СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Во всем мире организации накапливают или уже накопили в процессе своей административно-хозяйственной деятельности большие объемы данных, в том числе и в электронном виде. Эти коллекции данных хранят в себе большие потенциальные возможности по извлечению новой аналитической информации, на основе которой можно и необходимо строить стратегию организации, выявлять тенденции развития рынка, находить новые решения, обусловливающие успешное развитие в условиях конкурентной борьбы. Для некоторых организаций такой анализ является неотъемлемой частью их повседневной деятельности, другие только начинают активно приступать к нему. Для хранения, упорядочения и анализа больших объемов информации предназначены комплексы средств, именуемых информационными системами (ИС). Одними из видов таких, сегодня уже автоматизированных информационных систем (АИС), являются базы данных (БД), управляемые с помощью систем управления базами данных (СУБД), и информационно-аналитические системою.

База данных – это упорядоченный набор структурированной информации или данных, которые обычно хранятся в электронном виде в компьютерной системе. База данных обычно управляется системой управления базами данных (СУБД). Данные вместе с СУБД, а также приложения, которые с ними связаны, называются системой баз данных, или, для краткости, просто базой данных.

Данные в наиболее распространенных типах современных баз данных обычно хранятся в виде строк и столбцов формирующих таблицу. Этими данными можно легко управлять, изменять, обновлять, контролировать и упорядочивать. В большинстве баз данных для записи и запросов данных используется язык структурированных запросов (SQL)[1].

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

Информационная система железнодорожной пассажирской станции неразрывно связана с базой данных. Эта связь обеспечивает точное отслеживание движения поездов, билетную систему и контроль безопасности. База данных играет ключевую роль в функционировании информационной системы пассажирской станции, храня информацию о расписании, пассажирах и грузах. Без надежной базы данных невозможно обеспечить плавное движение поездов, своевременное информирование пассажиров и оперативное реагирование на ситуации.

В данной курсовой работе поставлена задача базы данных и создания информационной системы железнодорожной пассажирской станции.

Для создания информационной базы данных будет использоваться СУБД 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 «Информационная система железнодорожной пассажирской станции», которая содержит следующее описание исходя из выданного задания к курсовой работе: работников железнодорожной станции можно подразделить на водителей подвижного состава, диспетчеров, ремонтников подвижного состава, путей, кассиров, работников службы подготовки составов, справочной службы и других, которые административно относятся каждый к своему отделу. Каждая из перечисленных категорий работников имеет уникальные атрибуты-характеристики, определяемые профессиональной направленностью. В отделах существует разбиение работников на бригады. Отделы возглавляются начальниками, которые представляют собой администрацию железнодорожной станции. В функции администрации входит планирование маршрутов, составление расписаний, формирование кадрового состава железнодорожной станции. За каждым локомотивом закрепляется локомотивная бригада. За несколькими локомотивами закрепляется бригада техников-ремонтников, выполняющая рейсовый и плановый техосмотр (по определенному графику), ремонт, техническое обслуживание. Водители локомотивов обязаны проходить каждый год медосмотр, не прошедших медосмотр необходимо перевести на другую работу. Локомотив должен своевременно осматриваться техниками-ремонтниками и при необходимости ремонтироваться. Подготовка к рейсу включает в себя техническую часть (рейсовый техосмотр, мелкий ремонт) и обслуживающую часть (уборка вагонов, запас продуктов питания и тому подобное).

В расписании указывается тип поезда (скорый, пассажирский), номер поезда, дни и время отправления и прибытия, маршрут (начальный и конечный пункты назначения, основные узловые станции), стоимость билета. Билеты на поезд можно приобрести заранее или забронировать в железнодорожных кассах. До отправления поезда, если есть необходимость, билет можно вернуть. Отправление поездов может быть задержано из-за опозданий поездов, погодных условий, технических неполадок.

Железнодорожные маршруты можно разделить на следующие категории: внутренние, международные, туристические, специальные маршруты. Пассажиры могут сдавать свои вещи в багажное отделение.

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

Выясним нужный набор атрибутов для каждой сущности с целью построения инфологической концептуальной модели. Концептуальная модель - это отражение предметной области, для которой разрабатывается база данных. Не вдаваясь в теорию, отметим, что это некая диаграмма с принятыми обозначениями элементов. Так, все объекты, обозначающие вещи, обозначаются в виде прямоугольника. Атрибуты, характеризующие объект - в виде овала, а связи между объектами - ромбами. Мощность связи обозначаются стрелками (в направлении, где мощность равна многим - двойная стрелка, а со стороны, где она равна единице - одинарная)[2].

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

Таблица 1.1 – Описание сущностей

Таблица	Поле	Ключ	Описание
Speciality	IdSpeciality	PK	Идентификационный номер таблицы Speciality
	SalaryOneRate		Зарплата за одну ставку
	Name		Наименование специальности
	Description		Описание специальности
TypeDepartment	IdTypeDepartment	PK	Идентификационный номер таблицы TypeDepartment
	NameType		Название типа отдела
Department	IdDepartment	PK	Идентификационный номер таблицы Department
	Name		Название отдела
	Description		Описание отдела
	CountBrigade		Количество бригад в отделе
	IdTypeDepartment		Идентификационный номер таблицы TypeDepartment
TypeEngine	IdTypeEngine	PK	Идентификационный номер таблицы TypeEngine
	NameType		Название типа двигателя
Engine	IdEngine	PK	Идентификационный номер таблицы Engine
	Name		Название двигателя
	Efficiency		КПД

Продолжение таблицы 1.1

Таблица	Поле	Ключ	Описание
	NumberOfCylinders		Количество цилиндров
	IdTypeEngine	FK	Идентификационный номер таблицы TypeEngine
Locomative	IdLocomative	PK	Идентификационный номер таблицы Locomative
	Name		Название локоматива
	Weight		Вес
	TractionForce		Сила тяги
	InspectionDate		Дата тех. обслуживания
	IdEngine	FK	Идентификационный номер таблицы Engine
Brigade	IdBrigade	PK	Идентификационный номер таблицы Brigade
	CountWorker		Количество работников в бригаде
	NumberInDepartment		Номер бригады в отделе
	IdDepartment	FK	Идентификационный номер таблицы Department
	IdLocomative	FK	Идентификационный номер таблицы Locomative
Worker	IdWorker	PK	Идентификационный номер таблицы Worker
	Name		Имя работника
	Surname		Фамилия работника
	Email		Email работника
	Rate		Тарифная ставка работника
	IdSpeciality	FK	Идентификационный номер таблицы Speciality
	IdBrigade	FK	Идентификационный номер таблицы Brigade
TypeTrain	IdTypeTrain	PK	Идентификационный номер таблицы TypeTrain
	NameType		Название типа поезда
Train	IdTrain	PK	Идентификационный номер таблицы Train
	Name		Имя поезда
	NumberOfTrainCarriage		Количество вагонов

Продолжение таблицы 1.1

Таблица	Поле	Ключ	Описание
	IdLocomative	FK	Идентификационный номер таблицы Locomative
	IdTypeTrain	FK	Идентификационный номер таблицы TypeTrain
Driver	IdDriver	PK	Идентификационный номер таблицы Driver
	DateOfMedical Examination		Дата прохождения медосмотра
	IdWorker	FK	Идентификационный номер таблицы Worker
TrainDriver	IdTrainDriver	PK	Идентификационный номер таблицы TrainDriver
	IdDriver	FK	Идентификационный номер таблицы Driver
	IdTrain	FK	Идентификационный номер таблицы Train
TypeTrain Carriage	IdTypeTrainCarriage	PK	Идентификационный номер таблицы TypeTrainCarriage
	NameType	FK	Название типа вагона
TrainCarriage	IdTrainCarriage	PK	Идентификационный номер таблицы TrainCarriage
	NumberOfSeats		Количество мест в вагоне
	IdTrain	FK	Идентификационный номер таблицы Train
	IdTypeTrainCarriage	FK	Идентификационный номер таблицы TypeTrainCarriage
Route	IdRoute	PK	Идентификационный номер таблицы Route
	NameRoute		Название маршрута
	DepartureTime		Время отправления
	ArravalTime		Время прибытия
Ticket	IdTicket	PK	Идентификационный номер таблицы Ticket
	PlaceofNumber		Номер места
	IsTaken		Состояние места
	TimeofPurchase		Время покупки билета
	IdRoute	FK	Идентификационный номер таблицы Route

Продолжение таблицы 1.1

Таблица	Поле	Ключ	Описание
Schedule	IdSchedule	PK	Идентификационный номер таблицы Schedule
	IdTrain	FK	Идентификационный номер таблицы Train
	IdRoute	FK	Идентификационный номер таблицы Route
Station	IdStation	PK	Идентификационный номер таблицы Station
	NameStation		Название станции
StationRoute	IdStationRoute	PK	Идентификационный номер таблицы StationRoute
	IdRoute	FK	Идентификационный номер таблицы Route
	IdStation	FK	Идентификационный номер таблицы Station
	DepartureTime		Время отправления
	ArravalTime		Время прибытия

1.3 Определение связей между объектами

Между двумя сущностями может быть установлена связь. Отношения между сущностями характеризуются глаголом, который можно применить для взаимодействия между ними. Связь – это некое отношение между двумя типами сущностей[3].

Внешний ключ (FK) – это столбец или сочетание столбцов, которое применяется для принудительного установления связи между данными в двух таблицах с целью контроля данных, которые могут храниться в таблице внешнего ключа[4]. Другими словами, это указатель на строку другой таблицы.

Связи позволяют по одной сущности находить другие сущности, связанные с ней.

