СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc25751)

[1. Анализ и описание информационной системы: "Расписание занятий в вузе" 4](#_Toc18613)

[1.1 Особенности предметной области 4](#_Toc28101)

[1.2 Описание бизнес-процессов 5](#_Toc842)

[1.3 Постановка цели и задач системы 7](#_Toc2513)

[2. Проектирование базы данных для системы "Расписание занятий в вузе" 9](#_Toc24939)

[2.1 Проектирование концептуальной модели данных 9](#_Toc6077)

[2.2 Проектирование логической модели данных 11](#_Toc341)

[2.3 Проектирование физической модели данных 15](#_Toc4879)

[3. Разработка и реализация клиентского приложения "Расписание занятий в вузе" 17](#_Toc21964)

[3.1 Выбор средств реализации 17](#_Toc15241)

[3.2 Архитектура программного обеспечения 18](#_Toc32616)

[3.3 Функциональные возможности 20](#_Toc6070)

# ВВЕДЕНИЕ

Современные высшие учебные заведения сталкиваются с необходимостью эффективного управления учебным процессом, что требует сложной организации расписания аудиторных занятий. Расписание должно учитывать множество факторов: распределение преподавателей, загрузку аудиторий, особенности учебных групп, а также обеспечивать отсутствие временных и пространственных конфликтов.

Актуальность разработки информационной системы "Расписание занятий в вузе" обусловлена:

* Ростом количества студентов и преподавателей в высших учебных заведениях
* Усложнением учебных планов и программ
* Необходимостью оперативного реагирования на изменения в учебном процессе
* Требованиями к прозрачности и доступности информации о расписании

Целью данного проекта является разработка информационной системы для автоматизации составления и управления расписанием аудиторных занятий в вузе.

# 1. Анализ и описание информационной системы: "Расписание занятий в вузе"

## 1.1 Особенности предметной области

Предметная область системы расписания вуза охватывает процессы планирования, организации и контроля учебного процесса, включая распределение аудиторного фонда, преподавательского состава и учебных групп по временным интервалам. Каждое учебное занятие имеет комплекс параметров: дисциплину, вид занятия (лекция, практика, лабораторная работа), преподавателя, аудиторию, временной интервал ("пару"), показатель недели (числитель/знаменатель) и состав группы (полная группа или подгруппа).

Ключевым ресурсом в учебном процессе выступают преподаватели, чья специализация определяется читаемыми дисциплинами и научными интересами. Один преподаватель может вести несколько разных дисциплин для различных групп, а сложные дисциплины могут преподаваться несколькими преподавателями. Распределение преподавательской нагрузки требует учета их занятости, научной работы и административных обязанностей.

Аудиторный фонд включает помещения различного типа: лекционные залы, специализированные лаборатории, компьютерные классы и обычные учебные аудитории. Каждая аудитория имеет ограниченную вместимость и специализацию, что необходимо учитывать при составлении расписания. Некоторые занятия требуют специального оборудования, доступного только в определенных аудиториях.

Учебные группы формируются по направлениям подготовки и курсам, при этом одна дисциплина может читаться для нескольких групп одновременно (потоковые лекции), а другие занятия проводятся раздельно для каждой группы. Особую сложность представляет учет недельной цикличности (числитель/знаменатель) и необходимости избегать "окон" в расписании как для студентов, так и для преподавателей.

Все эти особенности делают разработку информационной системы расписания критически важной для обеспечения эффективного учебного процесса, оптимального использования ресурсов вуза и комфортных условий работы преподавателей и обучения студентов.

## 1.2 Описание бизнес-процессов

Основные бизнес-процессы системы расписания вуза направлены на формирование оптимального учебного графика, начиная с подготовки исходных данных и заканчивая публикацией утвержденного расписания. На начальном этапе (блок A1) администратор системы вводит базовые данные: перечень дисциплин, информацию о преподавателях, аудиториях и учебных группах, что создает основу для последующего составления расписания.

Далее следует процесс непосредственного составления расписания (A2), который включает распределение занятий по временным интервалам с учетом всех ограничений и требований. Этот процесс выполняется с помощью специализированного алгоритма, который проверяет отсутствие конфликтов по:

* занятости преподавателей
* занятости аудиторий
* занятости учебных групп
* недельной цикличности

Особое внимание уделяется процессу корректировки расписания (A3), который выполняется по мере необходимости в течение учебного семестра. Этот процесс включает обработку заявок на замены преподавателей, перенос занятий и оперативное внесение изменений с обязательной проверкой на отсутствие новых конфликтов.

Завершающим этапом является процесс публикации и анализа расписания (A4), который включает:

* формирование индивидуальных расписаний для преподавателей и групп
* экспорт данных в различные форматы (PDF, Excel)
* размещение информации на информационных стендах и сайте вуза
* анализ загрузки аудиторий и преподавателей
* формирование отчетности для деканатов и учебного управления

Все процессы выполняются с учетом нормативных требований образовательных стандартов и внутренних регламентов вуза, что обеспечивает соответствие расписания учебным планам и оптимальную организацию учебного процесса представлены на рисунке 1.

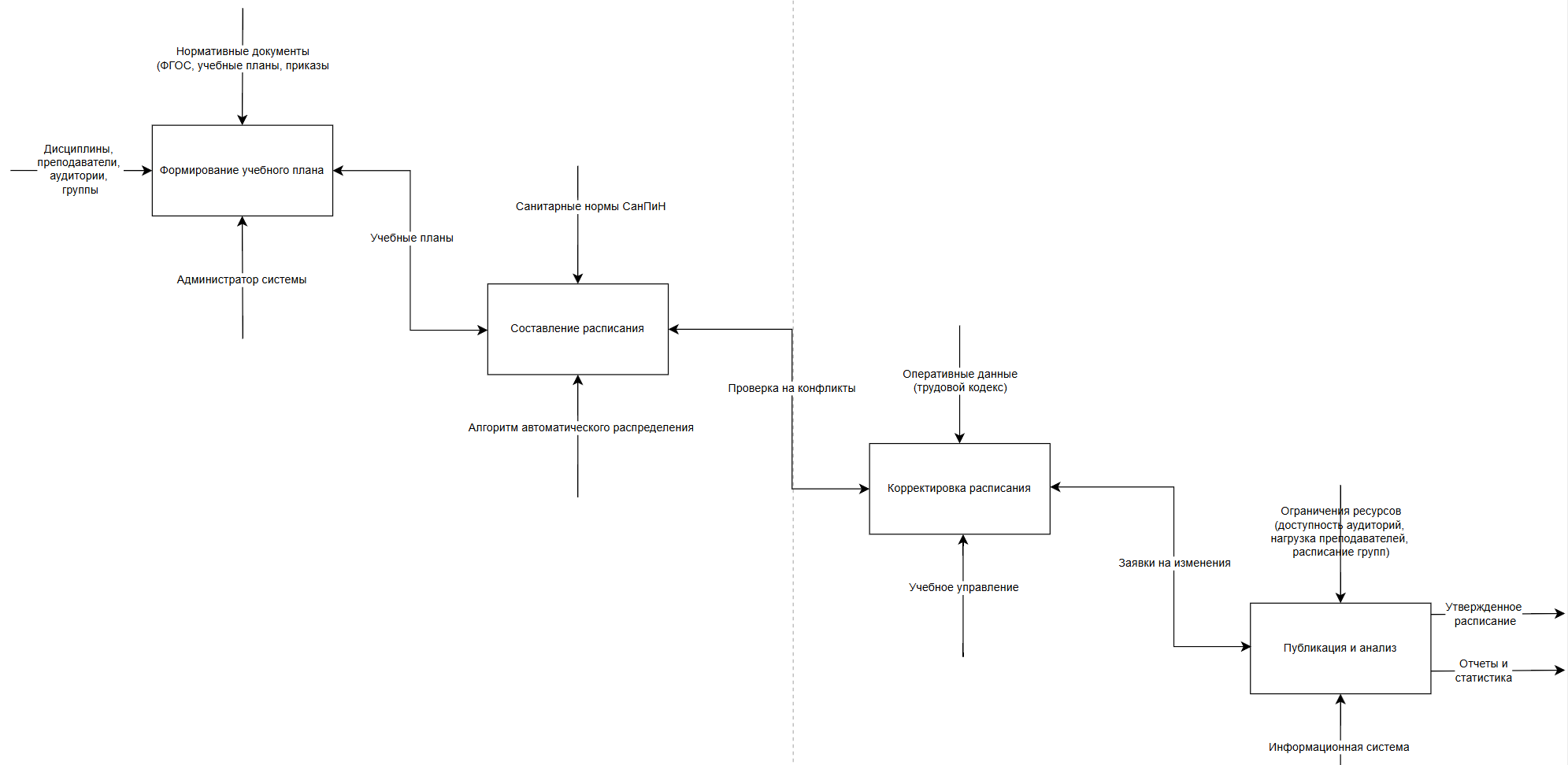


Рисунок 1 – Диаграмма IDEF0

1.3 Постановка цели и задач системы

Целью разработки информационной системы «Расписание занятий в вузе» является автоматизация процессов составления, корректировки и анализа учебного расписания с обеспечением оптимального распределения ресурсов (преподавателей, аудиторий, учебных групп) и исключением временных и пространственных конфликтов. Система должна стать централизованным инструментом для управления учебным процессом, позволяющим администрации вуза, преподавателям и студентам оперативно получать актуальную информацию о расписании, минимизировать «окна» в графиках и повысить эффективность использования аудиторного фонда.

