СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Без эффективного управления современное производство не может существовать. Базы данных, применяемые на различных предприятиях, предназначены для хранения разнообразной информации. Системы управления базами данных (СУБД) обеспечивают создание, хранение, редактирование и автоматизированный поиск информации, а также составление отчетов. Наиболее известными СУБД являются Oracle, MS Access, SQL и FoxPro. При сетевой работе база данных устанавливается на сервере предприятия, а доступ к ней осуществляется с рабочих мест.

Основные функции СУБД включают управление данными во внешней памяти, управление транзакциями, журнализацию и поддержку языков БД. Единая СУБД для предприятия должна обеспечивать настройку на различные формы документов, работу с графическими и нормативно-справочными данными, а также представление знаний в различных формах.

Использование СУБД позволяет упростить управление информацией, обеспечить резервное копирование и восстановление данных, а также защитить их от несанкционированного доступа. Для различных объемов информации обычно используются Microsoft Access, MS SQL Server или Oracle. Возможно совместное использование нескольких СУБД одновременно.

Важным аспектом является также производительность СУБД. Эффективное управление базами данных позволяет быстро обрабатывать большие объемы информации и выполнять сложные запросы. Это обеспечивает более высокую производительность работы предприятия в целом, ускоряет процессы принятия решений и повышает эффективность бизнес-процессов.

В зависимости от объема и характера информации на предприятии могут использоваться различные типы СУБД: от Microsoft Access для небольших организаций до MS SQL Server или Oracle для крупных корпораций. Нередко компании сочетают в своей работе несколько СУБД, чтобы эффективно управлять различными типами данных.

Одним из основных преимуществ использования СУБД является возможность упрощения процессов управления информацией, а также обеспечение надежности и целостности данных. Благодаря СУБД можно легко восстанавливать данные в случае их потери и защищать информацию от несанкционированного доступа.

Проектирование архитектуры базы данных играет критическую роль в обеспечении ее эффективной работы. Архитектура базы данных определяет структуру данных, их организацию, способы доступа и обработки, а также обеспечивает масштабируемость и производительность системы. Хорошо спроектированная архитектура базы данных учитывает требования к данным, особенности бизнес-процессов и обеспечивает оптимальное использование ресурсов сервера.

Существует две модели данных: жесткая общепринятая, предполагающая создание и поддержание централизованной базы данных, и гибкая, которая работает с разнородными базами данных, ведущимися независимо в разных подразделениях.

Централизованные системы управления базами данных позволяют оперативно отслеживать заказы, просчитывать и закупать требуемые объемы материалов, а также решать вопросы производственного планирования и оперативного управления производством.

Использование баз данных становится неотъемлемой частью профессиональной деятельности современного человека, и эффективное применение соответствующих технологий и программных продуктов становится все более актуальным. Область применения БД и СУБД для решения различных экономических задач очень обширна, и организации осознают необходимость в интеграции различных типов информации в свои бизнес-процессы.

В рамках данной курсовой работы планируется создание информационной системы проектного менеджера с использованием СУБД PostgreSQL и среды разработки CLion 2023.1f.

1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ

1.1 Анализ предметной области и выявление необходимого набора сущностей

Выбрана тема "Информационная Система Проектный Менеджер", которая предназначена для эффективного управления проектами в организации. Система охватывает широкий спектр функций, начиная от создания и планирования проектов, их реализации и контроля за выполнением задач, и заканчивая анализом и отчетностью.

В основе информационной системы лежат несколько ключевых таблиц, включая таблицы команд, заданий, пользователей, ролей, проектов, стадий проектов, статусы проектов, комментарии. Каждая из этих таблиц содержит важную информацию, необходимую для эффективного управления проектами.

Таблица команды содержит данные о командах сотрудников, работающих над определенными проектами. В таблице заданий хранится информация о задачах, которые необходимо выполнить в рамках проекта. Типы заданий определяют различные виды задач и их характеристики.

Таблица пользователей содержит информацию о сотрудниках организации, их ролях в проектах и других атрибутах. Роли определяют функциональные обязанности сотрудников в рамках команды. В таблице проектов содержится описание проектов, их цели, сроки и бюджеты.

Стадии проекта отражают этапы развития проекта, а статус проекта позволяет отслеживать текущее состояние проекта в процессе. Статусы и стадии проекта определяются администратором. Стадии уникальные для каждого проекта, а статусы могут быть одинаковыми для нескольких проектов в один и тот же момент времени.

Система также поддерживает возможность назначения ответственных команд за выполнение задач, установки сроков и обновления статуса, а также мониторинга прогресса выполнения задач. Кроме того, система предоставляет возможность анализа данных о проектах и создания отчетов для руководства о текущем состоянии и эффективности реализации проектов.

Так же система предоставляет функционал пользователем делится файлами и комментариями. Все комментарии хранятся в базе данных, а файлы могут иметь разный тип данных для удобной интерпретации на клиентах.

Таким образом, информационная система проектного менеджера играет ключевую роль в управлении проектами, повышая эффективность работы команды, сокращая временные и финансовые затраты и улучшая результаты проектов.

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

Определим набор атрибутов для каждой сущности с целью построения концептуальной модели данных (КМД). КМД представляет собой общую информационную модель предметной области, которая включает в себя вопросы о классификации, структуризации и семантической целостности данных.