Связи делятся на:

- Однозначная связь (One-to-One) – когда одна из таблиц ссылается на другую, но не наоборот. Например, таблица «Заказы» имеет внешний ключ, связанный с таблицей «Клиенты», что позволяет определить, какой клиент сделал заказ.
- Одноправленная связь (One-to-Many) – когда обе таблицы имеют внешние ключи, связанные друг с другом. Например, таблица «Авторы» имеет внешний ключ, связанный с таблицей «Книги», и таблица «Книги» также имеет

внешний ключ, связанный с таблицей «Авторы». Это позволяет найти авторов для конкретной книги и книги для конкретного автора.

– Множественные (Many-to-Many) связи – каждая запись в одной таблице может иметь несколько соответствующих записей в другой таблице, и наоборот. Например, множество студентов может быть зарегистрировано на множество курсов, и каждый курс может иметь множество студентов[5].

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

Инфологическая модель применяется на втором этапе проектирования БД после словесного описания предметной области. При создании информационных систем проект базы данных является фундаментом построения всей системы в целом. Следовательно, инфологическая модель должна включать формализованное описание предметной области, одинаково доступное для пользователей и специалистов по созданию БД.

Проектирование инфологической модели предметной области – частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели (entity-relationship model)[6].

Концептуальное проектирование БД – это процесс создания модели используемой информации, не зависящей от любых физических аспектов ее представления. Эта модель данных создается на основе информации, записанной в спецификациях требований пользователей. Концептуальное проектирование БД абсолютно не зависит от таких подробностей ее реализации, как тип выбранной целевой СУБД, набор создаваемых прикладных программ, используемые языки программирования, тип выбранной вычислительной платформы, а также от любых других особенностей физической реализации[7].

Чаще всего концептуальная модель базы данных включает в себя: описание информационных объектов или понятий предметной области и связей между ними; описание ограничений целостности, т.е. требований к допустимым значениям данных и к связям между ними.

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

Проектирование реляционной базы данных проходит в том же порядке, что и проектирование БД других моделей данных, но имеет свои особенности.

Проектирование схемы БД должно решать задачи минимизации дублирования данных и упрощения процедур их обработки и обновления.

Для разработки схемы реляционной базы данных необходимо определить набор таблиц, составляющих эту базу данных. Эти таблицы должны 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.

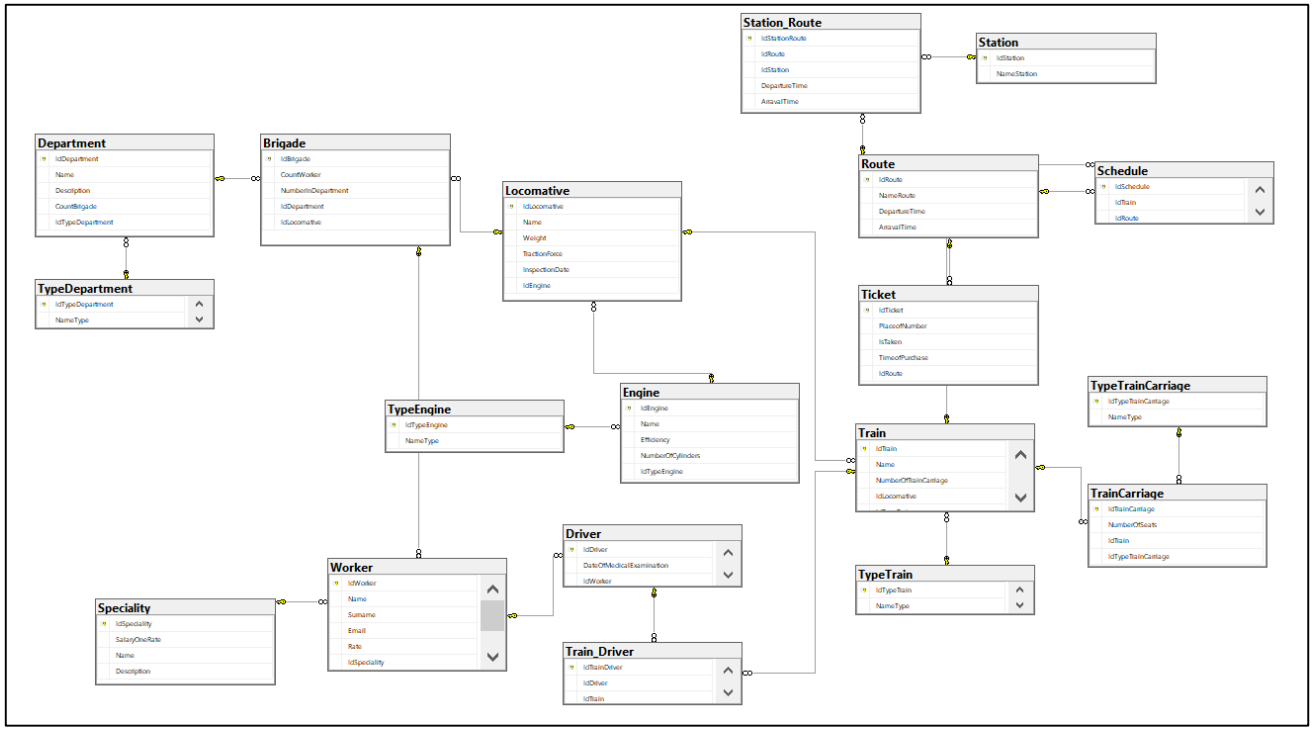

Рисунок 2.1 – Набор необходимых отношений базы данных

2.2 Задание первичных и внешних ключей определенных отношений

Ключи представляют способ идентификации строк в таблице. С помощью ключей мы также можем связывать строки между различными таблицами в отношения.

Первичный ключ (primary key) непосредственно применяется для идентификации строк в таблице. Он должен соответствовать следующим ограничениям:

- Первичный ключ должен быть уникальным все время.
- Он должен постоянно присутствовать в таблице и иметь значение.
- Он не должен часто менять свое значение. В идеале он вообще не должен изменять значение.

Как правило, первичный ключ представляет один столбец таблицы, но также может быть составным и состоять из нескольких столбцов[8].

Первичный ключ (англ. primary key) – в реляционной модели данных один из потенциальных ключей отношения, выбранный в качестве основного ключа (или ключа по умолчанию).

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

С точки зрения теории все потенциальные ключи отношения эквивалентны, то есть обладают одинаковыми свойствами уникальности и минимальности. Однако в качестве первичного обычно выбирается тот из потенциальных ключей, который наиболее удобен для тех или иных практических целей, например, для создания внешних ключей в других отношениях либо для создания кластерного индекса. Поэтому в качестве первичного ключа, как правило, выбирают тот, который имеет наименьший размер (физического хранения) или включает наименьшее количество атрибутов[9].

Внешний ключ (FK) – это столбец или сочетание столбцов, которое применяется для принудительного установления связи между данными в двух таблицах с целью контроля данных, которые могут храниться в таблице внешнего ключа. Если один или несколько столбцов, в которых находится первичный ключ для одной таблицы, упоминается в одном или нескольких столбцах другой таблицы, то в ссылке внешнего ключа создается связь между двумя таблицами. Этот столбец становится внешним ключом во второй таблице[4].

Поддержка внешних ключей также называется соблюдением ссылочной целостности. Реляционные СУБД поддерживают автоматический контроль ссылочной целостности.

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.

2.3 Третья нормальная форма

Нормализация – это процесс организации данных в базе данных, Она включает в себя создание таблиц и установление связей между ними в соответствии с правилами, разработанными как для защиты данных, так и для повышения гибкости базы данных, устраняя избыточность и несогласованную зависимость.

Избыточность данных приводит к непродуктивному расходованию свободного места на диске и затрудняет обслуживание баз данных. Например, если данные, хранящиеся в нескольких местах, потребуется изменить, в них придется внести одни и те же изменения во всех этих местах. Изменение адреса клиента проще реализовать, если эти данные хранятся только в таблице Customers и нигде в базе данных.

Первая нормальная форма

– Устраните повторяющиеся группы в отдельных таблицах.
– Создайте отдельную таблицу для каждого набора связанных данных.
– Идентифицируйте каждый набор связанных данных с помощью первичного ключа.

Не используйте несколько полей в одной таблице для хранения похожих данных. Например, для слежения за товаром, который закупается у двух разных поставщиков, можно создать запись с полями, определяющими код первого поставщика и код второго поставщика.

Что произойдет при добавлении третьего поставщика? Добавление поля не является ответом; он требует изменений в программе и таблице и не обеспечивает плавное размещение динамического числа поставщиков. Вместо этого можно поместить все сведения о поставщиках в отдельную таблицу Vendors (поставщики) и связать товары с поставщиками с помощью кодов товаров или поставщиков с товарами с помощью кодов поставщиков.

Вторая нормальная форма

– Создайте отдельные таблицы для наборов значений, относящихся к нескольким записям.
– Свяжите эти таблицы с помощью внешнего ключа.

Записи не должны зависеть от чего-либо, кроме первичного ключа таблицы (составного ключа, если это необходимо). Возьмем для примера адрес клиента в системе бухгалтерского учета. Этот адрес необходим не только таблице Customers, но и таблицам Orders, Shipping, Invoices, Accounts Receivable и Collections. Вместо того чтобы хранить адрес клиента как отдельный элемент в каждой из этих таблиц, храните его в одном месте: или в таблице Customers, или в отдельной таблице Addresses.

Третья нормальная форма

– Исключите поля, которые не зависят от ключа.

