN     TypeWork_Specialization_Discipline     TWSD     ON
SD.Specialization_DisciplineID = TWSD.FKSpecialization_DisciplineID
        JOIN TypeWork TW ON TWSD.FKTypeWorkID = TW.TypeWorkID
    WHERE
        SD.FKSpecializationID = @FKSpecializationID
END
GO
```

Листинг 4.6 – Запрос, возвращающий полную информацию о группе студентов заданным идентификатором

```
CREATE PROCEDURE GetGroupInfo
    @GroupID INT
AS
```

Продолжение листинга 4.6

```
BEGIN
    SELECT
        g.SupervisedGroupID,
        g.GroupName,
        g.StudentsCount,
        g.FKTeacherID,
        t.Login AS TeacherLogin,
        p.FirstName,
        p.LastName,
        p.Patronymic,
        g.FKSpecializationID,
        s.SpecializationName
    FROM
        SupervisedGroup g
    INNER JOIN
        Teacher t ON g.FKTeacherID = t.TeacherID
    INNER JOIN
        Person p ON t.FKPersonID = p.PersonID
    INNER JOIN
        Specialization s ON g.FKSpecializationID =
s.SpecializationID
    WHERE
        g.SupervisedGroupID = @GroupID
END
```

Листинг 4.7 – Запрос, возвращающий полную информацию о всех админах, зарегистрированных в базе данных

```
CREATE PROCEDURE GetAdmins
AS
BEGIN
    SELECT A.AdminID, A.Login, A.Password, P.FirstName, P.LastName,
P.Patronymic
    FROM Admin A
    INNER JOIN Person P ON A.FKPersonID = P.PersonID;
END
GO
```

Листинг 4.8 – Запрос, возвращающий информацию об участниках и событиях, в которых они принимали, либо будут принимать участие

```
CREATE FUNCTION GetParticipantEvents()
RETURNS TABLE
AS
RETURN
(
    SELECT EP.Events_ParticipantsID, P.LastName, P.FirstName,
P.Patronymic, E.EventName, E.EventDate
    FROM Participants P
    INNER JOIN Events_Participants EP ON P.ParticipantsID =
EP.FKParticipantsID
    INNER JOIN Events E ON EP.FKEventsID = E.EventsID
)
```

Листинг 4.9 – Запрос, возвращающий информацию об оценках студентов по группам, дисциплинам и типам работ

```
CREATE PROCEDURE GetGradesByGroupID
    @GroupID INT
AS
BEGIN
    SELECT
        g.GradeID,
        g.FKSupervisedGroupID,
        g.FKDisciplineID,
        d.DisciplineName,
        g.FKTypeWorkID,
        tw.TypeWorkName,
        g.AverageRating
    FROM
        Grade g
    JOIN
        Discipline d ON g.FKDisciplineID = d.DisciplineID
    JOIN
        TypeWork tw ON g.FKTypeWorkID = tw.TypeWorkID
    WHERE
        g.FKSupervisedGroupID = @GroupID
END
```

Листинг 4.10 – Запрос, принимающий год в качестве параметра и возвращающий все обучающие материалы, выпущенные после указанного года

```
CREATE PROCEDURE GetMaterialsAfterYear @Year INT
AS
BEGIN
    SELECT * FROM EducationalMaterials
    WHERE YEAR(PublicationYear) > @Year
END
```

Листинг 4.11 – Запрос, принимающий дату в качестве параметра и возвращающий список гостей, которые посетили мероприятие после указанной даты

```
CREATE PROCEDURE GetGuestsAfterDate
    @date DATE
AS
BEGIN
    SELECT g.FirstName, g.LastName, g.Patronymic
    FROM Guests g
    JOIN Events_Guests eg ON g.GuestsID = eg.FKGuestsID
    JOIN Events e ON eg.FKEventsID = e.EventsID
    WHERE e.EventDate > @date;
END;
```

Листинг 4.13 – Запрос, возвращающий общее количество студентов на факультете

```
SELECT SUM(StudentsCount) as TotalStudents
FROM SupervisedGroup;
```

Листинг 4.14 – Запрос, возвращающий все проекты со статусом «В разработке»

```
SELECT * FROM Projects WHERE Status = 'В разработке';
```

Листинг 4.15 – Запрос, возвращающий всех преподавателей с заданным именем