Атрибут – это характеристика сущности, обладающая уникальным наименованием в пределах конкретного типа сущности. Наименования атрибутов не должны совпадать с системными ключевыми словами, такими как «select» или «from», но могут быть одинаковыми для различных типов сущностей. Атрибуты определяют, 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
Projects	Id	PK	Идентификационный номер таблицы Projects
	ProjectStatusId	FK	Идентификационный номер таблицы ProjectsStatuses
	ProjectStageId	FK	Идентификационный номер таблицы ProjectsStages
ProjectDetails	Id	PK	Идентификационный номер таблицы ProjectDetails
	Title		Название проекта
	Description		Описание проекта
	StartDate		Дата начала работы над проектом
	ProjectId		Идентификационный номер таблицы Projects
ProjectStatus	Id	PK	Идентификационный номер таблицы ProjectStatus
	Title		Название статуса проекта
	Description		Описание статуса проекта
ProjectStage	Id	PK	Идентификационный номер таблицы ProjectStage
	Title		Название стадии проекта
	Description		Описание стадии проекта
	StartDate		Дата начала стадии
	EndDate		Дата окончания стадии

Продолжение таблицы 1.1

Таблица	Поле	Ключ	Описание
ProjectReports	id	PK	Идентификационный номер таблицы ProjectReports
	ProjectId	FK	Идентификационный номер таблицы Projects
	ReportId	FK	Идентификационный номер таблицы Reports
Report	id	PK	Идентификационный номер таблицы Report
	Value		Целочисленное значения отчета
	AdditionalInfo		Дополнительная информация
TaskTags	id	PK	Идентификационный номер таблицы TaskTags
	Tag		Название метки задания
	TaskId	FK	Идентификационный номер таблицы Task
TaskStatuses	id	PK	Идентификационный номер таблицы TaskStatus
	Title		Название состояния задания
	Staging		Значение тестируемости проекта в стадии
	Description		Описание стадии
Task	id	PK	Идентификационный номер таблицы Task
	Title		Заголовок задачи
	Description		Описание задачи
	TeamId	FK	Идентификационный номер таблицы Teams
	TaskStatusId	FK	Идентификационный номер таблицы TaskStatuses
Team	Id	PK	Идентификационный номер таблицы Team
	ProjectId	FK	Идентификационный номер таблицы Projects
	Description		Описание команды

Продолжение таблицы 1.1

Таблица	Поле	Ключ	Описание
Role	Id	PK	Идентификационный номер таблицы Role
	Title		Заголовок роли
	Description		Описание роли
	Priority		Приоритет роли
User	id	PK	Идентификационный номер таблицы User
	Name		ФИО пользователя
	Email		Почта пользователя
	RoleId	FK	Идентификационный номер таблицы Role
Comment	Id	PK	Идентификационный номер таблицы Comment
	AuthorId	FK	Идентификационный номер таблицы User
	Content		Тело комментария
	Posted		Дата публикации комментария
FileType	Id	PK	Идентификационный номер таблицы FileType
	Description		Описание типа файла
	Extension		Расширение файла
Attachment	Id	PK	Идентификационный номер таблицы Attachment
	AuthorId	FK	Идентификационный номер таблицы User
	FileTypeId	FK	Идентификационный номер таблицы FileType
	FileContent		Содержимое файла
	FileName		Название файла
	FileSize		Размер файла

1.3 Определение связей между объектами

В информационной системе станции объекты связаны между собой через определенные отношения, которые устанавливаются с помощью внешних ключей. Внешний ключ представляет собой ссылку на первичный ключ или уникальный атрибут другой таблицы, обеспечивая тем самым связь между сущностями. Эти связи могут быть однозначными или многозначными, обязательными или необязательными.

Для построения информационной системы станции необходимо анализировать все сущности и определять, как они взаимодействуют между собой. Например, можно рассмотреть отношения между пользователями и их мобильными устройствами, где каждый пользователь может иметь одно или несколько устройств, а каждое устройство принадлежит только одному пользователю. Также важно учитывать отношения, где каждый объект может быть связан только с одним другим объектом (один к одному) или может иметь несколько связанных объектов (один ко многим).

Кроме того, важно учитывать также отношения "многие ко многим", которые требуют использования посредника между двумя сущностями для их связи. Например, если мы рассматриваем связь между курсами и студентами, где один курс может иметь несколько студентов, а один студент может записаться на несколько курсов, нам понадобится таблица, которая будет отображать это соответствие между курсами и студентами.

Каждая связь имеет два конца и одно или два наименования. Наименование обычно выражается в неопределенной глагольной форме: «иметь», «принадлежать» и тому подобное. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся из-за их очевидности.

Связи делятся на:

– Многие ко многим: Для установки связи многие ко многим между двумя сущностями требуется использовать посредника, который будет хранить информацию о связи между этими сущностями. Этот посредник должен содержать два внешних ключа: один связан с первой сущностью, а другой – с второй.

– Один ко многим: может быть с обязательной или необязательной. Это означает, что одна сущность может быть связана с несколькими другими сущностями, но каждая из этих других сущностей может быть связана только с одной сущностью первого типа. Например, у пользователя может быть несколько мобильных устройств, каждое из которых имеет поле для владельца, но каждое устройство может быть связано только с одним пользователем.

– Один к одному: с обязательной или необязательной связью каждая сущность может быть связана только с одной другой сущностью, при этом значением в столбце, относящемуся ко второй сущности, является уникальное значение. Например, в системе голосования каждый избиратель может отдать

голос только один раз, поскольку система идентифицирует их по уникальному идентификатору, такому как удостоверение личности. Для обеспечения уникальности значения этого идентификатора можно использовать ограничение «primary key», которое не допускает наличие значений «null» и гарантирует уникальность.

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 информационной модели предметной области представляет собой частично формализованное описание объектов этой области с использованием определенной семантической модели, такой как ER-модель. В рамках этого процесса создается концептуальная модель, которая не привязана к какой-либо конкретной системе управления базами данных или модели данных. Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются взаимозаменяемыми и описывают образ предметной области без привязки к техническим деталям.

Она включает в себя описание информационных объектов и связей между ними, а также определение ограничений целостности, задающих требования к допустимым данным и их связям. Обычно концептуальная модель представляется с помощью графических нотаций, таких как ER-диаграммы, и она является основой для проектирования базы данных в будущем.