Значения в записи, которые не являются частью ключа этой записи, не принадлежат в таблице. Если содержимое группы полей может относиться более чем к одной записи в таблице, попробуйте поместить эти поля в отдельную таблицу.

Например, в таблицу Employee Recruitment (наем сотрудников) можно включить адрес кандидата и название университета, в котором он получил образование. Однако для организации групповой почтовой рассылки необходим полный список университетов. Если сведения об университетах будут храниться в таблице Candidates, составить список университетов при отсутствии кандидатов не получится. Таким образом, создайте вместо этого отдельную таблицу Universities и свяжите ее с таблицей Candidates при помощи ключа – кода университета[10].

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

Целостность базы данных – соответствие имеющейся в базе данных информации её внутренней логике, структуре и всем явно заданным правилам. Каждое правило, налагающее некоторое ограничение на возможное состояние базы данных, называется ограничением целостности.

Очевидно, что ограничения должны быть формально объявлены для СУБД, после чего СУБД должна предписывать их выполнение. Объявление ограничений сводится просто к использованию соответствующих средств языка базы данных, а соблюдение ограничений осуществляется с помощью контроля со стороны СУБД над операциями обновления, которые могут нарушить эти ограничения, и запрещения тех операций, которые их действительно нарушают. При первоначальном объявлении ограничения система должна проверить, удовлетворяет ли ему в настоящий момент база данных. Если это условие не соблюдается, ограничение должно быть отвергнуто; в противном случае оно принимается (то есть записывается в каталог системы) и начиная с этого момента соблюдается[11].

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

Ограничение целостности отношений заключается в том, что в любом отношении должны отсутствовать записи с одним и тем же значением первичного ключа. Конкретно требование состоит в том, что любая запись любого отношения должна быть отличной от любой другой записи этого отношения. Это требование автоматически удовлетворяется, если в системе не нарушаются базовые свойства отношений.

Ограничение целостности подразумевает, что в любом отношении не должны появляться записи с одним и тем же значением первичного ключа. То есть любая запись отношения должна быть отлична от любой другой записи этого же отношения. В проектируемой таблице все сущности будут иметь первичный ключ. Он необходим для её однозначной идентификации. Например, на клиентской стороне, также будет использоваться при удалении, добавлении или обновлении записи.

Условиями целостности называется набор правил, используемых для поддержания допустимых межтабличных связей и запрета на случайное изменение или удаление связанных данных.

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).

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.

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[12].

2.5 Графическое представление связей между внешними первичными ключами

По результатам 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

В системе управления базами данных SQL Server 2019 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.

В свою очередь стоит упомянуть о преимуществах использования SQL Server 2019 для разработки и управления базами данных. Например, SQL Server обладает мощными инструментами для обеспечения безопасности данных, такими как механизмы шифрования, аудита и управления доступом. Кроме того, SQL Server предлагает широкий спектр возможностей для оптимизации производительности баз данных, включая индексацию, хранение данных в памяти и оптимизацию запросов.

Также стоит отметить, что SQL Server 2019 поддерживает различные типы данных и функциональности, что делает его универсальным инструментом для создания разнообразных приложений. Благодаря активной поддержке и развитию со стороны Microsoft, SQL Server постоянно обновляется и совершенствуется, что обеспечивает надежность и эффективность работы с базами данных на долгосрочной основе.

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

Таблица 3.1 – Характеристики атрибутов таблицы Speciality

Поле	Тип данных	Описание
IdSpeciality	INT	Идентификационный номер таблицы Speciality
SalaryOneRate	MONEY	Зарплата за одну ставку
Name	VARCHAR(30)	Наименование специальности
Description	VARCHAR(100)	Описание специальности

Таблица 3.2 – Характеристики атрибутов таблицы TypeDepartment

Поле	Тип данных	Описание
IdTypeDepartment	INT	Идентификационный номер таблицы TypeDepartment
NameType	VARCHAR(50)	Название типа отдела

Таблица 3.3 – Характеристики атрибутов таблицы Department

Поле	Тип данных	Описание
IdDepartment	INT	Идентификационный номер таблицы Department
Name	VARCHAR(50)	Название отдела

Продолжение таблицы 3.3

Поле	Тип данных	Описание
Description	VARCHAR(150)	Описание отдела
CountBrigade	INT	Количество бригад в отделе
IdTypeDepartment	INT	Идентификационный номер таблицы TypeDepartment

Таблица 3.4 – Характеристики атрибутов таблицы TypeEngine

Поле	Тип данных	Описание
IdTypeEngine	INT	Идентификационный номер таблицы TypeEngine
NameType	VARCHAR(50)	Название типа двигателя

Таблица 3.5 – Характеристики атрибутов таблицы Engine

Поле	Тип данных	Описание
IdEngine	INT	Идентификационный номер таблицы Engine
Name	VARCHAR(20)	Название двигателя
Efficiency	FLOAT	КПД
NumberOfCylinders	INT	Количество цилиндров
IdTypeEngine	INT	Идентификационный номер таблицы TypeEngine

Таблица 3.6 – Характеристики атрибутов таблицы Locomative

Поле	Тип данных	Описание
IdLocomative	INT	Идентификационный номер таблицы Locomative
Name	VARCHAR(20)	Название локоматива
Weight	FLOAT	Вес
TractionForce	FLOAT	Сила тяги
InspectionDate	DATE	Дата тех. обслуживания
IdEngine	INT	Идентификационный номер таблицы Engine

Таблица 3.7 – Характеристики атрибутов таблицы Brigade

Поле	Тип данных	Описание
IdBrigade	INT	Идентификационный номер таблицы Brigade
CountWorker	INT	Количество работников в бригаде
NumberInDepartment	INT	Номер бригады в отделе
IdDepartment	INT	Идентификационный номер таблицы Department
IdLocomative	INT	Идентификационный номер таблицы Locomative

Таблица 3.8 – Характеристики атрибутов таблицы Worker

Поле	Тип данных	Описание
IdWorker	INT	Идентификационный номер таблицы Worker
Name	VARCHAR(50)	Имя работника
Surname	VARCHAR(50)	Фамилия работника
Email	VARCHAR(100)	Email работника
Rate	FLOAT	Тарифная ставка работника
IdSpeciality	INT	Идентификационный номер таблицы Speciality
IdBrigade	INT	Идентификационный номер таблицы Brigade

Таблица 3.9 – Характеристики атрибутов таблицы TypeTrain

Поле	Тип данных	Описание
IdTypeTrain	INT	Идентификационный номер таблицы TypeTrain
NameType	VARCHAR(10)	Название типа поезда

Таблица 3.10 – Характеристики атрибутов таблицы Train

Поле	Тип данных	Описание
IdTrain	INT	Идентификационный номер таблицы Train
Name	VARCHAR(20)	Имя поезда
NumberOfTrainCarriage	INT	Количество вагонов
IdLocomative	INT	Идентификационный номер таблицы Locomative
IdTypeTrain	INT	Идентификационный номер таблицы TypeTrain

Таблица 3.11 – Характеристики атрибутов таблицы Driver

Поле	Тип данных	Описание
IdDriver	INT	Идентификационный номер таблицы Driver
DateOfMedical Examination	DATE	Дата прохождения медосмотра
IdWorker	INT	Идентификационный номер таблицы Worker

Таблица 3.12 – Характеристики атрибутов таблицы TrainDriver

Поле	Тип данных	Описание
IdTrainDriver	INT	Идентификационный номер таблицы TrainDriver

Продолжение таблицы 3.12

Поле	Тип данных	Описание
IdDriver	INT	Идентификационный номер таблицы Driver
IdTrain	INT	Идентификационный номер таблицы Train

Таблица 3.13 – Характеристики атрибутов таблицы TypeTrainCarriage

Поле	Тип данных	Описание
IdTypeTrainCarriage	INT	Идентификационный номер таблицы TypeTrainCarriage
NameType	VARCHAR(20)	Название типа вагона

Таблица 3.14 – Характеристики атрибутов таблицы TrainCarriage

Поле	Тип данных	Описание
IdTrainCarriage	INT	Идентификационный номер таблицы TrainCarriage
NumberOfSeats	INT	Количество мест в вагоне
IdTrain	INT	Идентификационный номер таблицы Train
IdTypeTrainCarriage	INT	Идентификационный номер таблицы TypeTrainCarriage

Таблица 3.15 – Характеристики атрибутов таблицы Route

Поле	Тип данных	Описание
IdRoute	INT	Идентификационный номер таблицы Route
NameRoute	VARCHAR(20)	Название маршрута
DepartureTime	DATETIME	Время отправления
ArravalTime	DATETIME	Время прибытия

Таблица 3.16 – Характеристики атрибутов таблицы Ticket

Поле	Тип данных	Описание
IdTicket	INT	Идентификационный номер таблицы Ticket
PlaceofNumber	INT	Номер места
IsTaken	BIT	Состояние места
TimeofPurchase	DATETIME	Время покупки билета
IdRoute	INT	Идентификационный номер таблицы Route

Таблица 3.17 – Характеристики атрибутов таблицы Schedule

Поле	Тип данных	Описание
IdSchedule	INT	Идентификационный номер таблицы Schedule

Продолжение таблицы 3.17

Поле	Тип данных	Описание
IdTrain	INT	Идентификационный номер таблицы Train
IdRoute	INT	Идентификационный номер таблицы Route

Таблица 3.18 – Характеристики атрибутов таблицы Station

Поле	Тип данных	Описание
IdStation	INT	Идентификационный номер таблицы Station
NameStation	VARCHAR(60)	Название станции

Таблица 3.19 – Характеристики атрибутов таблицы StationRoute

Поле	Тип данных	Описание
IdStationRoute	INT	Идентификационный номер таблицы StationRoute
IdRoute	INT	Идентификационный номер таблицы Route
IdStation	INT	Идентификационный номер таблицы Station
DepartureTime	DATETIME	Время отправления
ArravalTime	DATETIME	Время прибытия

В результате было создано 19 таблиц и атрибуты для каждой из них, для реализации информационной системы железнодорожной пассажирской станции.