Для достижения поставленной цели решаются следующие задачи:

* Собрать и формализовать требования к учёту дисциплин, преподавателей, аудиторий, учебных групп и временных интервалов пар;
* Спроектировать логическую структуру базы данных, включая таблицы (Subjects, Teachers, Auditoriums, Groups, Schedule), их атрибуты и связи (внешние ключи);
* Реализовать механизмы проверки расписания на конфликты (пересечение занятий у преподавателей, групп и аудиторий, включая учёт недельной цикличност;
* Обеспечить гибкую корректировку расписания с возможностью ручного изменения занятий, замены преподавателей и переноса пар;
* Автоматизировать формирование отчётов. То есть индивидуальные расписания для преподавателей и групп, анализ загрузки аудиторий и статистика преподавательской нагрузки;
* Реализовать экспорт данных в форматы Excel и Word для публикации на сайте вуза и информационных стендах;
* Предусмотреть разграничение прав доступа для разных ролей: администраторы (полный доступ), преподаватели (просмотр и запрос изменений), студенты (только просмотр);
* Обеспечить масштабируемость системы для возможного подключения дополнительных модулей (например, учёта посещаемости или интеграции с системой электронного деканата).

В результате система позволит:

* сократить время составления и корректировки расписания,
* исключить ошибки, связанные с «накладками» в расписании,
* повысить прозрачность учебного процесса для всех участников,
* оптимизировать использование ресурсов вуза (аудитории, преподаватели).

Это соответствует как требованиям образовательных стандартов, так и потребностям пользователей (администрации, преподавателей, студентов).

# 2. Проектирование базы данных для системы "Расписание занятий в вузе"

## 2.1 Проектирование концептуальной модели данных

Концептуальное проектирование базы данных выполнено для формализации предметной области учебного расписания и определения состава сущностей, их атрибутов и взаимосвязей. На этом этапе создана независимая от СУБД модель, отражающая логическую структуру информации, обрабатываемой системой управления расписанием.

В процессе анализа предметной области были выделены следующие ключевые сущности:

* Subject (Дисциплина):

Атрибуты: subject\_id (ID), name (название)

Связи: преподается преподавателями (Teacher), изучается группами (Group)

* Teacher (Преподаватель):

Атрибуты: teacher\_id (ID), last\_name (фамилия), first\_name (имя), middle\_name (отчество), position (должность)

Связи: преподает дисциплины (Subject), ведет занятия (Schedule)

* Auditorium (Аудитория):

Атрибуты: auditorium\_id (ID), room\_number (номер), building (корпус), capacity (вместимость), type (тип)

Связи: используется для занятий (Schedule)

* Group (Учебная группа):

Атрибуты: group\_id (ID), name (название), course\_number (курс), faculty\_name (факультет)

Связи: изучает дисциплины (Subject), имеет занятия (Schedule)

* LessonType (Тип занятия):

Атрибуты: lesson\_type\_id (ID), name (название: лекция, практика и т.д.)

Связи: определяет вид занятий в расписании (Schedule)

* Schedule (Расписание):

Атрибуты: schedule\_id (ID), day\_of\_week (день недели), pair\_number (номер пары), numerator (числитель/знаменатель), is\_halfgroup (полгруппы)

Связи: объединяет все сущности (Subject, Teacher, Auditorium, Group, LessonType)

* User (Пользователь):

Атрибуты: user\_id (ID), login (логин), password (пароль), role (роль)

Связи: может быть преподавателем (Teacher) или относиться к группе (Group)

* Role (Роль):

Атрибуты: role\_id (ID), name (название: администратор, преподаватель, студент)

Связи: определяет права пользователей (User)

Концептуальная модель представлена на рисунке 2.

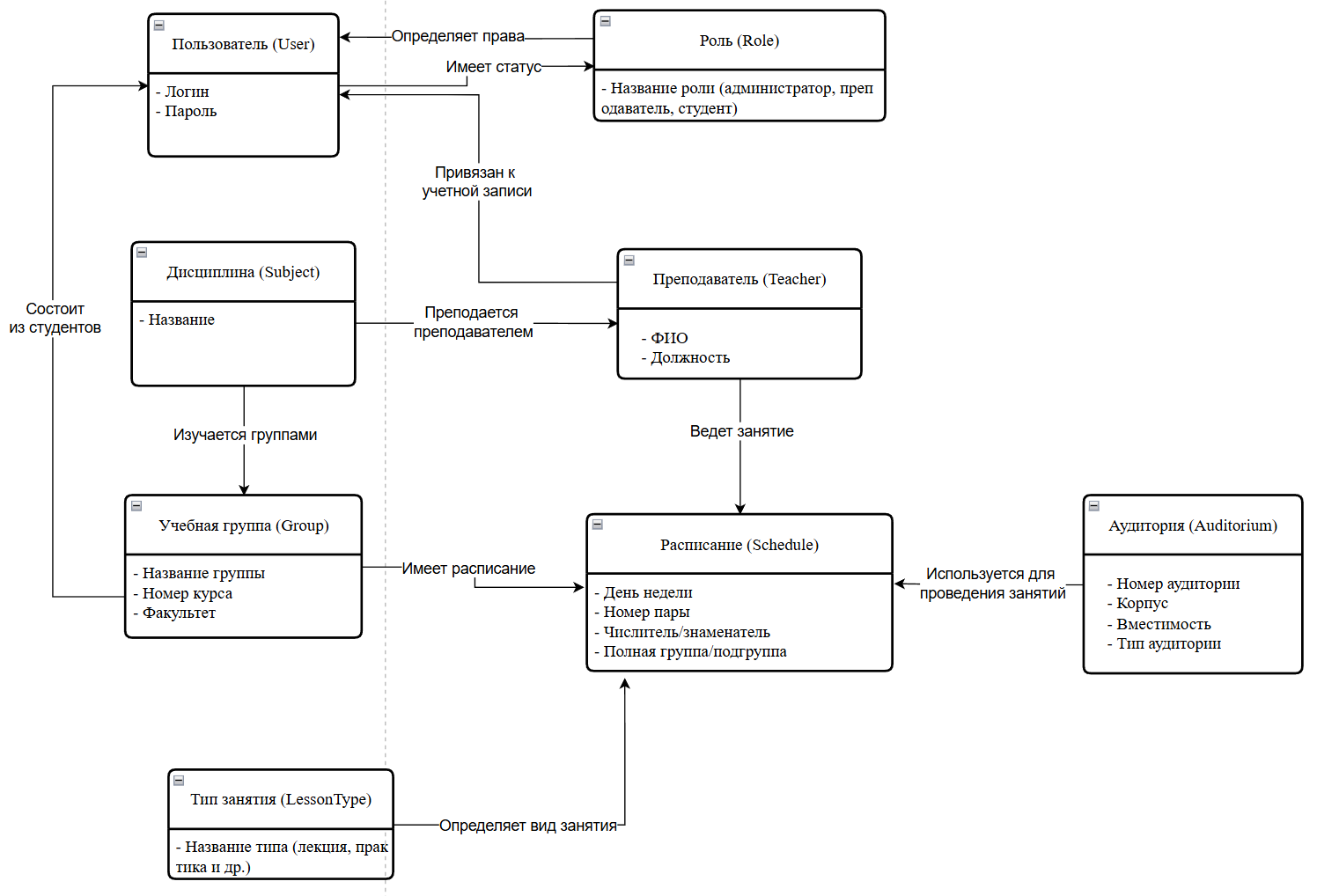


Рисунок 2 – Концептуальная модель

Концептуальная модель обеспечивает:

* Полноту учета всех параметров учебного процесса
* Отсутствие избыточности данных
* Возможность проверки расписания на конфликты
* Гибкость при изменении учебного плана
* Разграничение прав доступа для разных ролей

Разработанная концептуальная модель служит основой для последующих этапов проектирования - создания логической модели (нормализованных таблиц) и физической реализации базы данных в выбранной СУБД с учетом требований производительности и целостности данных. Модель позволяет эффективно организовать хранение информации о расписании и обеспечивает все необходимые взаимосвязи между объектами учебного процесса.