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 создания схемы реляционной базы данных необходимо определить набор таблиц, которые будут составлять базу данных. Этот набор таблиц должен содержать всю необходимую информацию, которая будет храниться в базе данных.

Согласно полученной концептуальной модели, необходимо определить набор отношений для базы данных. Если конец связи отмечен ключом, это указывает на то, что сущность с этого конца является основной относительно сущности на другом конце связи. Все отношения в представленном наборе приведены к виду "один ко многим", что означает, что конец связи с ключом указывает на то, что эта сущность используется несколько раз в зависимой сущности. На рисунке 2.1 представлены эти отношения.

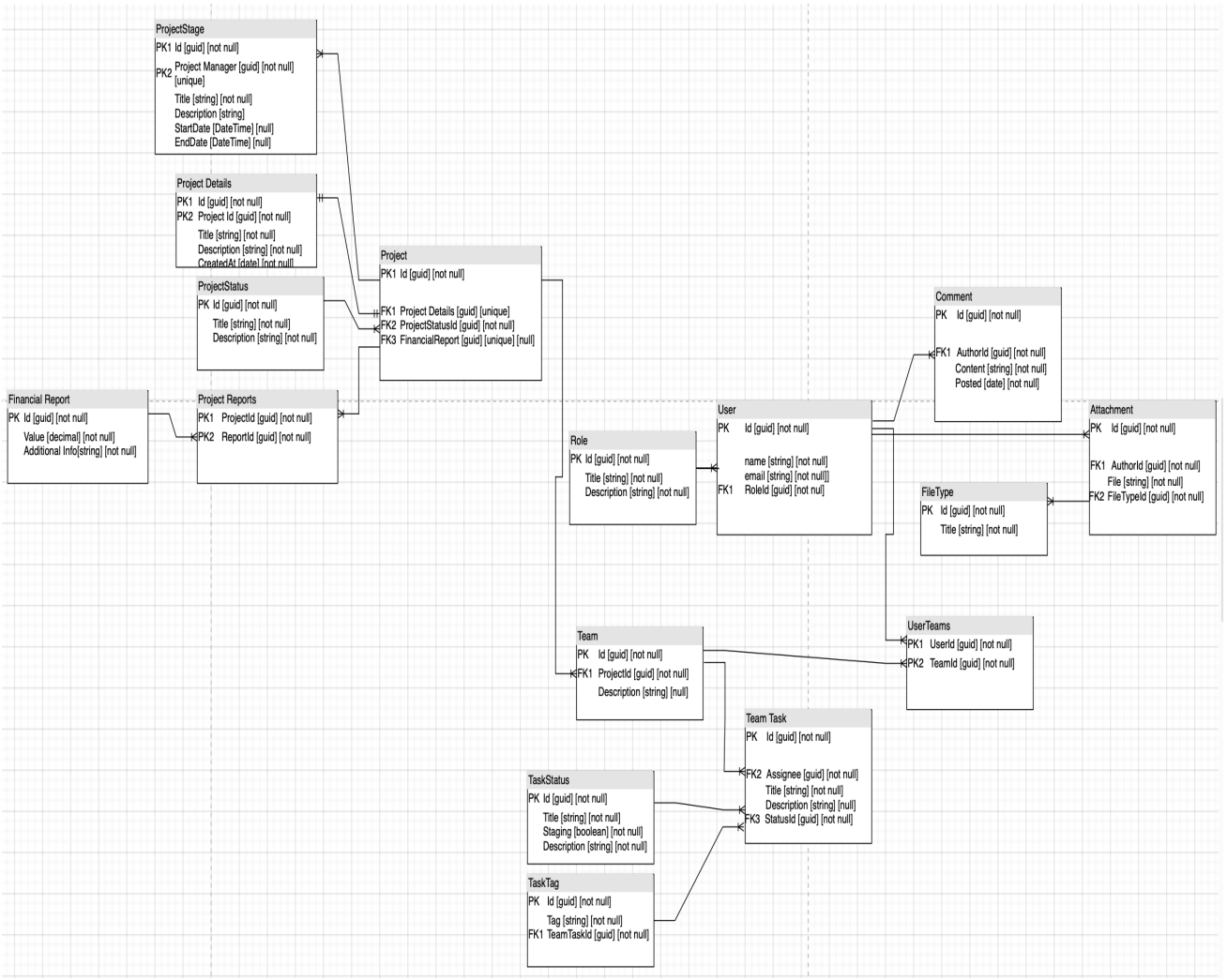


Рисунок 2.1 – Набор необходимых отношений базы данных

2.2 Задание первичных и внешних ключей определенных отношений

В каждой таблице базы данных присутствует первичный ключ, который является полем или комбинацией полей, обеспечивающих уникальную идентификацию записи. Первичный ключ должен быть минимальным в том смысле, что он не содержит лишних полей, удаление которых не повлияет на его уникальность [3].

Поле первичного ключа в базе данных функционирует как уникальный идентификатор записи. Кроме того, оно используется для установления связей между таблицами. В связанных таблицах поле первичного ключа из родительской таблицы становится внешним ключом в дочерней таблице. Внешний ключ в дочерней таблице ссылается на данные в родительской таблице, обеспечивая таким образом связь между записями двух таблиц.

Первичный ключ в реляционной модели данных – это ключевой атрибут отношения, который выделяется в качестве основного идентификатора (или основного ключа).

Если в базе данных присутствует только один потенциальный ключ, то этот ключ автоматически становится первичным. В случае, если имеется несколько потенциальных ключей, из них выбирается один в качестве первичного, а остальные называются «альтернативными».

При рассмотрении потенциальных ключей отношения с точки зрения теории, они все считаются одинаково значимыми, обладая свойствами уникальности и минимальности. Однако, когда дело доходит до выбора первичного ключа, практические аспекты играют решающую роль. Например, при создании внешних ключей в других отношениях или формировании кластерного индекса. В результате первичным ключом часто становится тот, который обладает наименьшим объемом данных (физическим хранением) или включает в себя наименьшее количество атрибутов.