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 – Получить перечень и общее число всех работников железнодорожной станции, начальников отделов, работников указанного отдела, по стажу работы на станции, половому признаку, возрасту, признаку наличия и количества детей, размеру заработной платы

```
SELECT Worker.Name, Surname, Department.Name, Speciality.Name AS
Speciality, Speciality.SalaryOneRate,
          (SELECT COUNT(*) FROM Worker w
        JOIN Brigade b ON w.IdBrigade = b.IdBrigade
        WHERE    b.IdDepartment    =    Brigade.IdDepartment)    AS
     TotalWorkersInDepartment
FROM Worker
JOIN Brigade
ON Worker.IdBrigade = Brigade.IdBrigade
JOIN Department
ON Brigade.IdDepartment = Department.IdDepartment
JOIN Speciality
ON Worker.IdSpeciality = Speciality.IdSpeciality
WHERE Speciality.SalaryOneRate * Worker.Rate > 40000
```

Листинг 4.2 – Получить перечень и общее число работников в бригаде, по всем отделам, в указанном отделе, обслуживающих некоторый локомотив, по возрасту, суммарной (средней) зарплате в бригаде

```
SELECT     Worker.Name,     Surname,     Brigade.NumberInDepartment,
Department.Name, Speciality.SalaryOneRate,
     (SELECT COUNT(*) FROM Brigade b
     WHERE     b.IdDepartment     =     Brigade.IdDepartment)     AS
     TotalBrigadesInDepartment
FROM Worker
JOIN Brigade
ON Worker.IdBrigade = Brigade.IdBrigade
JOIN Department
ON Brigade.IdDepartment = Department.IdDepartment
JOIN Speciality
ON Worker.IdSpeciality = Speciality.IdSpeciality
WHERE  Speciality.SalaryOneRate  >  (SELECT  AVG(SalaryoneRate)  FROM
Speciality)
```

Листинг 4.3 – Получить перечень и общее число водителей локомотивов, прошедших медосмотр либо не прошедших медосмотр в указанный год, по половому признаку, возрасту, размеру заработной платы

```
SELECT     Worker.Name,     Surname,     DateOfMedicalExamination,
Speciality.SalaryOneRate,(SELECT COUNT(*) FROM Driver
     WHERE DateOfMedicalExamination < '2024-01-01')
```

Продолжение листинга 4.3

```
FROM Driver
JOIN Worker
ON Driver.IdWorker = Worker.IdWorker
JOIN Speciality
ON Worker.IdSpeciality = Speciality.IdSpeciality
WHERE DateOfMedicalExamination < '2024-01-01'
```

Листинг 4.4 – Получить перечень и общее число локомотивов, приписанных к железнодорожной станции, находящихся на ней в указанное время, по времени прибытия на станции, по количеству совершенных маршрутов

```
SELECT DISTINCT Train.Name
FROM Locomative
JOIN Train
ON Locomative.IdLocomative = Train.IdLocomative
JOIN Schedule
ON Train.IdTrain = Schedule.IdTrain
JOIN Station_Route
ON Station_Route.IdRoute = Schedule.IdRoute
WHERE '2024-03-19 18:00:00' BETWEEN (SELECT ArravalTime FROM
Station_Route
                                    WHERE Station_Route.IdRoute =
Schedule.IdRoute AND DepartureTime IS null) AND
                                    (SELECT DepartureTime FROM
Station_Route
                                    WHERE Station_Route.IdRoute =
Schedule.IdRoute AND ArravalTime IS null)
```

Листинг 4.5 – Получить перечень и общее число локомотивов, прошедших плановый техосмотр за определенный период времени, отправленных в ремонт в обозначенное время, ремонтированных указанное число раз, по количеству совершенных рейсов до ремонта, по возрасту локомотива

```
SELECT Locomative.Name, Weight, InspectionDate,
     (SELECT COUNT(*) FROM Locomative
     WHERE InspectionDate BETWEEN '2023-01-01' AND '2024-07-05')
FROM Locomative
WHERE InspectionDate BETWEEN '2023-01-01' AND '2024-07-05'
```

Листинг 4.6 – Получить перечень и общее число поездов на указанном маршруте, по длительности маршрута, по цене билета и по всем этим критериям сразу

```
SELECT DISTINCT Name, (DATEDIFF(minute,
                              (SELECT ArravalTime FROM Station_Route
                              WHERE Station_Route.IdRoute =
Schedule.IdRoute AND DepartureTime IS NULL
                              ),
                              (SELECT DepartureTime FROM
Station_Route
                              WHERE Station_Route.IdRoute =
Schedule.IdRoute AND ArravalTime IS NULL
                              )))
```

Продолжение листинга 4.6

```
FROM Train
JOIN Schedule
ON Train.IdTrain = Schedule.IdTrain
JOIN Station_Route
ON Station_Route.IdRoute = Schedule.IdRoute
JOIN Route
ON Station_Route.IdRoute = Route.IdRoute
WHERE Route.NameRoute = '605B'
```

Листинг 4.7 – Получить перечень и среднее количество проданных билетов за указанный интервал времени на определённые маршруты, по длительности маршрута, по цене билета

```
SELECT IdTicket, PlaceofNumber, NameRoute,
                                                    (SELECT COUNT(*)
FROM  Ticket
                                                    JOIN Route
                                                    ON Ticket.IdRoute
= Route.IdRoute
WHERE TimeofPurchase BETWEEN '2024-03-10' AND '2024-03-17' AND
Route.NameRoute = '605B')
FROM Ticket
JOIN Route
ON Route.IdRoute = Ticket.IdRoute
WHERE  TimeofPurchase  BETWEEN  '2024-03-10'  AND  '2024-03-17'  AND
Route.NameRoute = '605B'
```

Листинг 4.8 – Получить перечень и общее число маршрутов указанной категории, следующих в определенном направлении

```
SELECT Route.NameRoute,
                          (SELECT COUNT(*)
                          FROM Route
                          JOIN Station_Route
                          ON Route.IdRoute = Station_Route.IdRoute
                          JOIN Station
                          ON Station.IdStation =
Station_Route.IdStation
                          WHERE Station.NameStation = 'Polotsk'
                          )
FROM Route
JOIN Station_Route
ON Route.IdRoute = Station_Route.IdRoute
JOIN Station
ON Station.IdStation = Station_Route.IdStation
WHERE Station.NameStation = 'Polotsk'
```

Листинг 4.9 – Получить перечень и общее число пассажиров на указанном рейсе, уехавших в указанный день, уехавших за границу в указанный день, по признаку сдачи вещей в багажное отделение, по половому признаку, по возрасту

```
SELECT DISTINCT IdTicket, (SELECT COUNT(DISTINCT IdTicket)
```

Продолжение листинга 4.9

```
                                    FROM Ticket
                                    JOIN Route
                                    ON Route.IdRoute = Ticket.IdRoute
                                    JOIN Station_Route
                                    ON Route.IdRoute =
Station_Route.IdRoute
                                    JOIN Station
                                    ON Station.IdStation =
Station_Route.IdStation
                                    WHERE Route.NameRoute = '605B'
AND Ticket.IsTaken = 1 AND (SELECT Station_Route.ArravalTime

                FROM Station_Route

                JOIN Route

                ON Route.IdRoute = Station_Route.IdRoute

                WHERE Station_Route.DepartureTime IS NULL AND
Route.NameRoute = '605B') BETWEEN '2024-03-19 00:00:00' and '2024-
03-20 00:00:00')
FROM Ticket
JOIN Route
ON Route.IdRoute = Ticket.IdRoute
JOIN Station_Route
ON Route.IdRoute = Station_Route.IdRoute
JOIN Station
ON Station.IdStation = Station_Route.IdStation
WHERE Route.NameRoute = '605B' AND Ticket.IsTaken = 1 AND (SELECT
Station_Route.ArravalTime
                                                    FROM Station_Route
                                                    JOIN Route
                                                    ON Route.IdRoute =
Station_Route.IdRoute
                                                    WHERE
Station_Route.DepartureTime IS NULL AND Route.NameRoute = '605B')
BETWEEN '2024-03-19 00:00:00' and '2024-03-20 00:00:00'
```

Листинг 4.10 – Получить перечень и общее число невыкупленных билетов на указанном рейс, день, некоторый маршрут

```
SELECT DISTINCT(IdTicket),
                                (SELECT COUNT(DISTINCT IdTicket)
                                FROM Ticket
                                JOIN Route
                                ON Ticket.IdRoute = Route.IdRoute
                                JOIN Station_Route
                                ON Route.IdRoute =
Station_Route.IdRoute
                                WHERE Route.NameRoute = '605B' AND
Ticket.IsTaken = 'false' AND (SELECT Station_Route.ArravalTime
```

Продолжение листинга 4.10

```
                                FROM Station_Route
                                JOIN Route
                                ON Route.IdRoute =
Station_Route.IdRoute
                                WHERE Station_Route.DepartureTime IS
NULL AND Route.NameRoute = '605B') BETWEEN '2024-03-19 00:00:00'
and '2024-03-20 00:00:00'
                                )
FROM Ticket
JOIN Route
ON Ticket.IdRoute = Route.IdRoute
JOIN Station_Route
ON Route.IdRoute = Station_Route.IdRoute
WHERE Route.NameRoute = '605B' AND Ticket.IsTaken = 'false' AND
(SELECT Station_Route.ArravalTime
                                                FROM Station_Route
                                                JOIN Route
                                                ON Route.IdRoute =
Station_Route.IdRoute
                                                WHERE
Station_Route.DepartureTime IS NULL AND Route.NameRoute = '605B')
BETWEEN '2024-03-19 00:00:00' and '2024-03-20 00:00:00'
```

5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

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

Был выбран именно этот язык по следующим причинам:

– Претендует на подлинную объектную ориентированность (а всякая языковая сущность претендует на то, чтобы быть объектом).