## 2.2 Проектирование логической модели данных

Проведем нормализацию данных для дальнейшего проектирования БД и построения логической и физической моделей данных. Возьмем некоторый набор ненормализованных данных, где будет две колонки: ФИО и данные. Ненормализованные данные представлены на рисунке 3.

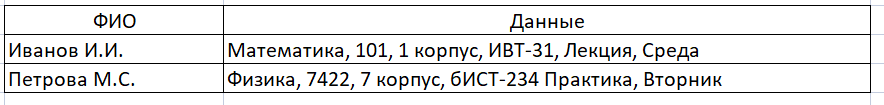


Рисунок 3 – Ненормализованные данные

Далее приведем данные из первой таблицы к Первой нормальной форме. Для этого сделаем так, чтобы все атрибуты содержали атомарные значения и не было повторяющихся значений. Создаём новую сущность и переносим туда все данные о рабочем, но только теперь их разделяем и даем название каждому.

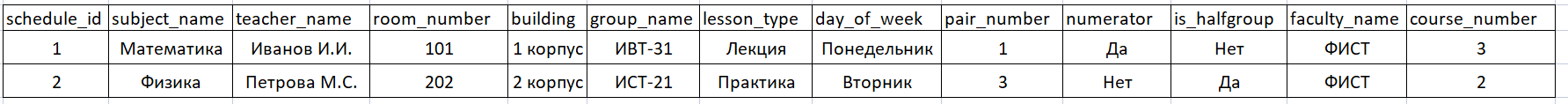


Рисунок 4 – Данные в 1NF

Приведение данных ко второй нормальной форме (2NF) для системы расписания занятий в вузе. Для приведения данных ко второй нормальной форме необходимо обеспечить, чтобы каждый неключевой атрибут полностью зависел от первичного ключа. Исходную ненормализованную структуру данных мы разделим на 6 основных сущностей:

* Сущность "Преподаватели" (Teachers)
* Сущность "Дисциплины" (Subjects)
* Сущность "Аудитории" (Auditoriums)
* Сущность "Учебные группы" (Groups)
* Сущность "Типы занятий" (LessonTypes)
* Сущность "Расписание" (Schedule)

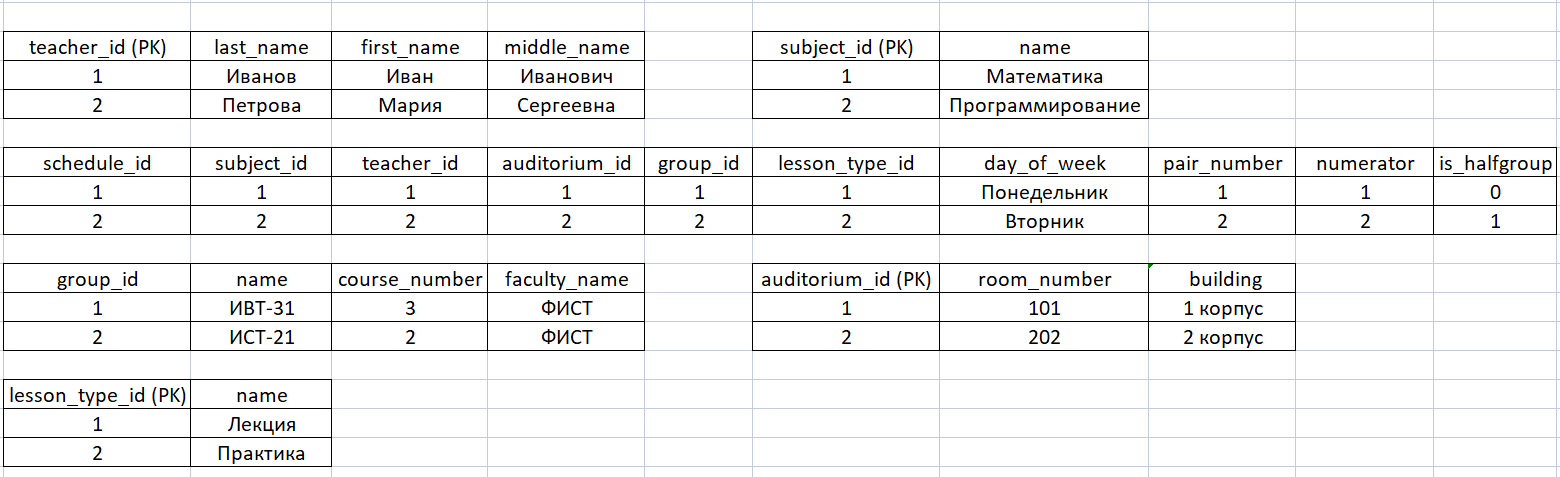


Рисунок 5 – Данные во 2NF

Третья нормальная форма (3NF) для системы расписания занятий в вузе, где данные уже соответствуют третьей нормальной форме, так как:

* Ни один неключевой атрибут не зависит от другого неключевого атрибута
* Все атрибуты зависят исключительно от первичного ключа своей сущности
* Отсутствуют транзитивные зависимости

Для полной нормализации я добавил две дополнительные сущности:

* Сущность "Users" (Пользователи)
* Сущность "Roles" (Роли)

Данные в 3NF представлены на рисунке 6.

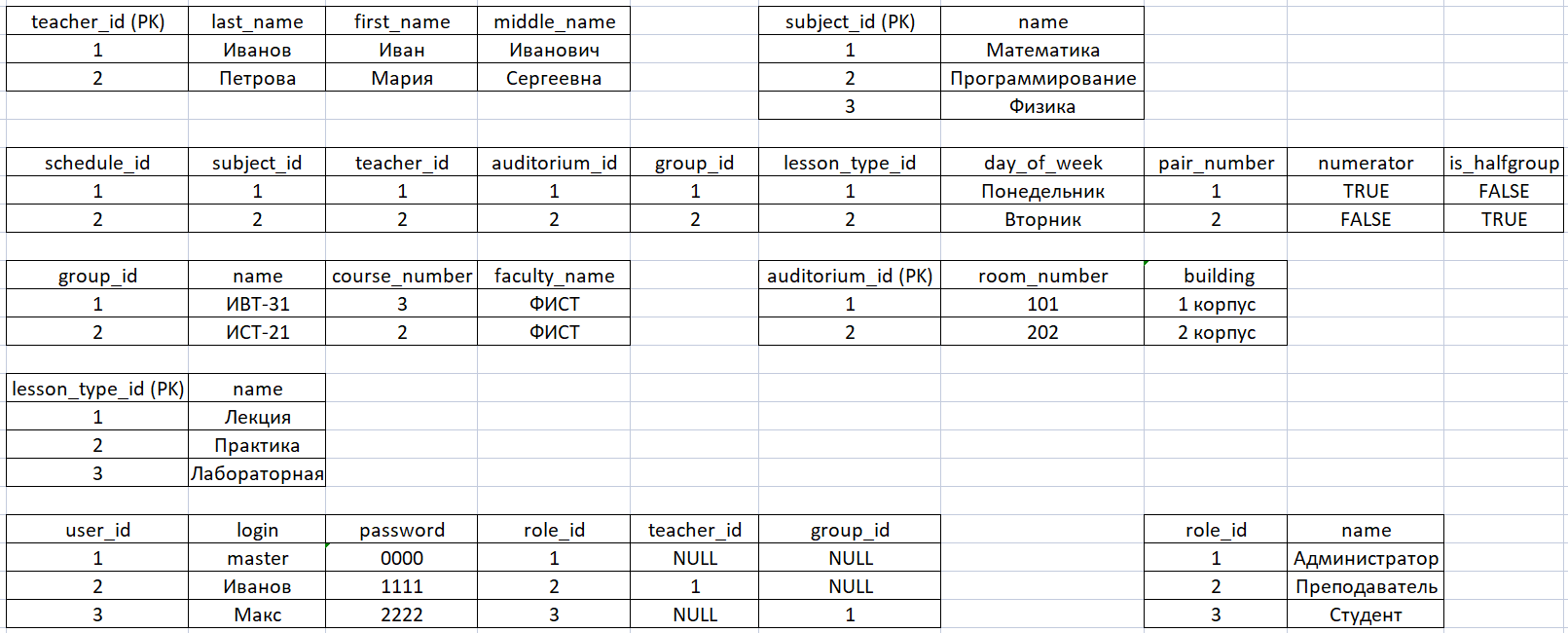


Рисунок 6 – Данные в 3NF

Перед построением логической модели данных была проведена нормализация, направленная на устранение избыточности и аномалий обновления, удаления и вставки данных. В результате модель приведена как минимум к третьей нормальной форме (3НФ), что обеспечивает логическую целостность и согласованность хранения информации. Далее представлена логическая модель рисунок 7.