В SQL, внешний ключ - это механизм, который определяет связь между двумя таблицами базы данных. Он представляет собой набор атрибутов в таблице, значения которых связаны со значениями ключа в другой таблице. Этот ключ обеспечивает целостность данных, гарантируя, что значение внешнего ключа существует в связанной таблице. Таблица, содержащая основной ключ, называется родительской таблицей, а таблица, содержащая внешний ключ, - дочерней таблицей. Используя внешние ключи, можно создавать связи между различными сущностями в базе данных, обеспечивая эффективное управление данными.

Сохранение ссылочной целостности, или поддержка внешних ключей, является одним из ключевых аспектов реляционных СУБД. Это означает, что СУБД автоматически контролирует соответствие значений внешнего ключа значениям первичного ключа в связанных таблицах. Такой подход обеспечивает целостность данных, предотвращая появление ошибочных или неконсистентных связей между данными. Благодаря автоматическому контролю ссылочной

целостности, администратор баз данных может быть уверен в надежности и консистентности данных, что является важным аспектом в проектировании и поддержке баз данных.

Первичные ключи, обозначаемые постфиксом PK, представляют собой уникальные поля в таблице, служащие для идентификации каждой записи. Они играют важную роль в обеспечении уникальности данных и упорядочивании таблицы. Вторичные ключи, обозначаемые постфиксом FK, используются для создания связей между таблицами. Эти ключи соотносятся с первичными ключами других таблиц и помогают устанавливать ссылки между различными записями. Важно отметить, что в данном исследовании первичные ключи будут единственным полем в каждой таблице, что подчеркивает их уникальность и значимость для структуры базы данных.

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

Процесс нормализации данных направлен на оптимизацию структуры базы данных путем устранения избыточности и предотвращения аномалий в хранении и обновлении данных. Целью этого процесса является достижение третьей нормальной формы (3NF) [5], что означает, что каждый не ключевой атрибут в таблице должен зависеть только от первичного ключа и не зависеть от других не ключевых атрибутов. Это помогает улучшить эффективность запросов к базе данных, обеспечить ее консистентность и предотвратить аномальные ситуации при вставке, обновлении или удалении данных. Нормализация является важным этапом проектирования баз данных, который способствует их более эффективному использованию и обеспечивает лучшую структуру данных для приложений.

Избыточные данные не только занимают драгоценное место на диске, но также могут создавать проблемы при обслуживании базы данных. Когда данные дублируются в нескольких местах, это увеличивает вероятность ошибок и несогласованности. Если требуется изменить эти данные, это придется сделать в каждом месте, где они дублируются, чтобы гарантировать их согласованность. Это делает процесс обслуживания более сложным и увеличивает риск ошибок. Нормализация данных, включая приведение к третьей нормальной форме, помогает устранить избыточность данных и минимизировать такие проблемы, обеспечивая более эффективное использование ресурсов и более легкое обслуживание базы данных.

Первая нормальная форма (1NF) является основой нормализации данных и требует, чтобы каждое поле в таблице базы данных было атомарным, то есть неделимым на более мелкие части, и не содержало повторяющихся групп значений. Это означает, что каждое поле должно содержать только одно значение для каждой записи, и не должно быть структурированных или множественных данных. Путем приведения данных к этой форме обеспечивается единообразие структуры данных в базе данных, что упрощает их обработку и улучшает надежность и эффективность хранения.

Неделимость поля означает, что значение этого поля не может быть разделено на более мелкие составляющие. Например, если в поле "Подразделение" содержится как название факультета, так и название кафедры, это нарушает требование неделимости поля. В таком случае рекомендуется разделить это поле на два отдельных поля: одно для названия факультета и другое для названия кафедры. Это поможет соблюсти первую нормальную форму данных (1NF) и обеспечит более гибкую и эффективную структуру для хранения и обработки информации в базе данных.

Когда в таблице существуют поля, содержащие однотипные значения, это означает, что мы имеем дело с повторяющимися данными. Например, представим, что мы хотим составить статистику по результатам экзаменов по различным предметам. В такой ситуации, создание отдельного поля для каждого предмета, где бы хранились оценки студентов, приводит к повторяющимся группам данных.

Вторая нормальная форма (2NF) требует, чтобы все поля в таблице зависели от ее первичного ключа, обеспечивая тем самым каждой записи уникальность и исключая избыточность данных. Если какие-то поля зависят только от части первичного ключа, то они должны быть выделены в отдельные таблицы. Такой подход помогает избежать избыточности информации и создает более логичную и эффективную структуру базы данных.

Третья нормальная форма (3NF) устанавливает требование независимости значений любого поля таблицы, не являющегося частью первичного ключа, от значений других полей, также не включенных в первичный ключ. Это означает, что каждое не ключевое поле должно зависеть только от первичного ключа и не иметь функциональных зависимостей от других не ключевых полей. В результате достигается более высокий уровень нормализации данных, что способствует уменьшению избыточности и повышению структурированности базы данных.

Исходя из предположения, что все атрибуты наших отношений атомарные, каждое отношение имеет первичный ключ, отсутствуют функциональные зависимости между не ключевыми атрибутами, а также отсутствуют зависимости не ключевых атрибутов от части составного ключа, можно сделать вывод о том, что отношения базы данных находятся в третьей нормальной форме (3NF). Это свидетельствует о высокой структурированности данных и отсутствии избыточности, что обеспечивает более эффективное хранение и обработку информации в базе данных.