```
DECLARE @name NVARCHAR(50) = 'Павел';
SELECT P.FirstName, P.LastName, P.Patronymic
FROM Person P
INNER JOIN Teacher T ON P.PersonID = T.FKPersonID
WHERE P.FirstName = @name;
```

5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Для реализации приложения была выбрана среда разработки Microsoft Visual Studio 2022, в качестве языка программирования – C#.

Существует множество причин, почему WPF приложение может быть полезно для работы с базой данных SQL. Например, WPF приложения могут обеспечить более удобный и интуитивно понятный пользовательский интерфейс, чем традиционные консольные приложения. Кроме того, WPF приложения позволяют более удобно настраивать и использовать множество графических элементов, таких как кнопки, поля ввода, таблицы и т.д., что делает их более удобными для работы с данными.

C# является одним из наиболее популярных языков программирования, используемых для работы с базами данных SQL. Он имеет множество инструментов и библиотек, которые облегчают работу с данными. Кроме того, C# позволяет создавать объектно-ориентированный код, что делает его более удобным для разработки крупных проектов.

Для объединения базы данных с интерфейсом приложения использовался SqlClient.

SQL-клиент – это приложение, которое позволяет подключаться к базе данных SQL и выполнять запросы к ней. SQL-клиенты могут быть использованы для выполнения различных задач, таких как создание таблиц, добавление данных, изменение данных, удаление данных.

Для работы был использован Microsoft SQL Server.

Microsoft SQL Server – это мощная система управления базами данных, которая имеет множество преимуществ. Некоторые из них:

– Безопасность данных: Microsoft SQL Server обеспечивает безопасность вашей базы данных, используя различные методы, такие как шифрование данных, контроль доступа и т.д. Это делает его идеальным выбором для хранения конфиденциальной информации.

– Простота настройки: установка и настройка Microsoft SQL Server проще, чем у других систем управления базами данных. Это позволяет быстро начать работу с базой данных и сосредоточиться на разработке приложения.

– Масштабируемость: Microsoft SQL Server может масштабироваться в зависимости от потребностей вашего бизнеса. Вы можете легко добавлять новые серверы и увеличивать объем данных, которые вы храните.

– Поддержка: Microsoft SQL Server имеет большое сообщество пользователей и разработчиков. Кроме того, Microsoft предоставляет широкий спектр инструментов и ресурсов для разработчиков, которые помогут создавать приложения, использующие базу данных SQL.

6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ

6.1 Разработка и построение интерфейса главной и рабочих форм

Главная страница представлена в виде страницы с возможностью авторизации пользователя как администратора системы, если его логин и пароль соответствуют значениям из таблицы в базе данных. В случае соответствия логина и пароля в таблице преподавателей, пользователь будет авторизирован как преподаватель и попадает в соответствующее окно, демонстрирующее возможности системы с ограниченным функционалом, нужным преподавателю.

При авторизации как администратор, пользователь попадает в окно работы с данными системы, включающими добавление, редактирование и удаление данных во всех таблицах, с возможностью создавать взаимосвязи между таблицами, а также закреплять за преподавателями их нагрузку.

Окно работы с информационной системой у преподавателя имеет ограниченный функционал: просматривать нагрузку за конкретного преподавателя, который авторизировался в системе, а также возможность выставления среднего балла за дисциплину у группы.

Все основные формы и виды выполнены в отдельных страницах программы с собственным функционалом и с использованием интерактивных полей ввода для взаимодействия с пользователем.

Скриншоты, а также описание работы главной и рабочих окон представлены в приложении Г.

Для создания интерфейса мы используем WPF, которое значительно облегчает разработку десктопного приложения, благодаря своему встроенному графическому редактору.

Графический редактор позволяет в графическом виде представить создаваемую форму и в принципе упрощает работу с графическими компонентами. Кроме того, можно легко вносить изменения в графическую составляющую при помощи XAML-кода, который связан с окном и представляет собой partial-класс, что позволяет компилировать одну функциональность из двух частей кода.

Visual Studio имеет еще одну связанную функциональность. Она обладает панелью графических инструментов. И мы можем, вместо создания элементов управления в коде C#, просто переносить их на форму с панели инструментов с помощь мыши.

6.2 Построение главного меню и кнопок панели инструментов

Навигационная панель программы представлена пунктами: Учебные материалы, мероприятия, проекты, преподаватели, дисциплины, группы, и окно «Другое». Часть функционала становится доступной после нажатия на одну из этих кнопок.

6.3 Выполнение программного кода на Microsoft Visual C#

Далее следует описание работы программы с базой данных. Все необходимые методы для закрепления нагрузки и кураторства за преподавателем находятся в классе TeacherPage. Методика закрепления кураторства за преподавателем реализована по аналогичной структуре, как и закрепление нагрузки. Код класса TeacherPage представлен в листинге 6.1.

Листинг 6.1 – Методы для работы с базой данных

```
public partial class TeacherPage : Page
 {

     private int _teacherId;
     private int GroupID;

     public TeacherPage()
     {
         InitializeComponent();
         Update();

         _teachersGrid.LayoutUpdated += DataGrid_LayoutUpdated;
     }

     private void Update()
     {
         FillDataGrid();
         FillTeachersComboBox();
         FillSupervisedGroupComboBox();
     }

     #region AddTeacher
     private void DataGrid_LayoutUpdated(object sender, EventArgs e)
     {

         foreach (var column in _teachersGrid.Columns)
         {
             if    (column.Header.ToString()    ==    "PersonID"    ||
column.Header.ToString() == "TeacherID" || column.Header.ToString()
== "PostID")
```

```
            {
                column.Visibility = Visibility.Hidden;
            }
        }

    }

    public void FillDataGrid()
    {
        using (SqlConnection connection = new
SqlConnection(SQLConnection.connectionString))
        {
            connection.Open();
            SqlCommand command = new SqlCommand("GetTeacherInfo",
connection);
            command.CommandType = CommandType.StoredProcedure;
            SqlDataAdapter dataAdapter = new
SqlDataAdapter(command);
            DataTable dataTable = new DataTable("TeacherInfo");
            dataAdapter.Fill(dataTable);
            _teachersGrid.ItemsSource = dataTable.DefaultView;
            dataAdapter.Update(dataTable);
        }
    }

    private void AddTeacherButton_Click(object sender,
RoutedEventArgs e)
    {
        AddTeacherWindow AddTeacherWindow = new AddTeacherWindow();
        AddTeacherWindow.ShowDialog();
        Update();

        AddTeacherWindow.Closed += UpdateOnClose;
    }

    private void UpdateOnClose(object sender, EventArgs e)
    {
        Update();
    }

    private void DeleteTeacherButton_Click(object sender,
RoutedEventArgs e)
    {

        MessageBoxResult result = MessageBox.Show("Вы уверены, что
хотите удалить эту запись?", "Подтверждение удаления",
MessageBoxButton.YesNo);
        if (result == MessageBoxResult.No)
        {
            return;
```

```
        }

        if (_teachersGrid.SelectedItem != null)
        {
            DataRowView                    row                    =
(DataRowView)_teachersGrid.SelectedItem;
            int teacherId = Convert.ToInt32(row["TeacherID"]);

            using     (SqlConnection     connection     =     new
SqlConnection(SQLConnection.connectionString))
            {
                connection.Open();
                SqlCommand          command          =          new
SqlCommand("DeleteTeacher", connection);
                command.CommandType = CommandType.StoredProcedure;
                command.Parameters.AddWithValue("@TeacherID",
teacherId);
                command.ExecuteNonQuery();
            }

            FillDataGrid();
            Update();
        }
    }


    private     void     EditTeacherButton_Click(object     sender,
RoutedEventArgs e)
    {
        if (_teachersGrid.SelectedItem != null)
        {
            DataRowView                selectedRow                =
(DataRowView)_teachersGrid.SelectedItem;
            EditTeacherWindow     editTeacherWindow     =     new
EditTeacherWindow(selectedRow);
            if (editTeacherWindow.ShowDialog() == true)
            {
                using    (SqlConnection    connection    =    new
SqlConnection(SQLConnection.connectionString))
                {
                    connection.Open();
                    SqlCommand         command         =         new
SqlCommand("EditTeacher", connection);
                    command.CommandType                          =
CommandType.StoredProcedure;
                    command.Parameters.AddWithValue("@TeacherID",
editTeacherWindow.TeacherID);
                    command.Parameters.AddWithValue("@FirstName",
editTeacherWindow.FirstName);
```