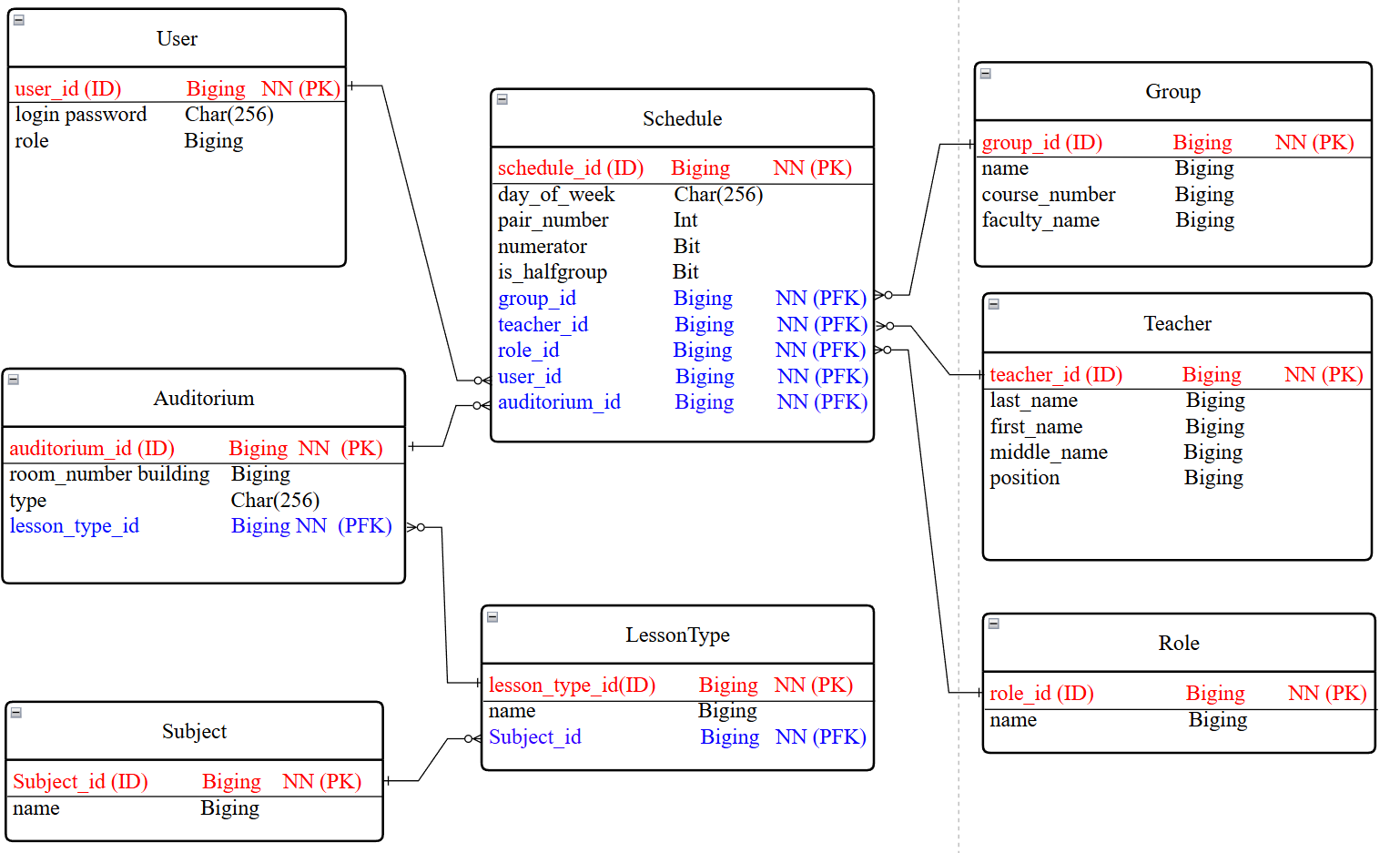


Рисунок 7 – Логическая модель данных

В процессе анализа предметной области были выделены следующие ключевые сущности:

* Subject (Дисциплина):

Атрибуты: subject\_id (ID), name (название)

Связи: преподается преподавателями (Teacher), изучается группами (Group)

* Teacher (Преподаватель):

Атрибуты: teacher\_id (ID), last\_name (фамилия), first\_name (имя), middle\_name (отчество), position (должность)

Связи: преподает дисциплины (Subject), ведет занятия (Schedule)

* Auditorium (Аудитория):

Атрибуты: auditorium\_id (ID), room\_number (номер), building (корпус), capacity (вместимость), type (тип)

Связи: используется для занятий (Schedule)

* Group (Учебная группа):

Атрибуты: group\_id (ID), name (название), course\_number (курс), faculty\_name (факультет)

Связи: изучает дисциплины (Subject), имеет занятия (Schedule)

* LessonType (Тип занятия):

Атрибуты: lesson\_type\_id (ID), name (название: лекция, практика и т.д.)

Связи: определяет вид занятий в расписании (Schedule)

* Schedule (Расписание):

Атрибуты: schedule\_id (ID), day\_of\_week (день недели), pair\_number (номер пары), numerator (числитель/знаменатель), is\_halfgroup (полгруппы)

Связи: объединяет все сущности (Subject, Teacher, Auditorium, Group, LessonType)

* User (Пользователь):

Атрибуты: user\_id (ID), login (логин), password (пароль), role (роль)

Связи: может быть преподавателем (Teacher) или относиться к группе (Group)

* Role (Роль):

Атрибуты: role\_id (ID), name (название: администратор, преподаватель, студент)

Связи: определяет права пользователей (User)

Основные связи между сущностями:

* Преподаватель (Teacher) может вести несколько дисциплин (Subject) - связь "многие-ко-многим"
* Дисциплина (Subject) изучается несколькими группами (Group) - связь "многие-ко-многим"
* Одно занятие в расписании (Schedule) связывает: 1 дисциплину, 1 преподавателя, 1 аудиторию, 1 группу и 1 тип занятия
* Пользователь (User) может иметь одну роль (Role) и привязываться к преподавателю или группе

## 2.3    Проектирование физической модели данных

Физическая модель данных представляет собой реализацию логической структуры базы данных в конкретной системе управления базами данных (СУБД) с учетом типов данных, ограничений целостности и индексации. На данном этапе определяются атрибуты таблиц с точными типами данных, устанавливаются первичные и внешние ключи, а также обеспечивается поддержка ссылочной и предметной целостности. Представлена на рисунке 7.