– Призван практически реализовать компонентно-ориентированный подход к программированию, который способствует меньшей машинно-архитектурной зависимости результирующего программного кода, большей гибкости, переносимости и легкости повторного использования как программ целиком, так и ее фрагментов.

– Совместимость с базами данных на SQL Server 2019.

Для объединения базы данных с интерфейсом приложения использовался Entity Framework Core.

Entity Framework представляет ORM-технологию (object-relational mapping - отображения данных на реальные объекты) от компании Microsoft для доступа к данным. Entity Framework Core позволяет абстрагироваться от самой базы данных и ее таблиц и работать с данными как с объектами классом независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Как технология доступа к данным Entity Framework Core работает поверх платформы .NET и поэтому может использоваться на различных платформах стека .NET. Это и стандартные платформы типа Windows Forms, консольные приложения, WPF, UWP и ASP.NET Core. При этом кроссплатформенная природа EF Core позволяет задействовать ее не только на ОС Windows, но и на Linux и Mac OS.

Поскольку Entity Framework Core работает на основе платформы .NET, то он развивается вместе с данной платформой. Текущая версия EF Core - 8.0 была выпущена в ноябре 2023 года вместе с .NET 8. И технология продолжает развиваться.

Entity Framework Core поддерживает множество различных систем баз данных. Таким образом, мы можем через EF Core работать с любой СУБД, если для нее имеется нужный провайдер. По умолчанию на данный момент Microsoft предоставляет ряд встроенных провайдеров: для работы с MS SQL Server, для SQLite, для PostgreSQL. Также имеются провайдеры от сторонних поставщиков, например, для MySQL[13].

Выбор СУБД – важный этап при реализации БД. Существующие критерии, по которым можно оценивать СУБД условно делятся на три большие области:

– функциональность;
– особенности разработки приложений;
– смешанные критерии.

Условность такого деления определяется тем, что некоторые особенности конкретной СУБД проявляются не только в одной области, в которой производится оценка, но и влияют на другие.

Существуют еще две области оценок СУБД:
– поддерживаемые платформы;
– производительность.

Рассмотрим критерии оценки объектных СУБД с точки зрения разработки высокоэффективных приложений.

Проведенное исследование позволяет разбить эту область на пять групп, в которых можно с большой степенью независимости анализировать характеристики СУБД и оценивать их влияние на возможные свойства разрабатываемых приложений:
– особенности объектной модели;
– архитектура СУБД;
– механизмы СУБД;
– программирование интерфейса приложений;
– запросы к объектам.

На основе вышесказанного была выбрана система SQL Server. Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия; конкурирует с другими СУБД в этом сегменте рынка.

MS SQL Server – это платформа для решения критически важных задач в масштабе предприятия, обладающая высокой доступностью, повышенной производительностью и безопасностью. Решение представляет собой хорошо масштабируемый, полностью реляционный, быстродействующий сервер, способный обрабатывать большие объемы данных для клиент-серверных приложений[14].

Расширенные функции безопасности, в сочетании со встроенными, удобными для использования инструментами и управляемым доступом к данным, позволяют организации выполнить требования строгих политик соответствия нормам.

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 перехода по представленным пунктам системы, с помощью которых можно просматривать всю информацию, касающейся станций, локомотивов, работников и так далее.

При запуске программы пользователь попадает на главную страницу программы, на которой расположены все пункты управления данными железнодорожной станции.

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

При проектировании программы были учтены все возможные случаи некорректной работы программы, поэтому большинство нештатных ситуаций сопровождается оповещениями с описанием проблемы.

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

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

Графический редактор позволяет в графическом виде представить создаваемое окно и в принципе упрощает работу с графическими компонентами.

Для открытия окна в режиме графического дизайнера необходимо в структуре проекта нажать на файл окна, либо левой кнопкой мыши двойным кликом, либо правой кнопкой мыши. В появившемся контекстном меню необходимо выбрать View Designer.

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 из названия всех таблиц. Данные пункты выполнены в виде вкладок. По переходу на каждую вкладку, отображается соответствующая таблица.

6.3 Выполнение программного кода на языке C#

Далее следует описание работы программы с базой данных. Все необходимые методы для работы с базами данных описаны в классе RailwayStationContext и связываются с базой данных при помощи того же класса, используя статический метод. Подключение к базе данных начинается с метода OnConfiguring. Код класса RailwayStationContext представлен в листинге 6.1.

Листинг 6.1 – Методы для работы с базой данных

```
using System;
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using RailwayStation.Models.Entities;

namespace RailwayStation.DBContext;

public partial class RailwayStationContext : DbContext
{
    private static RailwayStationContext _context;
    public RailwayStationContext()
    {
    }

    public
RailwayStationContext(DbContextOptions<RailwayStationContext>
options)
        : base(options)
    {
    }
    public static RailwayStationContext Context
    {
        get {
            if (_context == null)
                _context = new RailwayStationContext();
            return _context;
        }
    }

    public virtual DbSet<Brigade> Brigades { get; set; }

    public virtual DbSet<Department> Departments { get; set; }

    public virtual DbSet<Driver> Drivers { get; set; }

    public virtual DbSet<Engine> Engines { get; set; }

    public virtual DbSet<Locomative> Locomatives { get; set; }

    public virtual DbSet<Route> Routes { get; set; }
```

```
    public virtual DbSet<Schedule> Schedules { get; set; }

    public virtual DbSet<Speciality> Specialities { get; set; }

    public virtual DbSet<Station> Stations { get; set; }

    public virtual DbSet<StationRoute> StationRoutes { get; set; }

    public virtual DbSet<Ticket> Tickets { get; set; }

    public virtual DbSet<Train> Trains { get; set; }

    public virtual DbSet<TrainCarriage> TrainCarriages { get; set; }

    public virtual DbSet<TrainDriver> TrainDrivers { get; set; }

    public virtual DbSet<TypeDepartment> TypeDepartments { get; set;
}

    public virtual DbSet<TypeEngine> TypeEngines { get; set; }

    public virtual DbSet<TypeTrain> TypeTrains { get; set; }

    public virtual DbSet<TypeTrainCarriage> TypeTrainCarriages {
get; set; }

    public virtual DbSet<Worker> Workers { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder
optionsBuilder)      => optionsBuilder.UseSqlServer("Server=DESKTOP-
501ABTG;Database=RailwayStation;Trusted_Connection=True;Encrypt=fal
se;TrustServerCertificate=true;");

    protected     override     void     OnModelCreating(ModelBuilder
modelBuilder)
    {
        modelBuilder.Entity<Brigade>(entity =>
        {
            entity.HasKey(e                                          =>
e.IdBrigade).HasName("PK__Brigade__9F27544737715AC1");

            entity.ToTable("Brigade");

            entity.HasOne(d => d.IdDepartmentNavigation).WithMany(p
=> p.Brigades)
                .HasForeignKey(d => d.IdDepartment)
                .OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK__Brigade__IdDepar__45F365D3");
```

```
            entity.HasOne(d => d.IdLocomativeNavigation).WithMany(p
=> p.Brigades)
                .HasForeignKey(d => d.IdLocomative)

.HasConstraintName("FK__Brigade__IdLocom__46E78A0C");
        });

        modelBuilder.Entity<Station>(entity =>
        {
            entity.HasKey(e                                          =>
e.IdStation).HasName("PK__Station__FBB9BA0A5BAD0298");

            entity.ToTable("Station");

            entity.Property(e => e.NameStation)
                .HasMaxLength(60)
                .IsUnicode(false);
        });

        modelBuilder.Entity<Department>(entity =>
        {
            entity.HasKey(e                                          =>
e.IdDepartment).HasName("PK__Departme__DF1E6E4BB6AA6241");

            entity.ToTable("Department");

            entity.Property(e => e.Description)
                .HasMaxLength(150)
                .IsUnicode(false);
            entity.Property(e => e.Name)
                .HasMaxLength(30)
                .IsUnicode(false);

            entity.HasOne(d                                          =>
d.IdTypeDepartmentNavigation).WithMany(p => p.Departments)
                .HasForeignKey(d => d.IdTypeDepartment)
                .OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK__Departmen__IdTyp__3B75D760");
        });

        modelBuilder.Entity<Worker>(entity =>
        {
            entity.HasKey(e                                          =>
e.IdWorker).HasName("PK__Worker__18714E8AEE1978D7");

            entity.ToTable("Worker");

            entity.Property(e => e.Email)
                .HasMaxLength(100)
```

Продолжение листинга 6.1

```
                .IsUnicode(false);
            entity.Property(e => e.Name)
                .HasMaxLength(50)
                .IsUnicode(false);
            entity.Property(e => e.Surname)
                .HasMaxLength(50)
                .IsUnicode(false);

            entity.HasOne(d => d.IdBrigadeNavigation).WithMany(p =>
p.Workers)
                .HasForeignKey(d => d.IdBrigade)
                .OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK__Worker__IdBrigad__4AB81AF0");

            entity.HasOne(d => d.IdSpecialityNavigation).WithMany(p
=> p.Workers)
                .HasForeignKey(d => d.IdSpeciality)
                .OnDelete(DeleteBehavior.ClientSetNull)

.HasConstraintName("FK__Worker__IdSpecia__49C3F6B7");
        }
);

        OnModelCreatingPartial(modelBuilder);
    }

    partial void OnModelCreatingPartial(ModelBuilder modelBuilder);
}
```

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 позволяет без проблем хранить и извлекать нужную информацию. Данная база данных соответствует всем требованиям, которые предъявляются в задании. Разработанная система реагирует на ошибочный ввод данных, а также 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. А также не допустить выполнении некорректных действий с базой данных.