```
                    command.Parameters.AddWithValue("@LastName",
editTeacherWindow.LastName);
                    command.Parameters.AddWithValue("@Patronymic",
editTeacherWindow.Patronymic);
                    command.Parameters.AddWithValue("@Login",
editTeacherWindow.Login);
                    command.Parameters.AddWithValue("@Password",
editTeacherWindow.Password);
                    command.Parameters.AddWithValue("@PostID",
editTeacherWindow.PostID);
                    command.ExecuteNonQuery();
                }

                Update();
            }
        }
        else
        {
            MessageBox.Show("Выберите      преподавателя      для
редактирования");
        }
    }
    #endregion

    #region Disciplines

    private void TeachersComboBox_SelectionChanged(object sender,
SelectionChangedEventArgs e)
    {
        if (TeachersComboBox.SelectedValue != null)
        {
            int teacherId = (int)TeachersComboBox.SelectedValue;
            _teacherId = teacherId;
            FillDataGrid(teacherId);
        }
    }

    public void FillTeachersComboBox()
    {
        using      (SqlConnection      connection      =      new
SqlConnection(SQLConnection.connectionString))
        {
            connection.Open();
            SqlCommand command = new SqlCommand("SELECT TeacherID,
FirstName, LastName, Patronymic FROM Teacher INNER JOIN Person ON
Teacher.FKPersonID = Person.PersonID", connection);
            SqlDataAdapter        dataAdapter        =        new
SqlDataAdapter(command);
            DataTable dataTable = new DataTable("Teachers");
            dataAdapter.Fill(dataTable);
```

```
            foreach (DataRow row in dataTable.Rows)
            {
                string fullName = row["FirstName"].ToString() + " "
+ row["LastName"].ToString();
                if (row["Patronymic"] != DBNull.Value)
                {
                    fullName += " " + row["Patronymic"].ToString();
                }
                row["FirstName"] = fullName;
            }
            TeachersComboBox.ItemsSource = dataTable.DefaultView;
            TeachersComboBox.DisplayMemberPath = "FirstName";
            TeachersComboBox.SelectedValuePath = "TeacherID";

            SupTeachersComboBox.ItemsSource                          =
dataTable.DefaultView;
            SupTeachersComboBox.DisplayMemberPath = "FirstName";
            SupTeachersComboBox.SelectedValuePath = "TeacherID";


        }
    }


    public void FillDataGrid(int teacherId)
    {
        using      (SqlConnection      connection      =      new
SqlConnection(SQLConnection.connectionString))
        {
            connection.Open();
            SqlCommand           command           =           new
SqlCommand("GetTeacherDisciplines", connection);
            command.CommandType = CommandType.StoredProcedure;
            command.Parameters.AddWithValue("@TeacherID",
teacherId);
            SqlDataAdapter         dataAdapter         =         new
SqlDataAdapter(command);
            DataTable           dataTable           =           new
DataTable("TeacherDisciplines");
            dataAdapter.Fill(dataTable);

            if
(!dataTable.Columns.Contains("TypeWork_Specialization_DisciplineID"
))
            {
                DataColumn           column           =           new
DataColumn("TypeWork_Specialization_DisciplineID", typeof(int));
                dataTable.Columns.Add(column);
            }
```

```
            if (!dataTable.Columns.Contains("GroupID"))
            {
                DataColumn column = new DataColumn("GroupID",
typeof(int));
                dataTable.Columns.Add(column);
            }

            _teachers_disciplinesGrid.AutoGeneratingColumn      +=
(sender, e) =>
            {
                if                (e.PropertyName                ==
"TypeWork_Specialization_DisciplineID"    ||    e.PropertyName    ==
"GroupID")
                {
                    e.Cancel = true;
                }
            };

            _teachers_disciplinesGrid.ItemsSource                =
dataTable.DefaultView;
            dataAdapter.Update(dataTable);
        }
    }


    private void AssignTeacher_Click(object sender, RoutedEventArgs
e)
    {
        TeacherAssignWindow     TeacherAssignWindow     =     new
TeacherAssignWindow(_teacherId);
        TeacherAssignWindow.ShowDialog();
        Update();

        TeacherAssignWindow.Closed += UpdateOnClose;
    }

    private    void    DeleteAssignTeacher_Click(object    sender,
RoutedEventArgs e)
    {

        MessageBoxResult result = MessageBox.Show("Вы уверены, что
хотите    удалить    эту    запись?",    "Подтверждение    удаления",
MessageBoxButton.YesNo);
        if (result == MessageBoxResult.No)
        {
            return;
        }

        DataRowView                    rowView                    =
_teachers_disciplinesGrid.SelectedItem as DataRowView;
```

Продолжение листинга 6.1

```
            if (rowView != null)
            {
                int teacherId = _teacherId;
                int       typeWork_Specialization_DisciplineID       =
Convert.ToInt32(rowView["TypeWork_Specialization_DisciplineID"]);
                int groupId = Convert.ToInt32(rowView["GroupID"]);

                using     (SqlConnection     connection     =     new
SqlConnection(SQLConnection.connectionString))
                {
                    connection.Open();
                    SqlCommand          command          =          new
SqlCommand("RemoveTeacherAssignment", connection);
                    command.CommandType = CommandType.StoredProcedure;
                    command.Parameters.AddWithValue("@TeacherID",
teacherId);

command.Parameters.AddWithValue("@TypeWork_Specialization_Disciplin
eID", typeWork_Specialization_DisciplineID);
                    command.Parameters.AddWithValue("@GroupID",
groupId);
                    command.ExecuteNonQuery();
                }

                FillDataGrid(teacherId);
            }
        }

        #endregion
    }
```

Добавление новых записей в таблицу реализована по одинаковой структуре. На примере окна добавления нового преподавателя можно увидеть, что сначала программа подключается к базе данных через строку подключения, далее выполняет запрос и получает результат. Используя полученные данные, заполняется ComboBox с информацией. Остальные пользовательские контроллы доступны для заполнения данных пользователем, так как являются элементами TextBox. Программа также обрабатывает их неправильное заполнение и выводит соответствующую ошибку для пользователя. После нажатия кнопки сохранения данных выполняется подключение к базе данных, а затем выполняется процедура запроса для заполнения данных в нужных таблицах. Код класса AddTeacherWindow представлен в листинге 6.2.

Листинг 6.2 – Окно создания преподавателя