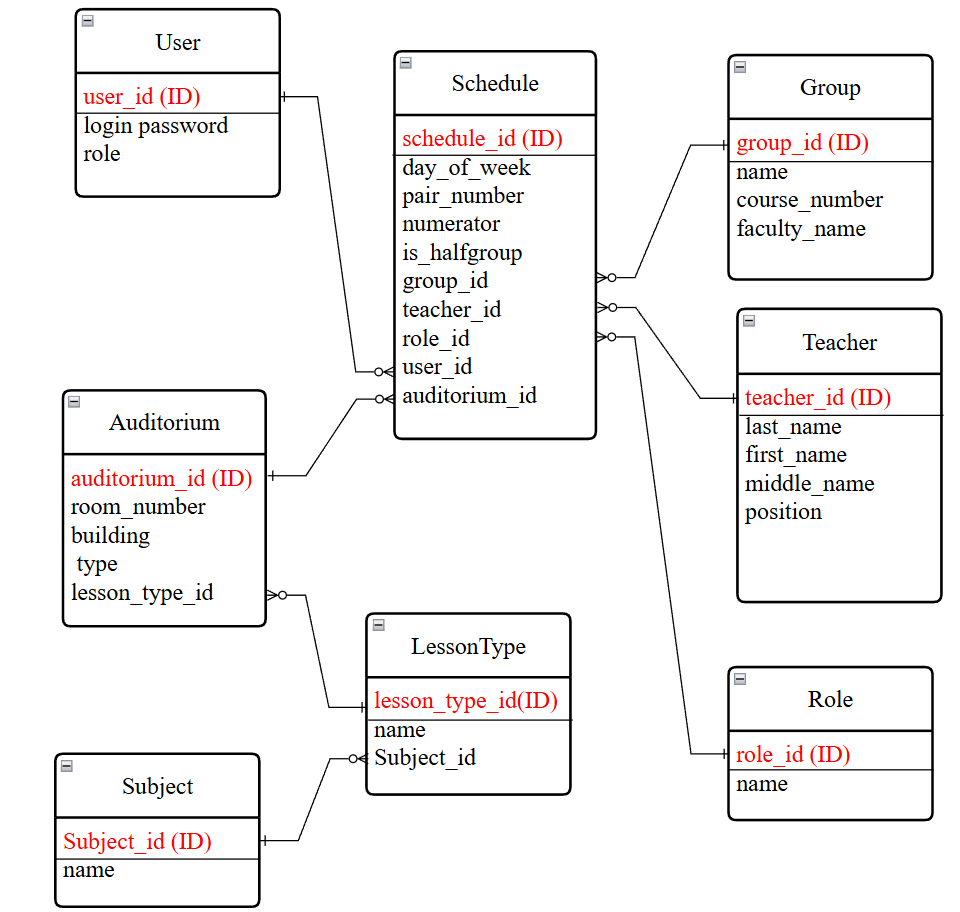


Рисунок 8 – Физическая модель данных

На основании логической модели разработана физическая структура базы данных для СУБД MySQL.

# Разработка и реализация клиентского приложения "Расписание занятий в вузе"

## 3.1 Выбор средств реализации

Для разработки клиентского приложения информационной системы "Расписание занятий в вузе" был выбран язык C# и платформа .NET Framework, которые обеспечивают:

* Надежную интеграцию с Microsoft SQL Server
* Удобное создание графического интерфейса с помощью Windows Forms
* Поддержку работы с документами через Microsoft Office Interop (Word, Excel)

Графический интерфейс реализован с использованием:

* Windows Forms для основного функционала
* DataGridView для отображения табличных данных
* Компонентов для работы с БД (SqlConnection, SqlCommand)

Для работы с базой данных использована:

* Microsoft SQL Server как основная СУБД
* Технология ADO.NET для подключения и выполнения запросов
* Встроенный механизм ConfigurationManager для хранения строк подключения

Дополнительные технологии:

* Microsoft.Office.Interop для экспорта в Word/Excel
* Entity Framework для объектно-реляционного отображения
* LINQ для удобной работы с данными
* Выбор данного стека технологий обусловлен:
* Оптимальной производительностью для учебного заведения
* Хорошей интеграцией с продуктами Microsoft
* Надежностью и стабильностью работы
* Поддержкой со стороны разработчика

## 3.2 Архитектура программного обеспечения

Программное обеспечение информационной системы "Расписание занятий в вузе" построено по модульному принципу с разделением на уровни по ролям пользователей, что обеспечивает расширяемость, читаемость кода и упрощает сопровождение. Архитектура проекта условно делится на несколько функциональных пакетов (модулей), каждый из которых реализует логику в пределах определённой зоны ответственности.

Приложение построено по трехслойной архитектуре с четким разделением:

* Уровень представления (View):
* Формы Windows Forms (Form1, EditScheduleForm и др.)
* Пользовательские контролы для отображения данных
* Диалоговые окна для взаимодействия с пользователем

Уровень бизнес-логики (Controller):

* Классы для работы с расписанием (ScheduleManager)
* Модуль проверки конфликтов (ScheduleValidator)
* Генератор отчетов (ReportGenerator)
* Логика авторизации (AuthService)

Уровень данных (Model):

* Классы сущностей (Subject, Teacher, Group и др.)
* Репозитории для работы с БД (ScheduleRepository)
* DTO-объекты для передачи данных между слоями

Основные модули системы:

* Модуль администрирования:
* Управление расписанием
* Редактирование справочников
* Настройка системы

Модуль преподавателя:

* Просмотр личного расписания
* Запрос на изменения
* Формирование отчетов

Модуль студента:

* Просмотр расписания группы
* Экспорт в различные форматы
* Фильтрация по параметрам

Ключевые особенности архитектуры:

* Четкое разделение по ролям (администратор, преподаватель, студент)
* Использование паттерна Repository для работы с БД
* Поддержка транзакций при изменении данных
* Кэширование часто используемых данных
* Гибкая система экспорта данных

Структура проекта:

* BDLaba11 (основное пространство имен)
* Forms (все формы приложения)
* Models (сущности базы данных)
* Services (бизнес-логика)
* Repositories (работа с БД)
* Utilities (вспомогательные классы)
* DTO (объекты передачи данных)

Такая архитектура обеспечивает:

* Масштабируемость системы
* Простоту поддержки кода
* Безопасность данных
* Гибкость при изменении требований

Модульная диаграмма классов представлена на рисунке 10.

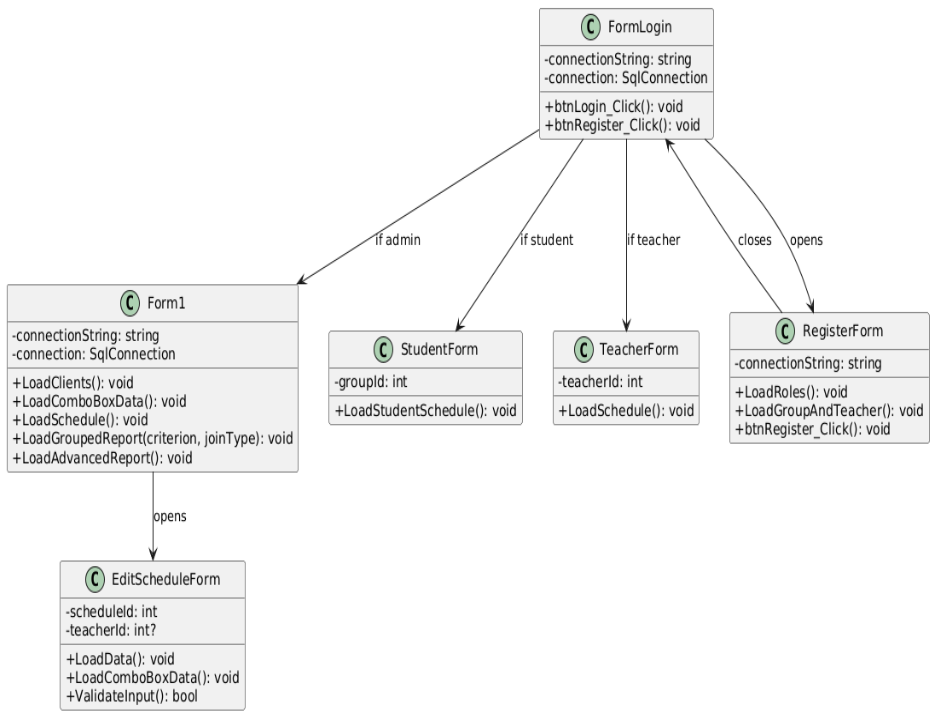


Рисунок 9 – Диаграмма классов

## 3.3 Функциональные возможности

При запуске программы нас встречает FormLogin где можно выполнить регистрацию или вход. Окно регистрации и входа представлено на рисунке 10 и 11.