Разработанная информационная система железнодорожной пассажирской станции удовлетворяет всем требованиям комфортного использования и обеспечивает беспроблемное хранение и изменение информации.

В результате выполнения курсовой работы были спроектированы и реализованы: база данных информационной системы железнодорожной пассажирской станции, программа, которая эффективно взаимодействует с базой данных. Были реализованы такие сущности, как отделы, бригады, работники, специальность и другие, для работы с данной системой. Программное средство реализовано с помощью языка программирования C#. Среда разработки для реализации данного курсовой работы, была Visual Studio 2022. Для объединения базы данных с интерфейсом приложения использовался Entity Framework Core.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Что такое базы данных [Электронный ресурс]. Режим доступа https://www.oracle.com/cis/database/what-is-database/. Дата доступа 21.02.2024.

2 Концептуальная модель базы данных [Электронный ресурс]. Режим доступа: https://www.oracle.com/cis/database/what-is-database/. Дата доступа: 22.11.2022.

3 ER-модель Понятие связи. [Электронный ресурс]. Режим доступа: hhttps://www.bestprog.net/ru/2019/01/27/er-model-the-concept-of-relationship-the-relationship-capacity-types-of-relationships-examples-ru/. Дата доступа 22.11.2022.

4 Ограничения первичных и внешних ключей [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/sql/relational-databases/tables/primary-and-foreign-key-constraints?view=sql-server-ver16. Дата доступа 22.11.2022.

5 Связи в базах данных [Электронный ресурс]. Режим доступа: https://foxminded.ua/ru/svyaz-v-baze-dannyh/. Дата доступа 19.11.2022.

6 Инфологическая модель данных. [Электронный ресурс]. Режим доступа: https://eduherald.ru/ru/article/view?id=20048. Дата доступа 20.11.2022.

7 Исследование современных методов проектирования базы данных [Электронный ресурс]. Режим доступа: https://natural-sciences.ru/ru/article/view?id=27081#:~:text=%D0%9A%D0%BE%D0%BD%D1%86%D0%B5%D0%BF%D1%82%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%91%D0%94%20%2D%20%D1%8D%D1%82%D0%BE%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81,%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%BD%D0%BE%D0%B9%20%D0%B2%20%D1%81%D0%BF%D0%B5%D1%86%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%86%D0%B8%D1%8F%D1%85%20%D1%82%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B9%20%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D0%B5%D0%B9. Дата доступа: 22.11.2022.

8 Ключи. [Электронный ресурс]. Режим доступа: https://metanit.com/sql/tutorial/1.2.php. Дата доступа 22.11.2022.

9 Первичный ключ [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B2%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BB%D1%8E%D1%87 Дата доступа 22.11.2022.

10 Описание основных приемов нормализации базы данных [Электронный ресурс]. Режим доступа: https://learn.microsoft.com/ru-ru/office/troubleshoot/access/database-normalization-description. Дата доступа 19.11.2022.

11 Целостность базы данных. [Электронный ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/%D0%A6%D0%B5%D0%BB%D0%BE%D1%81%D1%82%D0%BD%D0%BE%D1%81%D1%82%D1%8C_%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85. Дата доступа 20.11.2022.

12 Определение ограничений целостности для внешних ключей отношений и для отношений в целом. [Электронный ресурс]. Режим доступа: https://prog.bobrodobro.ru/58005. Дата доступа 22.11.2022.

13 Введение в Entity Framework Core [Электронный ресурс]. Режим доступа: https://metanit.com/sharp/efcore/1.1.php Дата доступа 22.11.2022.