```
public partial class AddTeacherWindow : Window
{
    public AddTeacherWindow()
    {
        InitializeComponent();
```

```
        FillPostComboBox();
    }
    public void FillPostComboBox()
    {
        using (SqlConnection connection = new
SqlConnection(SQLConnection.connectionString))
        {
            connection.Open();
            SqlCommand command = new SqlCommand("SELECT PostID,
PostName FROM Post", connection);
            SqlDataAdapter dataAdapter = new
SqlDataAdapter(command);
            DataTable dataTable = new DataTable("Posts");
            dataAdapter.Fill(dataTable);
            PostComboBox.ItemsSource = dataTable.DefaultView;
            PostComboBox.DisplayMemberPath = "PostName";
            PostComboBox.SelectedValuePath = "PostID";
        }
    }

    private void AddButton_Click(object sender, RoutedEventArgs e)
    {
        if (string.IsNullOrWhiteSpace(FirstNameTextBox.Text) ||
string.IsNullOrWhiteSpace(LastNameTextBox.Text) ||
string.IsNullOrWhiteSpace(LoginTextBox.Text) ||
string.IsNullOrWhiteSpace(PasswordTextBox.Text) ||
PostComboBox.SelectedValue == null)
        {
            MessageBox.Show("Пожалуйста, заполните все поля.",
"Предупреждение", MessageBoxButton.OK, MessageBoxImage.Warning);
            return;
        }

        string connectionString = SQLConnection.connectionString;

        try
        {
            using (SqlConnection connection = new
SqlConnection(connectionString))
            {
                connection.Open();

                SqlCommand command = new SqlCommand("AddNewTeacher",
connection);
                command.CommandType = CommandType.StoredProcedure;

                command.Parameters.AddWithValue("@FirstName",
FirstNameTextBox.Text);
```

Продолжение листинга 6.2

```
                command.Parameters.AddWithValue("@LastName",
LastNameTextBox.Text);
                command.Parameters.AddWithValue("@Patronymic",
PatronymicTextBox.Text);
                command.Parameters.AddWithValue("@Login",
LoginTextBox.Text);
                command.Parameters.AddWithValue("@Password",
PasswordTextBox.Text);
                command.Parameters.AddWithValue("@PostID",
PostComboBox.SelectedValue);

                command.ExecuteNonQuery();
            }

            this.Close();
            MessageBox.Show("Новый учитель добавлен успешно!");
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Произошла ошибка при добавлении нового
учителя:      {ex.Message}",      "Ошибка",      MessageBoxButton.OK,
MessageBoxImage.Error);
        }
    }
```

Редактирование различных данных осуществляется через схожую структуру, представленную классом EditTeacherWindow. При открытии окна все пользовательские контроллы заполняются нужными данными, которые подлежат изменению, учитывая выбранный ID преподавателя. После внесения требующихся изменений при нажатии на кнопку сохранения в окне, из которого было вызвано окно редактирования (в данном случае TeacherPage), выполняется запрос на сервер для внесения изменений в таблицы. EditTeacherWindow представлен в листинге 6.3.

Листинг 6.3 – Окно редактирования преподавателя

```
public partial class EditTeacherWindow : Window
{
    public int TeacherID { get; private set; }
    public string FirstName { get; private set; }
    public string LastName { get; private set; }
    public string Patronymic { get; private set; }
    public string Login { get; private set; }
    public string Password { get; private set; }
    public int PostID { get; private set; }

    public EditTeacherWindow(DataRowView row)
    {
        InitializeComponent();

        TeacherID = (int)row["TeacherID"];
```

```
            FirstNameTextBox.Text = (string)row["FirstName"];
            LastNameTextBox.Text = (string)row["LastName"];
            PatronymicTextBox.Text = (string)row["Patronymic"];
            LoginTextBox.Text = (string)row["Login"];
            PasswordTextBox.Text = (string)row["Password"];
            PostID = (int)row["PostID"];

            FillPostComboBox();
            PostComboBox.SelectedValue = PostID;
        }

        public void FillPostComboBox()
        {
            using (SqlConnection connection = new
SqlConnection(SQLConnection.connectionString))
            {
                connection.Open();
                SqlCommand command = new SqlCommand("SELECT PostID,
PostName FROM Post", connection);
                SqlDataAdapter dataAdapter = new
SqlDataAdapter(command);
                DataTable dataTable = new DataTable("Posts");
                dataAdapter.Fill(dataTable);
                PostComboBox.ItemsSource = dataTable.DefaultView;
                PostComboBox.DisplayMemberPath = "PostName";
                PostComboBox.SelectedValuePath = "PostID";
            }
        }

        private void SaveButton_Click(object sender, RoutedEventArgs e)
        {
            FirstName = FirstNameTextBox.Text;
            LastName = LastNameTextBox.Text;
            Patronymic = PatronymicTextBox.Text;
            Login = LoginTextBox.Text;
            Password = PasswordTextBox.Text;
            PostID = (int)PostComboBox.SelectedValue;