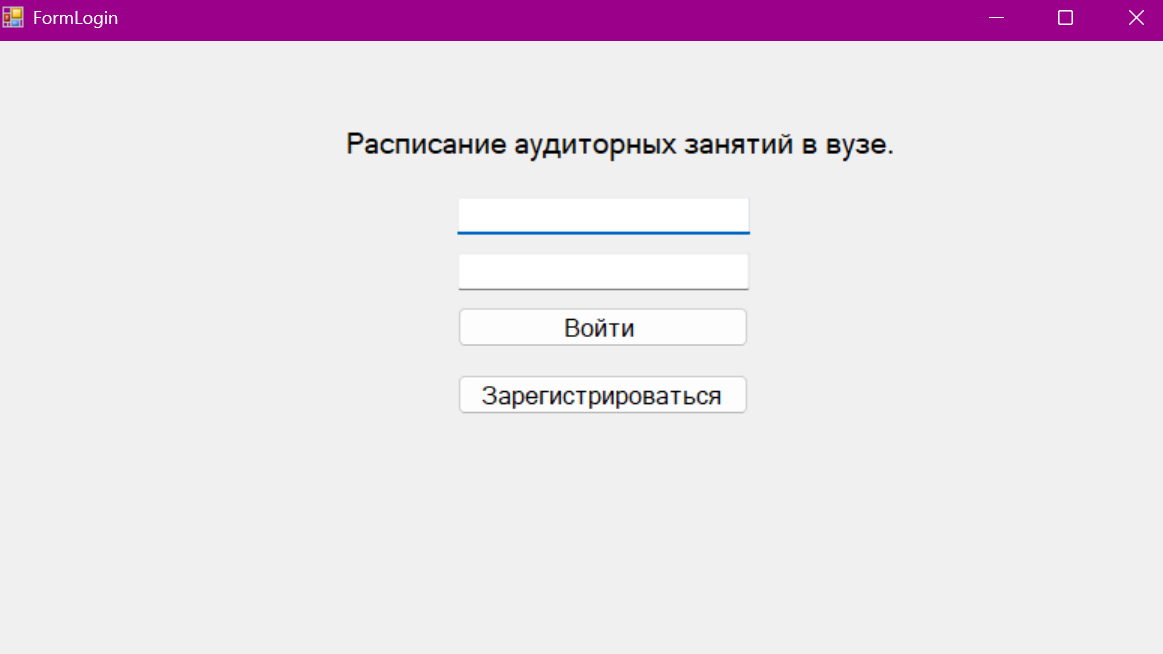


Рисунок 10 – Окно входа клиента

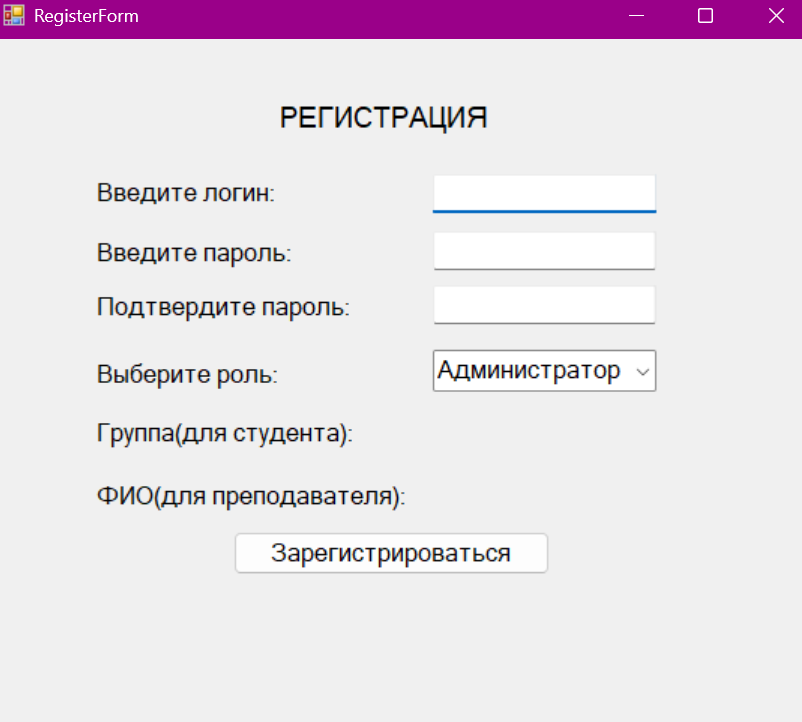


Рисунок 11 – Окно регистрации клиента

После регистрации или входа клиент под своим статусом попадает в свой личный кабинет, где может отслеживать расписание, изменять его и тд. Сейчас я продемонстрирую, как выглядит каждый отдел на следующих ресунках.

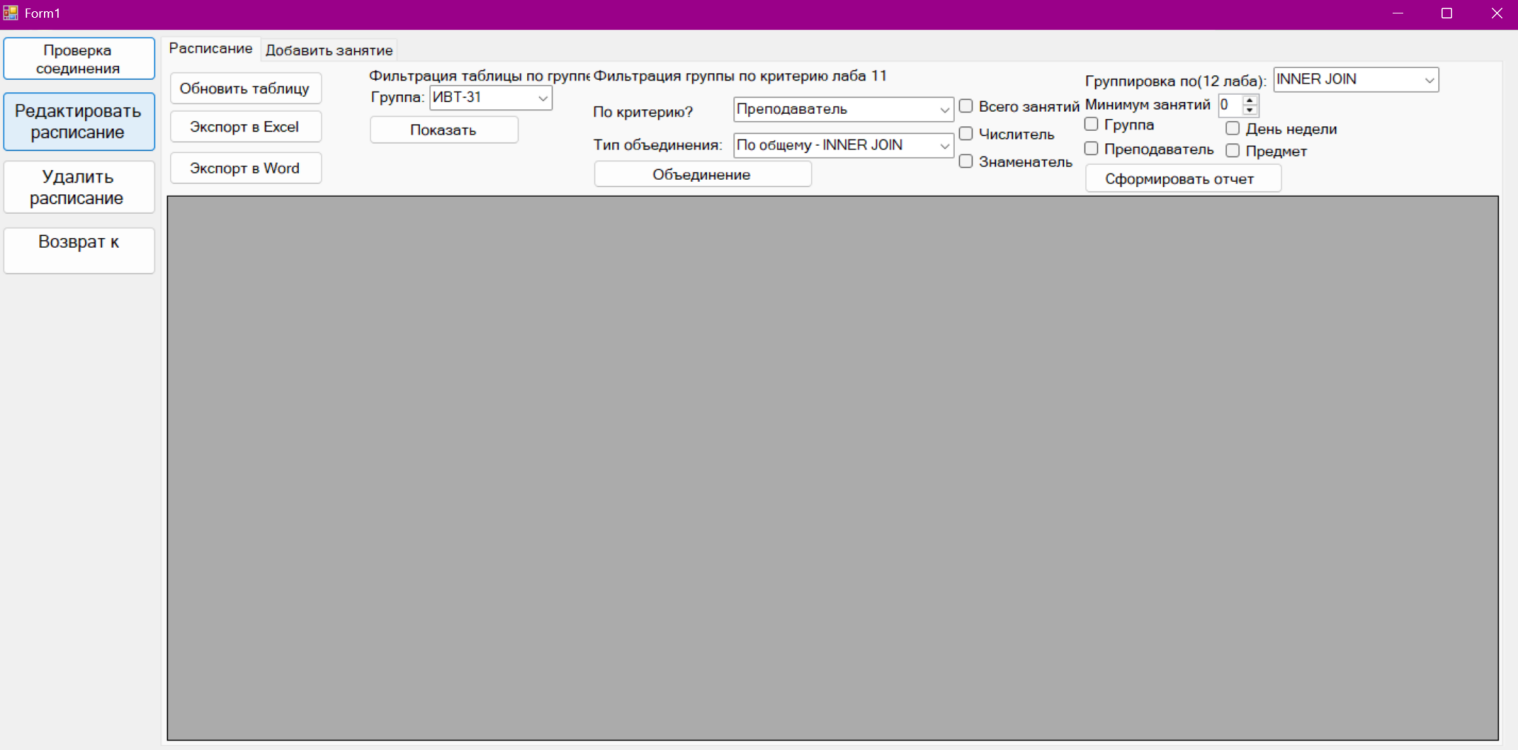


Рисунок 12 – Окно администратора

Главное отличие в том, что администратор может просматривать все расписания вуза не зависимо от преподавателя, факультета или же группы. Свободно редактировать, изменять и удалять. Здесь также осталась фильтрация из 11 и 12 лабораторных работ. Перейдем к окну преподавателя.