14 Microsoft SQL Server [Электронный ресурс]. Режим доступа: https://ipos.by/server/. Дата доступа 19.11.2022.

ПРИЛОЖЕНИЕ А
(обязательное)
Концептуальная схема БД

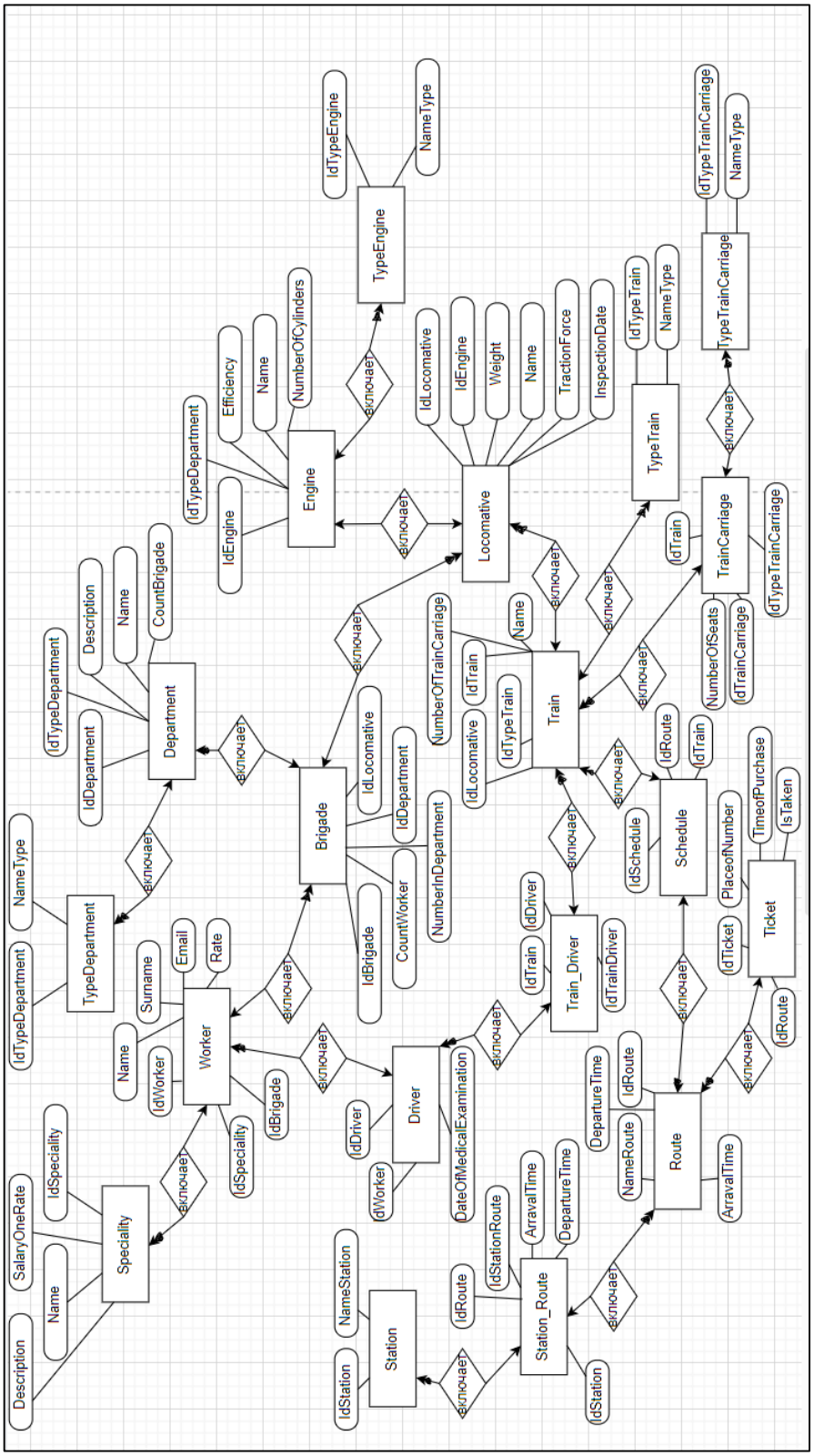

Рисунок А.1 – Концептуальная модель проектируемой базы данных

ПРИЛОЖЕНИЕ Б
(обязательное)
Схема реляционной базы данных

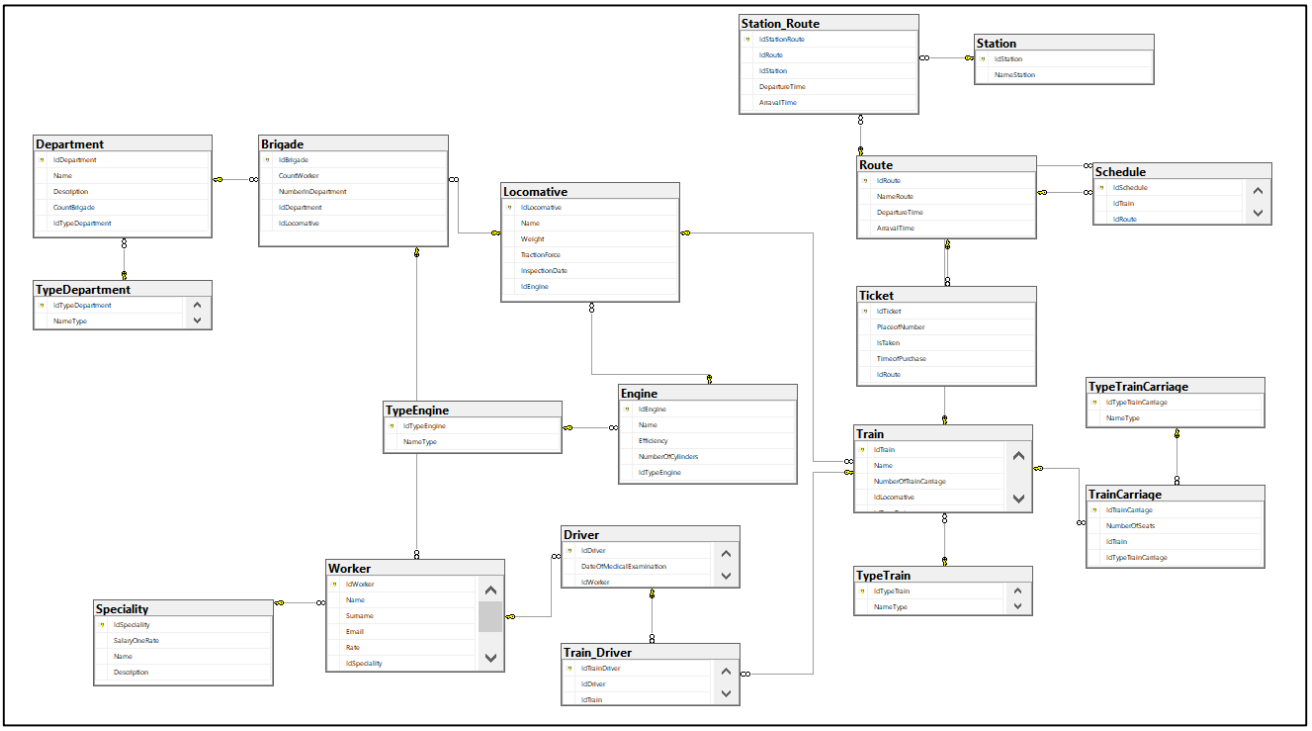

Рисунок Б.1 – Схема реляционной базы данных

ПРИЛОЖЕНИЕ В
(обязательное)
Описание задания курсовой работы

Реализованная база данных должна состоять не менее чем из 15-и таблиц, а также выполнять все указанные в варианте запросы. Кроме того, в программе, при выводе данных, пользователь не должен видеть уникальные идентификаторы. Добавление, удаление, редактирование и поиск данных также не должно осуществляться с помощью уникальных идентификаторов.

Работников железнодорожной станции можно подразделить на водителей подвижного состава, диспетчеров, ремонтников подвижного состава, путей, кассиров, работников службы подготовки составов, справочной службы и других, которые административно относятся каждый к своему отделу. Каждая из перечисленных категорий работников имеет уникальные атрибуты-характеристики, определяемые профессиональной направленностью. В отделах существует разбиение работников на бригады. Отделы возглавляются начальниками, которые представляют собой администрацию железнодорожной станции. В функции администрации входит планирование маршрутов, составление расписаний, формирование кадрового состава железнодорожной станции. За каждым локомотивом закрепляется локомотивная бригада. За несколькими локомотивами закрепляется бригада техников-ремонтников, выполняющая рейсовый и плановый техосмотр (по определенному графику), ремонт, техническое обслуживание. Водители локомотивов обязаны проходить каждый год медосмотр, не прошедших медосмотр необходимо перевести на другую работу. Локомотив должен своевременно осматриваться техниками-ремонтниками и про необходимости ремонтироваться. Подготовка к рейсу включает в себя техническую часть (рейсовый техосмотр, мелкий ремонт) и обслуживающую часть (уборка вагонов, запас продуктов питания и т.п.).

В расписании указывается тип поезда (скорый, пассажирский . . .), номер поезда, дни и время отправления и прибытия, маршрут (начальный и конечный пункты назначения, основные узловые станции), стоимость билета. Билеты на поезд можно приобрести заранее или забронировать в железнодорожных кассах. До отправления поезда, если есть необходимость, билет можно вернуть. Отправление поездов может быть задержано из-за опозданий поездов, погодных условий, технических неполадок.

Железнодорожные маршруты можно разделить на следующие категории: внутренние, международные, туристические, специальные маршруты. Пассажиры могут сдавать свои вещи в багажное отделение.

Виды запросов в информационной системе:

– Получить перечень и общее число всех работников железнодорожной станции, начальников отделов, работников указанного отдела, по стажу работы

на станции, половому признаку, возрасту, признаку наличия и количества детей, размеру заработной платы.

– Получить перечень и общее число работников в бригаде, по всем отделам, в указанном отделе, обслуживающих некоторый локомотив, по возрасту, суммарной (средней) зарплате в бригаде.

– Получить перечень и общее число водителей локомотивов, прошедших медосмотр либо не прошедших медосмотр в указанный год, по половому признаку, возрасту, размеру заработной платы.

– Получить перечень и общее число локомотивов, приписанных к железнодорожной станции, находящихся на ней в указанное время, по времени прибытия на станции, по количеству совершенных маршрутов.

– Получить перечень и общее число локомотивов, прошедших плановый техосмотр за определенный период времени, отправленных в ремонт в обозначенное время, ремонтированных указанное число раз, по количеству совершенных рейсов до ремонта, по возрасту локомотива.

– Получить перечень и общее число поездов на указанном маршруте, по длительности маршрута, по цене билета и по всем этим критериям сразу.

– Получить перечень и среднее количество проданных билетов за указанный интервал времени на определённые маршруты, по длительности маршрута, по цене билета.

– Получить перечень и общее число маршрутов указанной категории, следующих в определенном направлении.

– Получить перечень и общее число пассажиров на указанном рейсе, уехавших в указанный день, уехавших за границу в указанный день, по признаку сдачи вещей в багажное отделение, по половому признаку, по возрасту.

– Получить перечень и общее число невыкупленных билетов на указанном рейс, день, некоторый маршрут.

ПРИЛОЖЕНИЕ Г
(обязательное)
Главные и рабочие окна приложения

В ходе проектирования было определено, что при запуске приложения открываться главное окно с меню вкладок, при нажатии на них, будет отображена указанная таблица. Окно входа в систему представлено на рисунке Г.1.

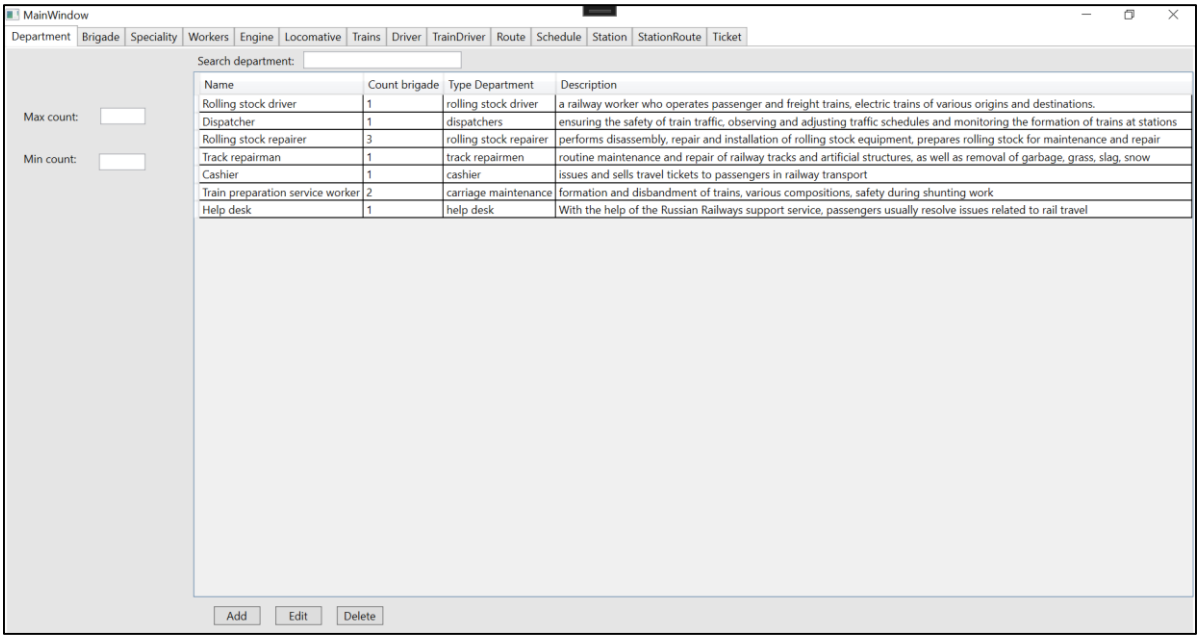

Рисунок Г.1 – Главное окно приложения

При нажатие на вкладку «Worker», перейдем к полю для работы таблицей «Worker».

Поле для работы с таблицей «Worker» представлено на рисунке Г.2.