            this.DialogResult = true;
        }
    }
```

ЗАКЛЮЧЕНИЕ

В процессе выполнения данной курсовой работы были закреплены навыки проектировании баз данных и реализации их в MS SQL Server 2019.

Были определены основные цели системы в соответствии с выбранным вариантом и выделены требования, которым должна удовлетворять система.

Спроектированная база данных соответствует всем требованиям, которые предъявляются в задании. Данная база позволяет без проблем хранить и извлекать нужную информацию. Разработанная система способна определять возникающие ошибки и уведомлять об этом пользователя, чтобы в любой момент он знал из-за чего или почему произошла ошибка, и устранил её.

Разработанная информационная система кафедры (преподаватели и предметы) удовлетворяет всем требованиям комфортного использования и обеспечивает беспроблемное хранение и изменение информации, а также реализует различный функционал, требующийся для каждого варианта пользователя: администратора и преподавателя.

В результате выполнения курсовой работы были спроектированы и реализованы: база данных информационной системы кафедра (преподаватели и предметы), программа, которая эффективно взаимодействует с базой данных. Программное средство реализовано с помощью языка программирования C#.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 База данных и требования к базе данных [Электронный ресурс]. Режим доступа: https://cyberleninka.ru/article/n/baza-dannyh-i-trebovaniya-k-baze-dannyh/viewer

2 База данных. Система управления базами данных. Компоненты системы баз данных [Электронный ресурс]. Режим доступа: https://studfile.net/preview/3545314/page:21/

3 Алгоритмы назначения первичных ключей в заполненных таблицах [Электронный ресурс]. Режим доступа: https://cyberleninka.ru/article/n/algoritmy-naznacheniya-pervichnyh-klyuchey-v-zapolnennyh-tablitsah/viewer

4 Внешние ключи и их целостность [Электронный ресурс]. Режим доступа: https://kazedu.com/referat/71716/3