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

Ограничения целостности - это особые механизмы в базах данных, разработанные с целью предотвратить введение в базу данных некорректных данных. Их главное назначение заключается в том, чтобы гарантировать, что данные, сохраняемые в базе данных, соответствуют определенным правилам и ограничениям, установленным администратором базы данных. Эти ограничения могут включать в себя правила для проверки корректности значений полей, требования к уникальности данных, а также условия для обеспечения связности между данными. Путем использования ограничений целостности обеспечивается надежность и консистентность данных в базе, что является ключевым аспектом в управлении информацией.

Ограничение целостности отношений состоит в том, что в каждом отношении базы данных не должно быть записей с одинаковым значением первичного ключа. Суть этого требования заключается в том, чтобы каждая запись в отношении была уникальной и не дублировалась. Это обеспечивает консистентность данных и предотвращает появление дублирующихся или некорректных записей в базе данных. Автоматическое выполнение этого требования обеспечивается при соблюдении базовых свойств отношений в системе, что является фундаментальным принципом реляционной модели данных.

Ограничение целостности предполагает, что в каждом отношении базы данных не должно быть записей с одинаковым значением первичного ключа. Другими словами, каждая запись в отношении должна быть уникальной и отличаться от всех остальных записей в этом отношении. Первичный ключ играет ключевую роль в этом процессе, поскольку обеспечивает однозначную идентификацию каждой сущности в таблице. Например, на стороне клиента первичный ключ также используется при выполнении операций удаления, добавления или обновления записи. Это гарантирует целостность данных и предотвращает появление дублирующийся или некорректных записей в базе данных.

Условия целостности представляют собой набор правил и ограничений, которые обеспечивают соблюдение допустимых отношений между таблицами в базе данных и предотвращают случайные изменения или удаления связанных данных. Эти правила определяются и реализуются в базе данных с целью обеспечить целостность и надежность хранения данных. К примеру, условия целостности могут включать в себя требование существования связи между записями в различных таблицах (внешние ключи), а также ограничения на обновление или удаление данных, которые нарушают целостность связанных

записей. Все это направлено на поддержание структурной целостности базы данных и обеспечение надежности её работы.

Для обеспечения автоматического обновления связанных полей при изменении или удалении записей в главной таблице необходимо использовать механизмы обеспечения целостности данных. Один из таких механизмов - каскадное обновление связанных полей и каскадное удаление связанных записей. При использовании каскадного обновления, если значение первичного ключа в главной таблице изменяется, то все связанные поля в дочерних таблицах автоматически обновляются соответствующим образом. Аналогично, при каскадном удалении, если запись в главной таблице удаляется, все связанные записи в дочерних таблицах также автоматически удаляются. Эти механизмы обеспечивают целостность данных в базе и предотвращают появление битых ссылок или несогласованных записей.

Для обеспечения соблюдения ограничений целостности для внешних ключей отношений и для отношений в целом крайне важно, чтобы типы вводимых данных соответствовали типам столбцов в таблицах. Это означает, что данные, которые вводятся в таблицы, должны быть совместимы с типами данных, определенными для соответствующих столбцов. Кроме того, все обязательные поля в таблицах, то есть те поля, которые не могут содержать значения NULL, должны быть заполнены для каждой записи. Это обеспечивает консистентность данных и предотвращает появление неполных или некорректных записей в базе данных. Все эти меры направлены на поддержание целостности данных и обеспечение надежности работы базы данных.

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 процессе нормализации и определения первичных ключей, внешних ключей и связей между сущностями была разработана схема реляционной базы данных, представленная на рисунке Б.1 в приложении Б. На данной схеме изображены все отношения базы данных, а также связи между внешними и первичными ключами.

3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

Использование системы управления базами данных PostgreSQL для реализации спроектированной базы данных станции обусловлено несколькими весомыми причинами. Во-первых, PostgreSQL широко распространен и имеет долгую историю успешного применения в различных отраслях и организациях, что гарантирует его надежность и стабильность. Кроме того, доступность бесплатных версий PostgreSQL позволяет использовать его как для небольших проектов, так и для крупных предприятий без затрат на лицензирование.

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

Таблица 3.1 – Характеристики атрибутов таблицы Projects

Поле	Тип данных	Описание
Id	TEXT	Идентификационный номер таблицы Projects
ProjectStatusId	TEXT	Идентификационный номер таблицы ProjectsStatuses
ProjectStageId	TEXT	Идентификационный номер таблицы ProjectsStages

Таблица 3.2 – Характеристики атрибутов таблицы ProjectDetails

Поле	Тип данных	Описание
Id	TEXT	Идентификационный номер таблицы ProjectDetails
Title	TEXT	Название проекта
Description	TEXT	Описание проекта
StartDate	DATE	Дата начала работы над проектом
EndDate	DATE	Дата окончания проекта

Таблица 3.3 – Характеристики атрибутов таблицы ProjectStage

Поле	Тип данных	Описание
Id	TEXT	Идентификационный номер таблицы ProjectStatus
Title	TEXT	Название статуса проекта
Description	TEXT	Описание статуса проекта

Таблица 3.4 – Характеристики атрибутов таблицы ProjectReports

Поле	Тип данных	Описание
id	TEXT	Идентификационный номер таблицы ProjectReports
ProjectId	TEXT	Идентификационный номер таблицы Projects
ReportId	TEXT	Идентификационный номер таблицы Reports

Таблица 3.5 – Характеристики атрибутов таблицы Report

Поле	Тип данных	Описание
Id	TEXT	Идентификационный номер таблицы Report
Value	MONEY	Целочисленное значения отчета
AdditionalInfo	TEXT	Дополнительная информация

Таблица 3.6 – Характеристики атрибутов таблицы TaskTags

Поле	Тип данных	Описание
id	TEXT	Идентификационный номер таблицы TaskTags
Tag	NVARCHAR(25)	Название метки задания
TaskId	TEXT	Идентификационный номер таблицы Task