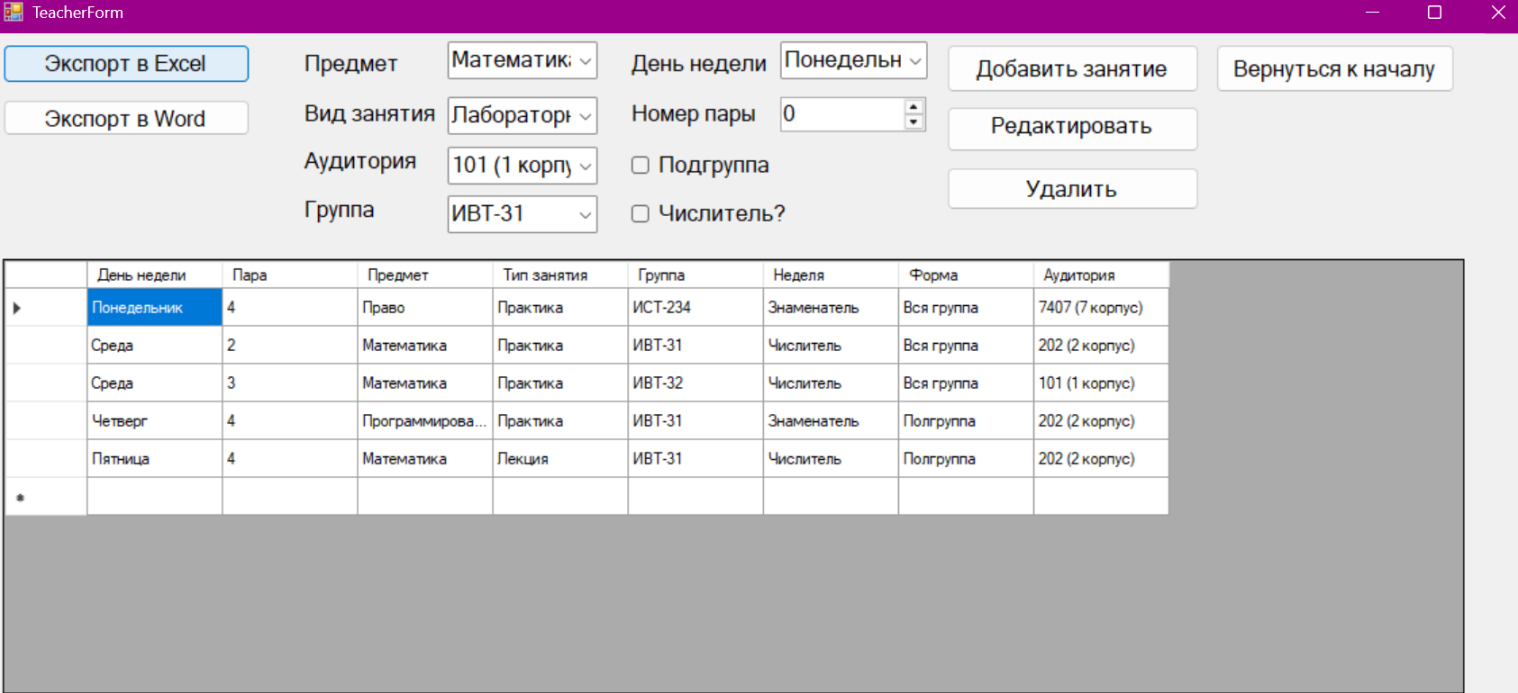


Рисунок 13 – Окно преподавателя

Преподаватель может видеть только свое расписание и свободно его изменять, добавлять и удалять. Перейдем к окну студента.

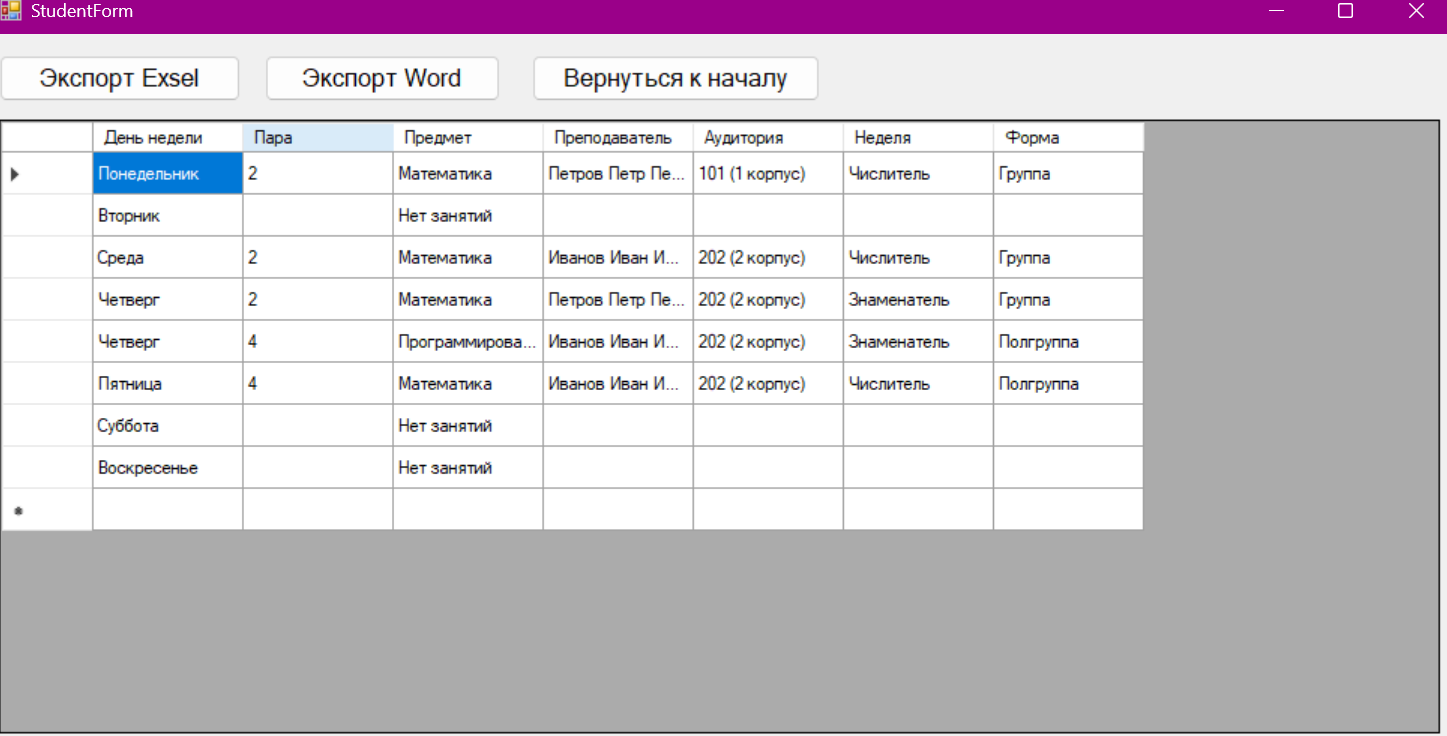


Рисунок 14 – Окно студента

Студент также, как и преподаватель может видеть только свое расписание. На этом функционал студента заканчивается. Дальше следует окно изменения расписания.