5 Нормализация отношений. Шесть нормальных форм [Электронный ресурс]. Режим доступа: https://habr.com/ru/articles/254773/

ПРИЛОЖЕНИЕ А
(обязательное)
Концептуальная схема БД

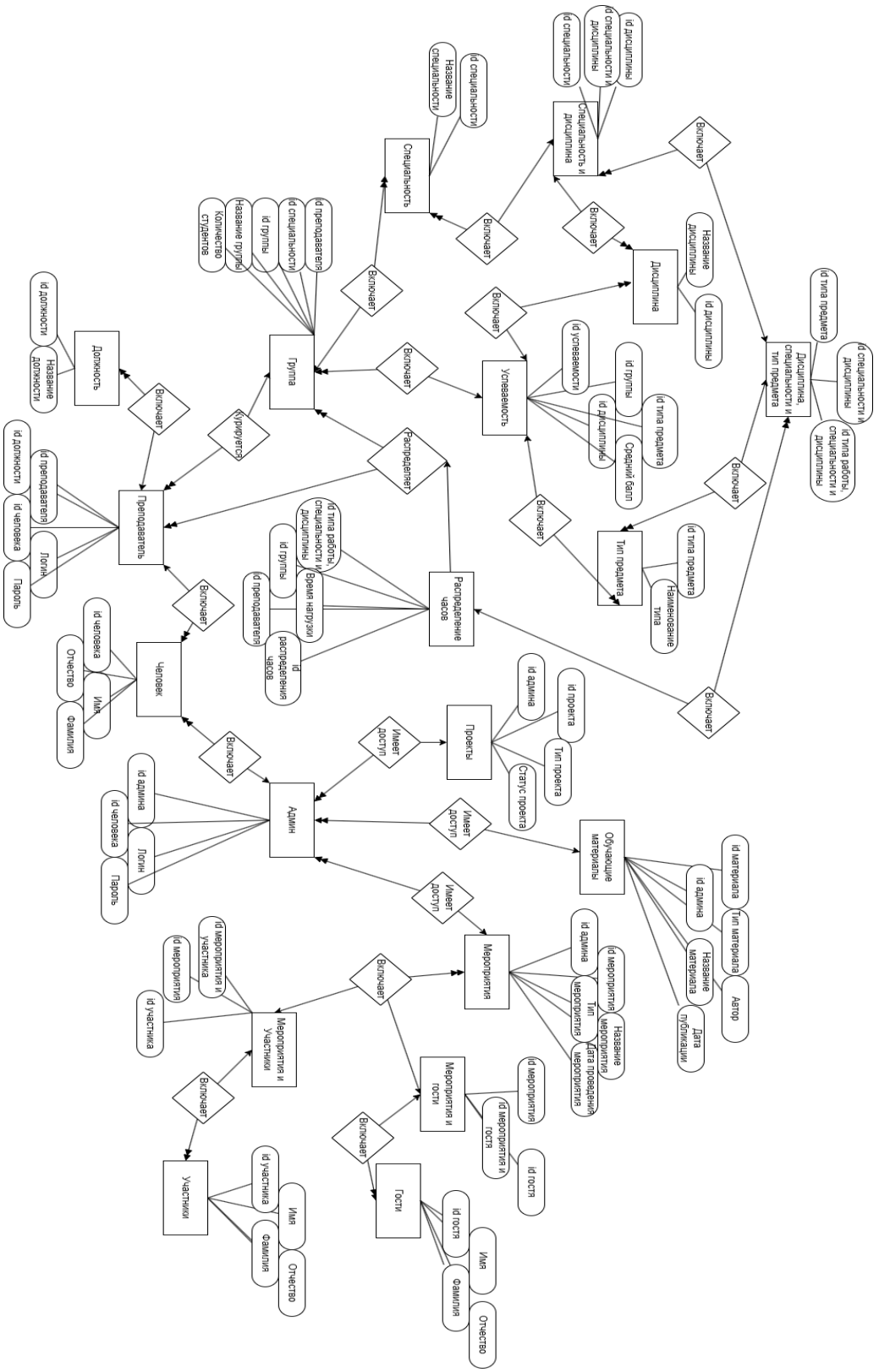

Рисунок А.1 – Концептуальная модель проектируемой базы данных

ПРИЛОЖЕНИЕ Б
(обязательное)
Схема реляционной базы данных

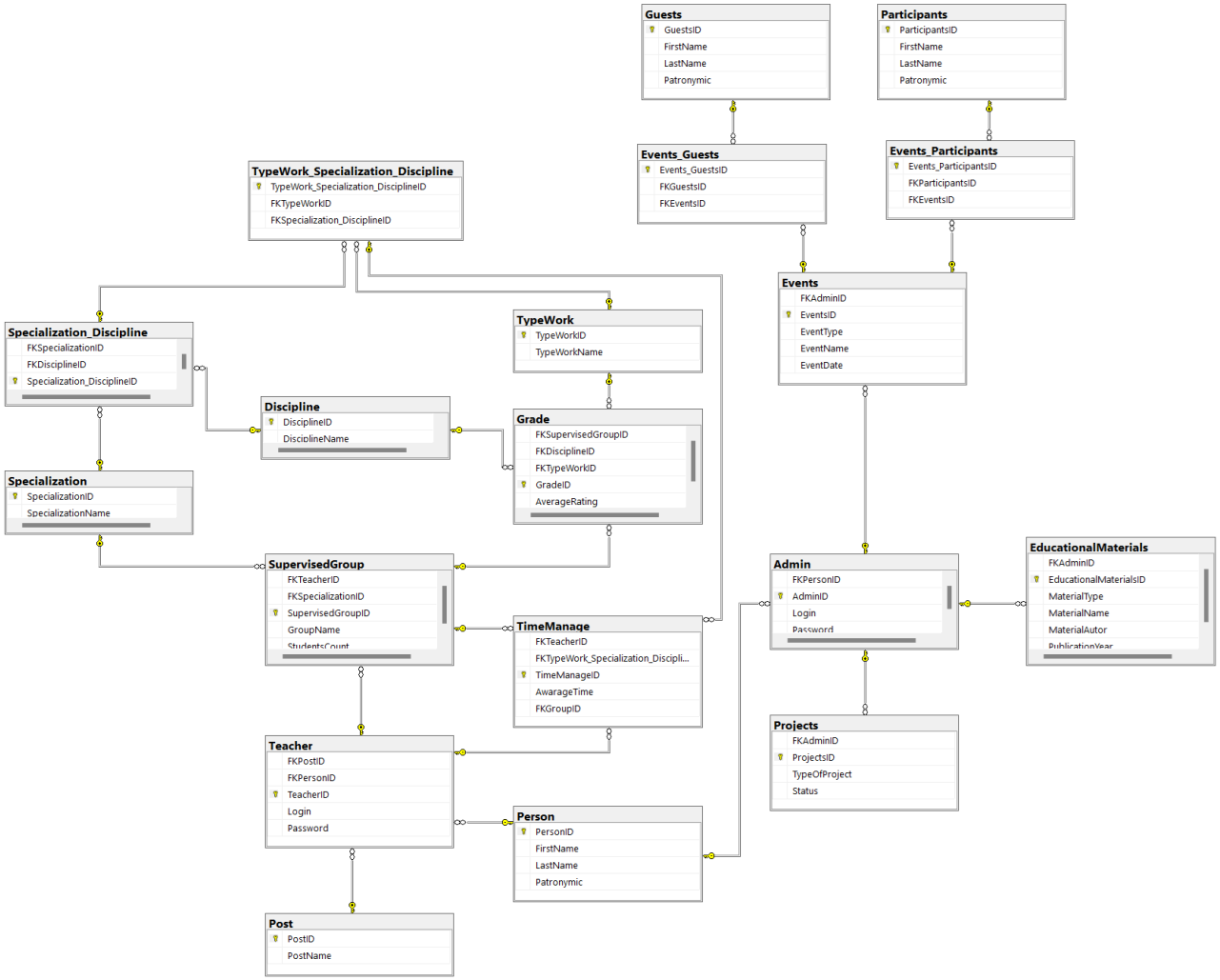

Рисунок Б.1 – Схема реляционной базы данных

ПРИЛОЖЕНИЕ В
(справочное)
Описание задания курсовой работы

Реализованная база данных должна состоять не менее чем из 15-и таблиц.

В таблице находятся данные о фамилии, имени и отчестве преподавателя, сами преподаватели находятся в отдельной таблице и наследуют эти данные из прошлой таблицы. Каждый преподаватель имеет свою должность, может вести разные дисциплины, которые разделяются на типы занятий (лабораторные работы, практические занятия, лекции), курирует одну или несколько групп студентов. За преподавателем закрепляется определенное количество часов нагрузки. В нагрузку входит: количество часов на занятие, специальность, которую содержит в себе группа студентов, у которой ведется дисциплина, а также тип занятия. Пользователем системы является преподаватель, который может только просматривать информацию о закреплениях для своей нагрузки, а также проставлять средний балл для дисциплин у группы.

Также у информационной системы присутствует администратор, который может редактировать, добавлять и удалять данные для всех существующих таблиц. Кроме того, администратор добавляет в информационную систему обучающие материалы, содержащие в себе информацию о дате публикации, авторе и их название. Также присутствует таблица мероприятий, проходящих на факультете, в которой администратор заносит информацию о гостях и участниках данного мероприятия.

Виды запросов в информационной системе:

– запрос, возвращающий значение общего количества студентов на факультете;