Таблица 3.7 – Характеристики атрибутов таблицы TaskStatus

Поле	Тип данных	Описание
id	TEXT	Идентификационный номер таблицы TaskStatus
Title	TEXT	Название состояния задания
Staging	BOOLEAN	Значение тестируемости проекта в стадии
Description	TEXT	Описание стадии

Таблица 3.8 – Характеристики атрибутов таблицы Task

Поле	Тип данных	Описание
id	TEXT	Идентификационный номер таблицы Task
Title	TEXT	Заголовок задачи
Description	TEXT	Описание задачи
TeamId	TEXT	Идентификационный номер таблицы Teams
TaskStatusId	TEXT	Идентификационный номер таблицы TaskStatuses

Таблица 3.9 – Характеристики атрибутов таблицы Team

Поле	Тип данных	Описание
Id	TEXT	Идентификационный номер таблицы Team
ProjectId	TEXT	Идентификационный номер таблицы Projects
Description	TEXT	Описание команды

Таблица 3.10 – Характеристики атрибутов таблицы Role

Поле	Тип данных	Описание
Id	TEXT	Идентификационный номер таблицы Role
Title	NVARCHAR(25)	Заголовок роли
Description	TEXT	Описание роли
Priority	INTEGER	Приоритет роли

Таблица 3.11 – Характеристики атрибутов таблицы User

Поле	Тип данных	Описание
id	TEXT	Идентификационный номер таблицы User
Name	TEXT	ФИО пользователя
Email	TEXT	Почта пользователя
RoleId	TEXT	Идентификационный номер таблицы Role

Таблица 3.12 – Характеристики атрибутов таблицы Comment

Поле	Тип данных	Описание
Id	TEXT	Идентификационный номер таблицы Comment
AuthorId	TEXT	Идентификационный номер таблицы User
Content	TEXT	Тело комментария
Posted	DATE	Дата публикации комментария

Таблица 3.13 – Характеристики атрибутов таблицы FileType

Поле	Тип данных	Описание
Id	TEXT	Идентификационный номер таблицы FileType
Description	TEXT	Описание типа файла
Extension	NVARCHAR(10)	Расширение файла

Таблица 3.14 – Характеристики атрибутов таблицы Attachment

Поле	Тип данных	Описание
Id	TEXT	Идентификационный номер таблицы Attachment
AuthorId	TEXT	Идентификационный номер таблицы User
FileTypeId	TEXT	Идентификационный номер таблицы FileType
FileContent	TEXT	Содержимое файла
FileName	NVARCHAR(25)	Название файла
FileSize	INTEGER	Размер файла

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 – Поиск информации о проекте по его названию

```
SELECT * FROM project_details WHERE title LIKE '%%';
```

Листинг 4.2 – Получить список и общее количество пользователей в системе, в разрезе их ролей, сгруппированных по ролям и по электронной почте.

```
SELECT users.name, users.email, roles.name AS role_name
FROM users
INNER JOIN roles ON users.fk_role_id = roles.id;
```

Листинг 4.3 – Получить список всех файлов и их типов для определенного пользователя. Этот запрос позволяет получить подробную информацию о файлах, загруженных конкретным пользователем. Запрос осуществляет соединение таблицы вложенных файлов с таблицей типов файлов для получения информации о типах файлов, а также соединяет таблицу пользователей для определения файлов, загруженных указанным пользователем.

```
SELECT attachments.file_name, file_types.extension AS file_type
FROM attachments
INNER JOIN file_types ON attachments.fk_file_type_id = file_types.id
INNER JOIN users ON users.id = '' WHERE users.id = attachments.fk_author_id;
```

Листинг 4.4 – Получить перечень всех этапов проекта и соответствующие задачи. Этот запрос позволяет получить список всех этапов проекта вместе с названиями соответствующих задач. Он осуществляет соединение таблиц этапов проекта и задач, чтобы ассоциировать каждую задачу с соответствующим этапом проекта. Запрос также включает соединение с таблицей команд для ограничения результатов только задачами из определенного проекта.

```
SELECT project_stages.title AS stage_title, tasks.title AS task_title
FROM project_stages, tasks
INNER JOIN teams ON teams.fk_project_id = '';
```

Листинг 4.5 – Получить список названий задач, их статусов и тегов для задач, принадлежащих определенной команде проекта. Этот запрос позволяет получить информацию о задачах, включая их названия, статусы и теги, для задач, принадлежащих определенной команде в рамках проекта. Запрос осуществляет соединение таблицы задач с таблицами статусов задач и тегов задач для ассоциации соответствующей информации с каждой задачей. Он также

использует подзапрос для выбора идентификаторов команды из таблицы команд, ограничивая результаты только задачами, принадлежащими указанной команде проекта.

```
SELECT tasks.title AS task_title, task_statuses.title AS status_title, task_tags.tag
FROM tasks
INNER JOIN task_statuses ON tasks.fk_task_status_id = task_statuses.id
LEFT JOIN task_tags ON tasks.id = task_tags.fk_task_id
WHERE tasks.fk_team_id IN (SELECT id FROM teams WHERE fk_project_id = '');
```

Листинг 4.6 – Получить количество комментариев, оставленных определенным пользователем. Запрос выполняет фильтрацию комментариев по идентификатору автора и затем группирует их по этому идентификатору. В результате получается общее количество комментариев, оставленных указанным пользователем.

```
SELECT comments.fk_author_id as user_id, COUNT(comments.id) AS comment_count
FROM comments
WHERE comments.fk_author_id = ''
GROUP BY comments.fk_author_id;
```

Листинг 4.7 – Получить количество команд для определенного проекта. Этот запрос позволяет узнать, сколько команд связано с определенным проектом. Запрос фильтрует команды по их принадлежности к указанному проекту и затем группирует результаты по идентификатору проекта. В итоге мы получаем количество команд, связанных с этим проектом.

```
SELECT teams.fk_project_id as project_id, COUNT(teams.id) AS teams_count
FROM teams
WHERE teams.fk_project_id = ''
GROUP BY teams.fk_project_id;
```

5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

Для создания приложения была выбрана среда разработки CLion, а в качестве языка программирования был выбран C++. Выбор языка и среды разработки обусловлен следующими причинами:

– Язык C++ известен своей высокой производительностью и эффективностью.