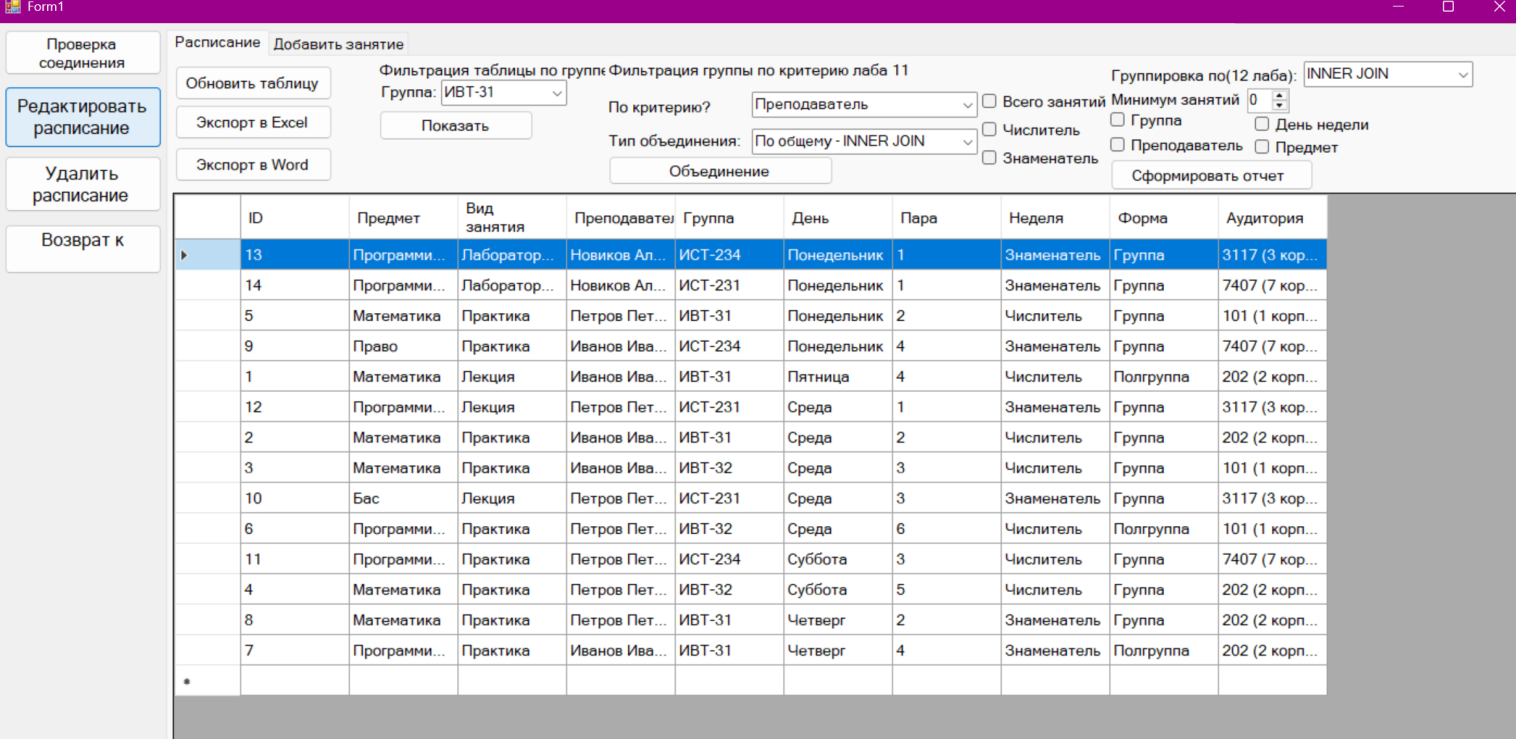


Рисунок 15 – Окно изменения расписания

На изображении показано расписание занятия у Новиков Алексей Андреевич с группой ИСТ-234. Поменяем преподавателя и группу на Иванов Иван Иванович и ИСТ-231. На рисунке 12 представлен результат выполнения запроса.

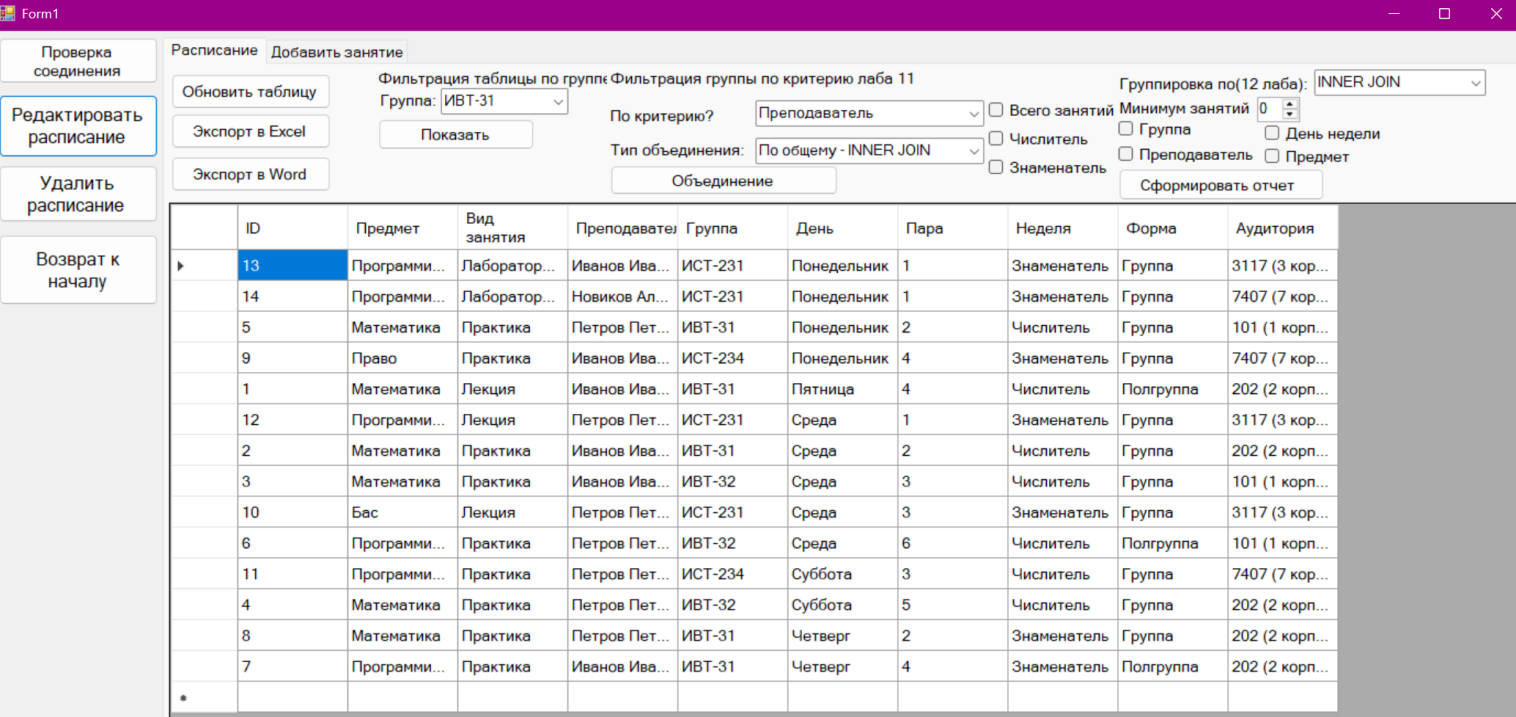


Рисунок 16 – Результат выполнения редактирования расписания.

Для реализации основных функций системы, таких как авторизация пользователей, управление расписанием и формирование отчетов, используются следующие SQL-запросы:

Таблица 2 – Основные SQL запросы системы

|  |  |
| --- | --- |
| SQL запрос | Назначение запроса |
| INSERT INTO Users (login, password, role\_id, teacher\_id, group\_id) VALUES (?, ?, ?, ?, ?) | Регистрация нового пользователя с указанием роли (администратор, преподаватель или студент) |
| SELECT r.name FROM Users u JOIN Roles r ON u.role\_id = r.role\_id WHERE u.login = ? | Получение роли пользователя для авторизации |
| SELECT s.schedule\_id, sub.name AS subject, t.last\_name + ' ' + t.first\_name AS teacher, g.name AS group, a.room\_number + ' (' + a.building + ')' AS auditorium, s.day\_of\_week, s.pair\_number FROM Schedule s JOIN Subjects sub ON s.subject\_id = sub.subject\_id JOIN Teachers t ON s.teacher\_id = t.teacher\_id JOIN Groups g ON s.group\_id = g.group\_id JOIN Auditoriums a ON s.auditorium\_id = a.auditorium\_id WHERE s.day\_of\_week = ? ORDER BY s.pair\_number | Получение расписания на определенный день недели |
| SELECT t.teacher\_id, t.last\_name + ' ' + t.first\_name AS teacher\_name, s.day\_of\_week, COUNT(\*) AS lessons\_count FROM Schedule s JOIN Teachers t ON s.teacher\_id = t.teacher\_id GROUP BY t.teacher\_id, t.last\_name, t.first\_name, s.day\_of\_week ORDER BY t.teacher\_id, s.day\_of\_week | Формирование отчета о нагрузке преподавателей |
| INSERT INTO Schedule (subject\_id, teacher\_id, auditorium\_id, group\_id, lesson\_type\_id, day\_of\_week, pair\_number, numerator, is\_halfgroup) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?) | Добавление нового занятия в расписание |
| UPDATE Schedule SET subject\_id = ?, teacher\_id = ?, auditorium\_id = ?, group\_id = ?, lesson\_type\_id = ?, day\_of\_week = ?, pair\_number = ?, numerator = ?, is\_halfgroup = ? WHERE schedule\_id = ? | Обновление информации о занятии |
| DELETE FROM Schedule WHERE schedule\_id = ? | Удаление занятия из расписания |
| SELECT g.group\_id, g.name, s.day\_of\_week, COUNT(\*) AS lessons\_count FROM Schedule s JOIN Groups g ON s.group\_id = g.group\_id GROUP BY g.group\_id, g.name, s.day\_of\_week ORDER BY g.group\_id, s.day\_of\_week | Формирование отчета о загрузке учебных групп |
| SELECT a.auditorium\_id, a.room\_number + ' (' + a.building + ')' AS auditorium, s.day\_of\_week, COUNT(\*) AS usage\_count FROM Schedule s JOIN Auditoriums a ON s.auditorium\_id = a.auditorium\_id GROUP BY a.auditorium\_id, a.room\_number, a.building, s.day\_of\_week ORDER BY a.auditorium\_id, s.day\_of\_week | Формирование отчета об использовании аудиторий |
| WITH CTE AS (SELECT \*, ROW\_NUMBER() OVER (PARTITION BY name ORDER BY subject\_id) AS rn FROM Subjects) DELETE FROM CTE WHERE rn > 1 | Удаление дубликатов дисциплин |

Благодаря этим запросам и осуществляется взаимодействие программы с базой данных. Использование четко структурированных методов и запросов обеспечило надёжную и удобную работу системы.