– запрос, возвращающий информацию о дисциплинах и связанных с ними группах, которые преподает учитель с заданным идентификатором;

– запрос, возвращающий полную информацию о типах работ, дисциплинах и специализациях, связанных с ними;

– запрос, возвращающий полную информацию о группах, которые курируются определенным преподавателем;

– запрос, возвращающий названия дисциплин и типов работ, связанных с заданной специализацией;

– запрос, возвращающий полную информацию о группе студентов заданным идентификатором;

– запрос, возвращающий полную информацию о всех админах, зарегистрированных в базе данных;

– запрос, возвращающий информацию об участниках и событиях, в которых они принимали, либо будут принимать участие;

– запрос, возвращающий информацию об оценках студентов по группам, дисциплинам и типам работ;

– запрос, принимающий год в качестве параметра и возвращающий все обучающие материалы, выпущенные после указанного года;

– запрос, принимающий дату в качестве параметра и возвращающий список гостей, которые посетили мероприятие после указанной даты;

– запрос, возвращающий общее количество студентов на факультете;

– запрос, возвращающий все проекты со статусом «В разработке»;

– запрос, возвращающий всех преподавателей с заданным именем.

Необходимо предусмотреть возможность просмотра преподавателю отчета о его нагрузке, а также о курируемых им группах.

ПРИЛОЖЕНИЕ Г
(обязательное)
Главная и рабочие формы приложения

В ходе проектирования было определено, что при запуске приложения сначала будет открываться окно для входа в аккаунт. Исходя из введенных данных пользователь сможет войти либо в качестве администратора, либо в качестве преподавателя. Окно входа в систему представлено на рисунке Г.1.

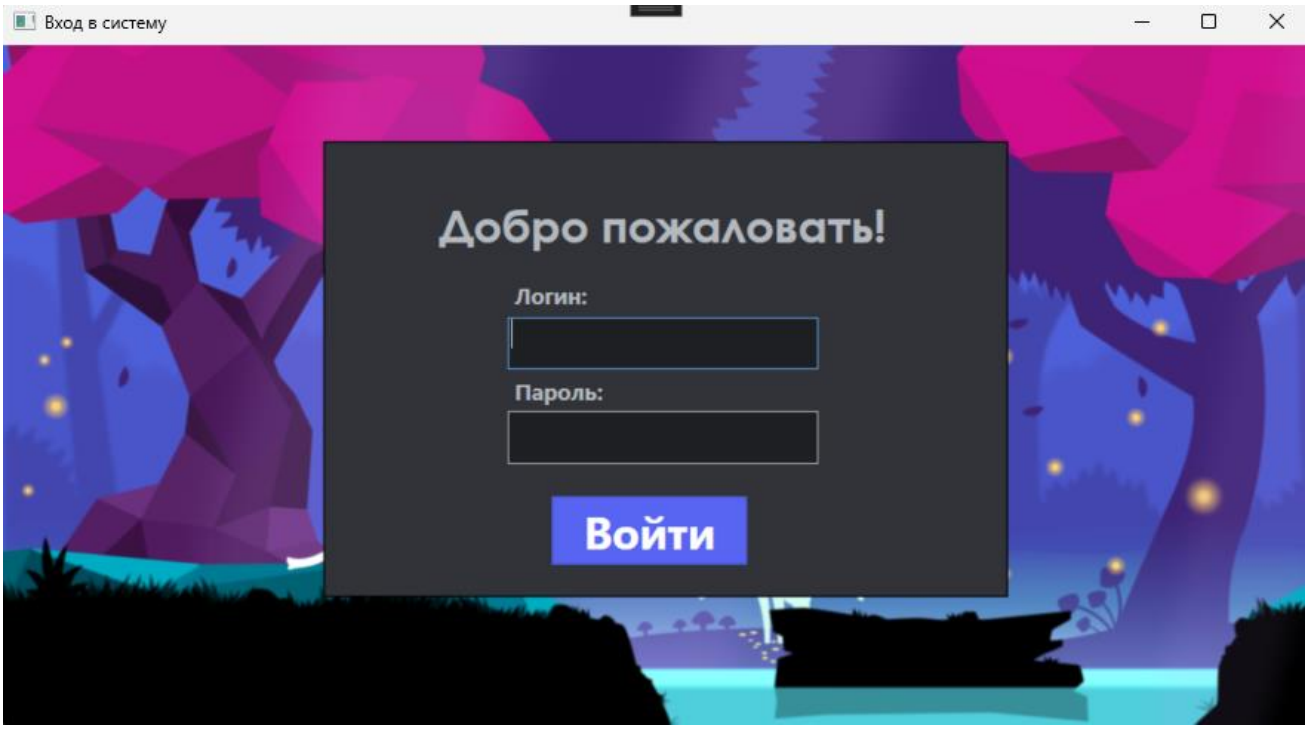

Рисунок Г.1 – Окно входа в аккаунт

После входа в аккаунт в качестве администратора, программа откроет главное меню, где и представлен весь функционал приложения. Вид главного меню представлен на рисунке Г.2.