– Он обеспечивает более низкий уровень абстракции, что особенно полезно для разработки системного и высокопроизводительного программного обеспечения.

– C++ предлагает широкий набор возможностей объектно-ориентированного программирования, что обеспечивает гибкость и модульность кода.

– CLion, в свою очередь, предоставляет удобное и мощное средство разработки с поддержкой множества функций, таких как автодополнение кода, отладка и интеграция с системами контроля версий.

Для интеграции базы данных с интерфейсом приложения использовался pqxx.

Pqxx – это библиотека для работы с базами данных с использованием языка структурированных запросов (SQL). pqxx также предоставляет интерфейс для выполнения операций SQL в приложениях, упрощая взаимодействие с базой данных [6].

В общем, технология pqxx предоставляет различные способы для управления базой данных или несколькими базами данных. Некоторые SQL-инструменты могут обладать деревьями объектов или другими визуальными интерфейсами. Они также могут включать различные изменения в файлах, связанные с элементами SQL. Эти ресурсы облегчают менеджерам баз данных работу с информацией, хранящейся в определенном наборе таблиц. Pqxx основана на Open Database Connectivity.

Open Database Connectivity – это стандартный интерфейс доступа к базам данных, разработанный для обеспечения универсального доступа к данным из различных источников данных. Эта технология позволяет приложениям взаимодействовать с базами данных, независимо от их типа и поставщика.

ODBC обеспечивает универсальный способ доступа к данным, позволяя приложениям работать с различными СУБД (системами управления базами данных), такими как MySQL, PostgreSQL, Microsoft SQL Server и другими.

Так же ODBC предоставляет интерфейс, который независим от операционной системы, что позволяет разработчикам создавать приложения, работающие на различных платформах.

ODBC поддерживает работу в многопоточных приложениях, обеспечивая безопасный доступ к базам данных из нескольких потоков выполнения.

ODBC обеспечивает совместимость с языком SQL (Structured Query Language), что позволяет выполнять запросы к данным на уровне языка запросов.

PostgreSQL – это мощная и расширяемая объектно-реляционная система управления базами данных (СУБД), которая широко используется в различных приложениях по всему миру. PostgreSQL является проектом с открытым исходным кодом, что позволяет пользователям свободно использовать, изменять и распространять его без ограничений. Так же она предлагает широкий набор возможностей, включая поддержку сложных запросов, транзакций, триггеров, хранимых процедур и многих других функций, что делает его подходящим для широкого спектра приложений. Помимо всего прочего, PostgreSQL обеспечивает высокую степень масштабируемости и надежности, позволяя обрабатывать большие объемы данных и обеспечивать доступность приложения даже при больших нагрузках. PostgreSQL поддерживает расширения, что позволяет разработчикам создавать дополнительные модули и функции для расширения возможностей СУБД в соответствии с конкретными требованиями приложения.

В современном мире PostgreSQL получило широкое распространение, особенно в области высоконагруженных систем. Такие компании как Яндекс, Netflix, GitHub используют PostgreSQL в своих системах. Благодаря этому, в исходный код PostgreSQL вносят изменения высоко квалифицированные программисты со всего мира, что только увеличивает надежность и производительность данной СУБД.

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 проектов, задач, отчетов и так далее.

При запуске программы пользователь попадает на главную страницу программы, на которой расположены все пункты управления проектами.

Все основные формы и виды выполнены в отдельных окнах 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 Raylib, которая 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.

Для открытия формы в режиме графического дизайнера необходимо в структуре проекта нажать на файл формы, 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: Меню, Таблицы, Запрос на получение перечня, Справка. Данные пункты выполнены в виде выпадающего меню. Подпунктами меню «Таблицы» являются названия таблиц используемой базы данных [4].

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 DBMethods и связываются с базой данных при помощи класса SqlConnection. Подключение к базе данных начинается с формирования строки подключения. Код класса DBMethods представлен в листинге 6.1.

Листинг 6.1 – Методы для работы с базой данных

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.IO;
using System.Reflection;
using System.Windows.Forms;
namespace TrainsApp
{
class DBMethods
{
SqlConnection connection = new SqlConnection(@"Data Source=DESKTOP-
JPAQV1A;Initial Catalog=TrainS;Integrated Security=True");
public void OpenConnection()
{
if (connection.State == System.Data.ConnectionState.Closed)
{
connection.Open();
}
}
public void CloseConnectiom()
{
if (connection.State == System.Data.ConnectionState.Open)
{
connection.Close();
}
}
public SqlConnection GetConnection()
{
return connection;
}
public void DefaultUpdateGrid(string query, DataGridView show)
{
SqlDataAdapter adapter = new SqlDataAdapter();
DataTable dt = new DataTable();
try
{
SqlCommand Comand = new SqlCommand(query, GetConnection());
adapter.SelectCommand = Comand;
adapter.Fill(dt);
show.DataSource = dt;
```