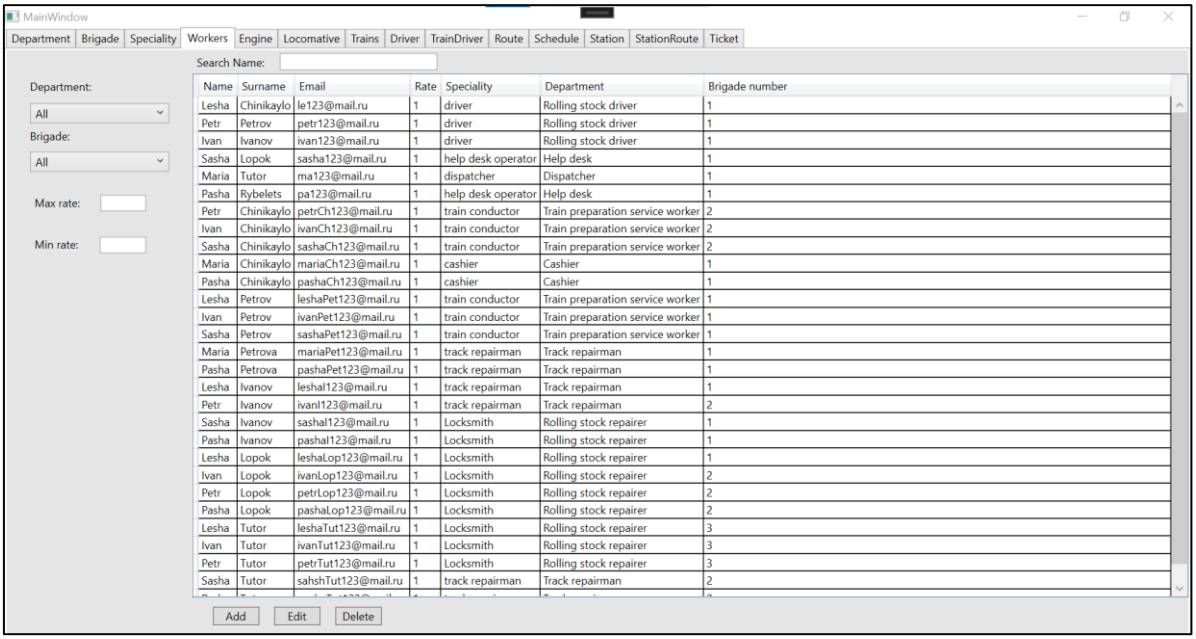

Рисунок Г.2 – переход к таблице «Worker»

После выбора определённой таблицы, приложение определяет какое окно добавления или редактирования открывать после нажатия на кнопки «Add» и «Edit».

На рисунке Г.3 представлено окно добавления новых работников, после нажатия на кнопку «Add»

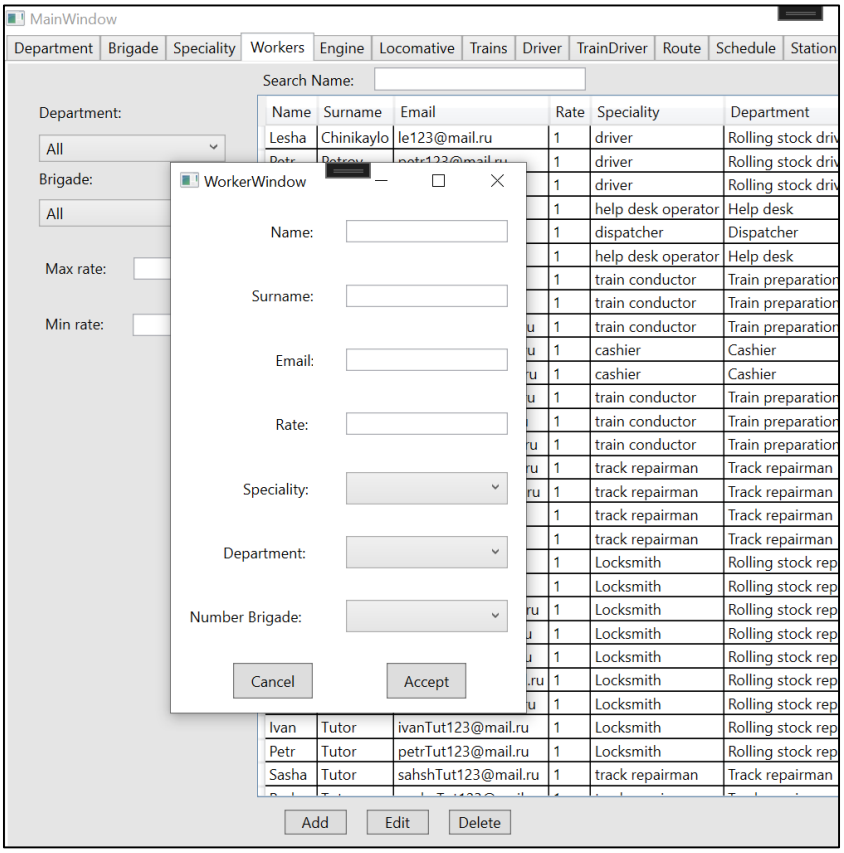

Рисунок Г.3 – Окно добавления записи

При редактировании записей мы обязаны указать какую именно запись мы хотим отредактировать, после этого нам отобразиться окно редактирования с выгруженными полями выбранной записи. Окно редактирования представлено на рисунке Г.4.

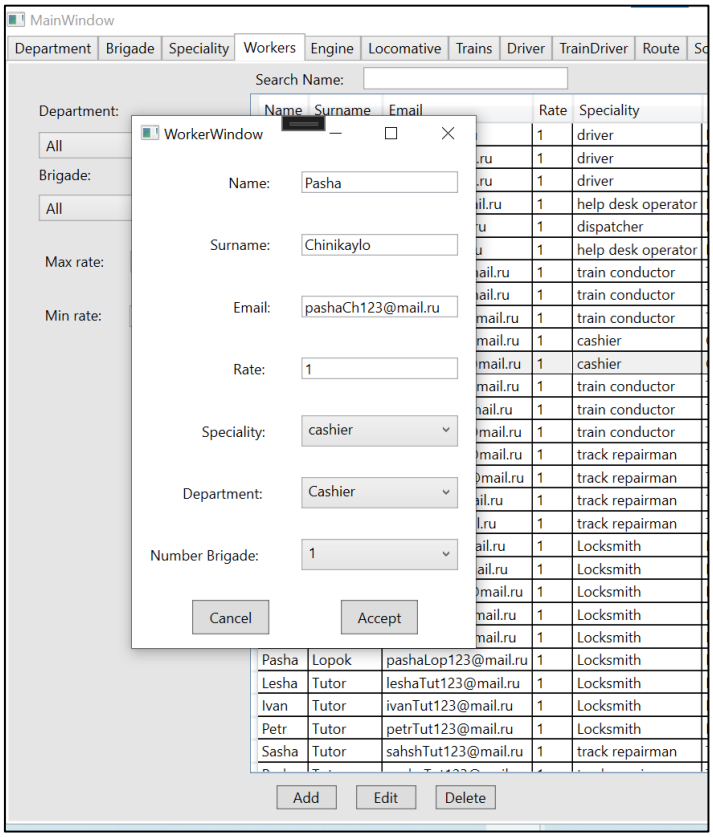

Рисунок Г.4 – Окно редактирования записи

После нажатия на кнопку «Delete», в зависимости от выбора, будет произведено удаление. Главное окно приложения после удаления записи представлено на рисунке Г.5.

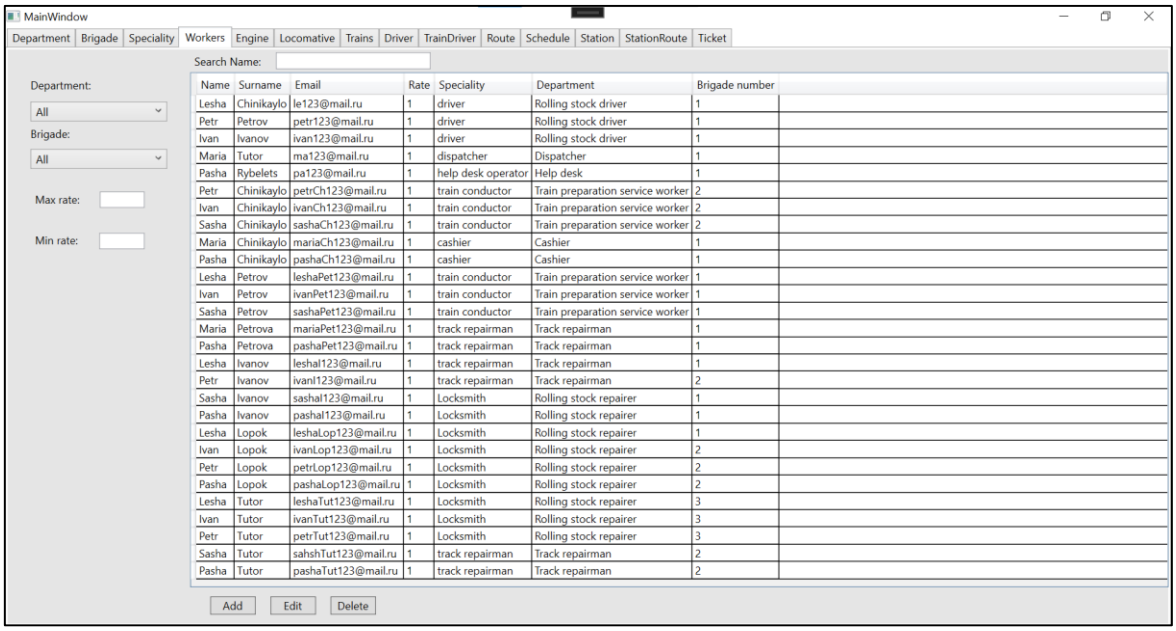

Рисунок Г.5 – Главное окно приложения после удаления записи

Любые изменения будут сохранятся в используемой базе данных.