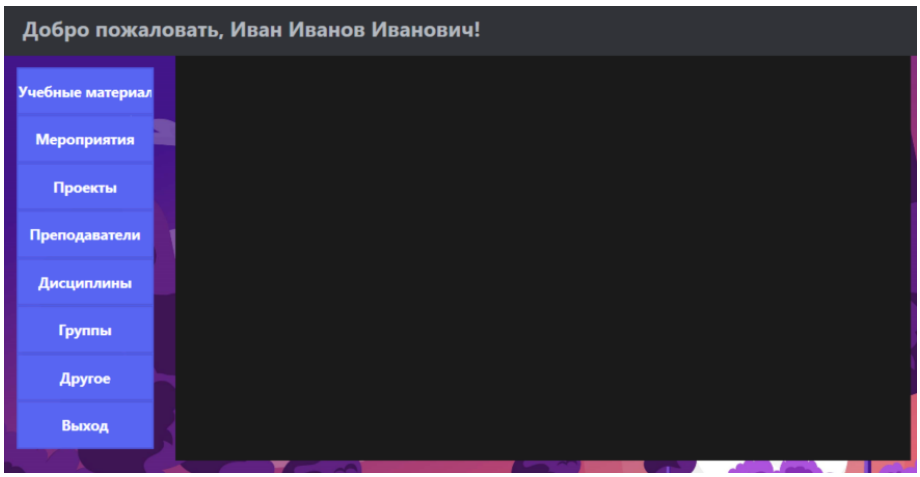

Рисунок Г.2 – Главное меню приложения

При выборе требующегося пункта меню пользователь переходит на соответствующую страницу в данном окне, где расположен дополнительный функционал работы с данными таблиц или функционал создания связей между данными.

Пример такого окна можно увидеть на рисунке Г.3.

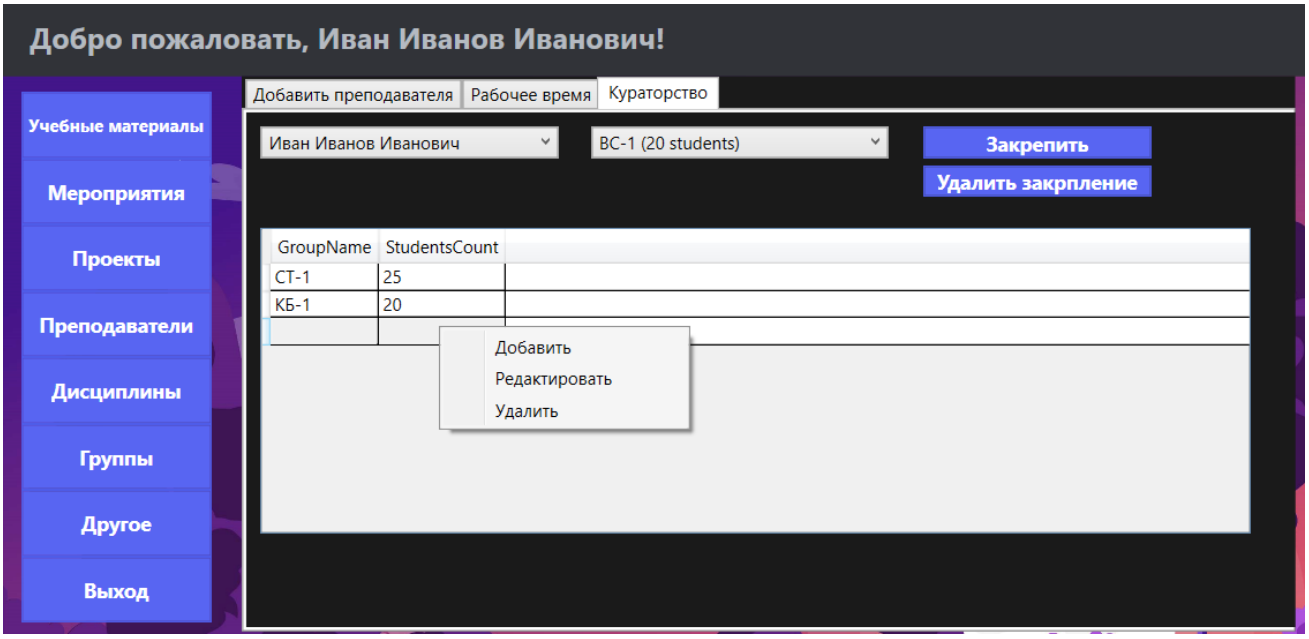

Рисунок Г.3 – Окно закрепления курируемой группы за преподавателем

Пользователю предлагается выбрать информацию из ComboBox-элемента после чего уже заполненные данные отобразятся в виде таблицы. При нажатии на правую клавишу мыши по таблице с данными пользователь увидит контекстное меню с предоставленными вариантами работы с данными. В зависимости от выбранного пункта, приложение запустит соответствующий блок кода. Если пользователь также выбирает данные во втором ComboBox и нажимает на кнопку закрепления, то в базу данных добавится новая запись о закреплении.

Пример окна заполнения новых данных представлен на рисунке Г.4.

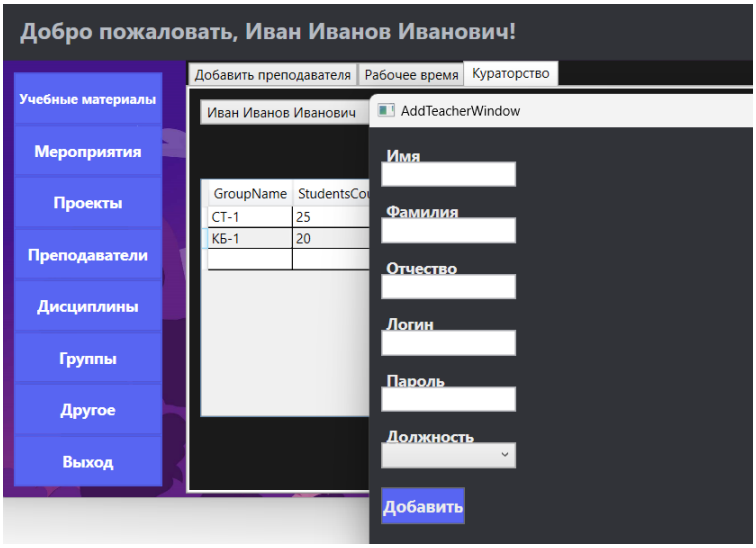

Рисунок Г.4 – Окно редактирования записи

При вводе некорректных данных пользователем, программа отобразит исключение с текстом ошибки, которую должен исправить пользователь перед тем, как добавит данные. Таким образом программа реализует интуитивно понятный пользовательский интерфейс, который предотвращает некорректную работу с данными, а также реагирует на непреднамеренные ошибки пользователя.

На рисунке Г.5 представлены окна, реализующие дополнительный функционал приложения, представленный обработкой запросов с введенными пользователем данными и отображением полученного результата.

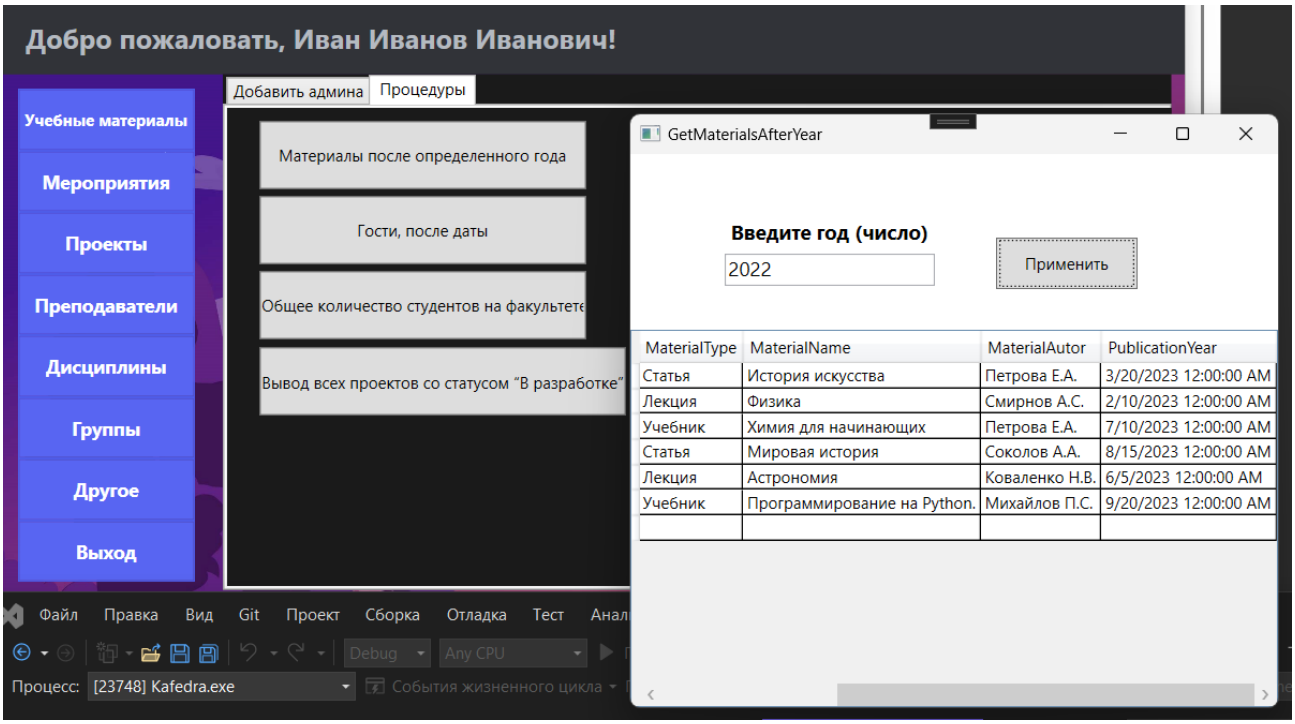

Рисунок Г.5 – Окно с функционалом дополнительных запросов

При нажатии на соответствующую кнопку отображается окно с нужным вводом данных и их отображением.