```
show.Columns[0].Visible = false;
}
catch
{
MessageBox.Show("Что-то пошло не так, повторите попытку ил обратитесь
в      службу      поддержки",      "",      MessageBoxButtons.OK,
MessageBoxIcon.Error);
}
}
public void CUOperations(string query)
{
try
{
SqlCommand Comand = new SqlCommand(query, GetConnection());
OpenConnection();
Comand.ExecuteNonQuery();
CloseConnectiom();
}
catch
{
MessageBox.Show("Не удалось внести данные, проверьте правильность и
повторите попытку", "", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
public void DeleteOperation(string tableName, int elementId)
{
try
{
SqlCommand Comand = new SqlCommand($"DELETE FROM {tableName} WHERE
_id = {elementId}", GetConnection());
OpenConnection();
Comand.ExecuteNonQuery();
CloseConnectiom();
}
catch
{
MessageBox.Show("Не удалось внести данные, проверьте правильность и
повторите попытку", "", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
}
}
}
```

ЗАКЛЮЧЕНИЕ

В процессе выполнения данной курсовой работы были закреплены навыки проектировании баз данных и реализации их в MS SQL Server 2018.

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 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.

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. Программное средство реализовано с помощью языка программирования C#.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Информатика [Электронный ресурс]. Режим доступа: http://flash-library.narod.ru/Ch-Informatics/lektion/lektion7.html. Дата доступа 21.11.2022.

2 СУБД [Электронный ресурс]. Режим доступа: https://clck.ru/NnKES. Дата доступа: 22.11.2022.

3 Понятие первичного и вторичного ключа [Электронный ресурс]. Режим доступа: http://informatic.ugatu.ac.ru/lib/office/Proekt.htm. Дата доступа 22.11.2022.

4 Пример оформления курсовой работы [Электронный ресурс]. Режим доступа: https://studbooks.net/2156021/informatika/zaklyuchenie. Дата доступа 22.11.2022.

5 Нормализация [Электронный ресурс]. Режим доступа: https://metanit.com/sql/tutorial/2.1.php/. Дата доступа 19.11.2022.

6 Изучаем C#, 3-е издание. Эндрю Стиллмен, Дженнифер Грин. [Электронный ресурс]. Режим доступа: https://codernet.ru/books/c_sharp/endryu_stillmen_-_izuchaem_c_3-e_izdanie/. Дата доступа 20.11.2022.

ПРИЛОЖЕНИЕ А
(обязательное)
Концептуальная схема БД

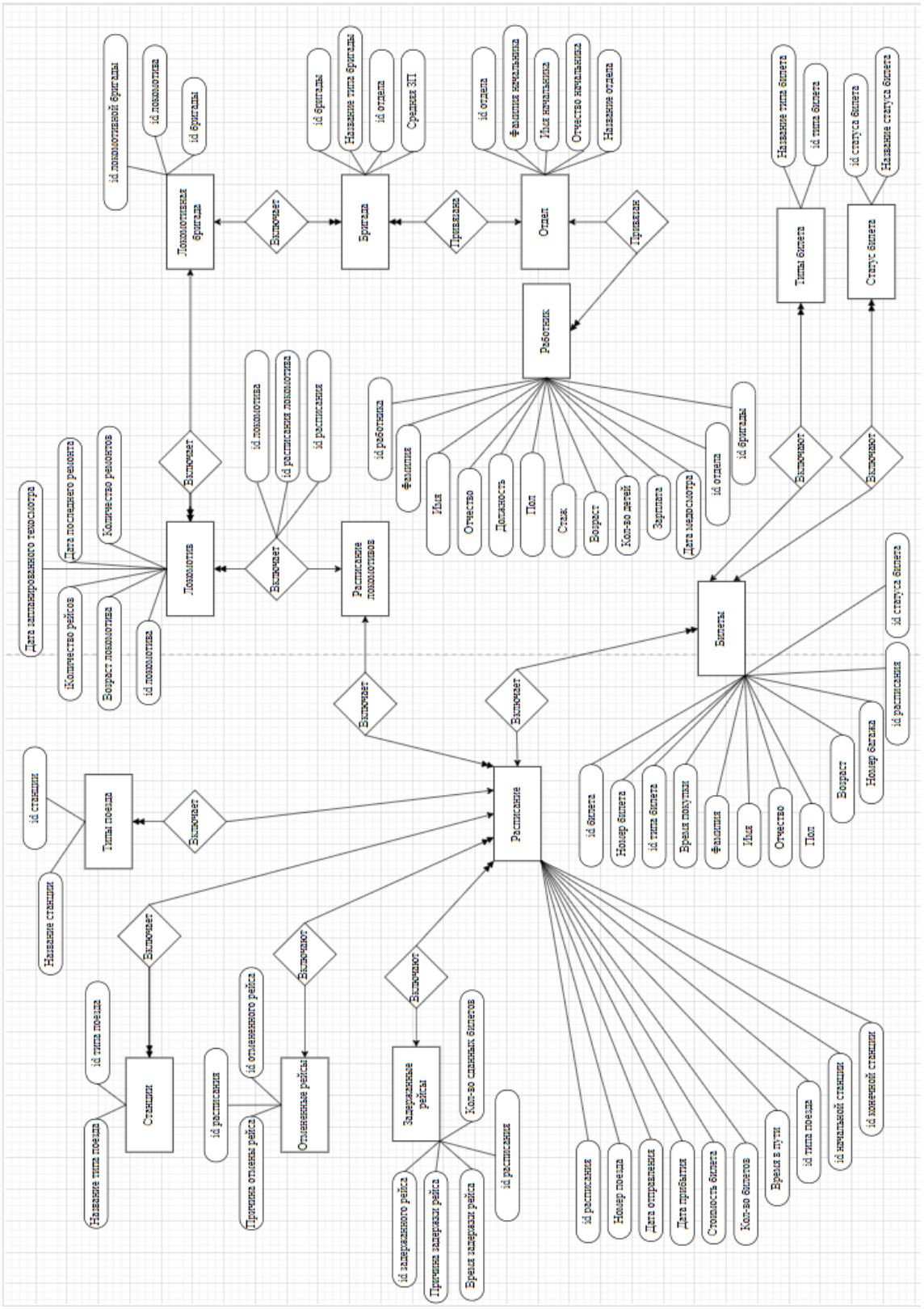


Рисунок А.1 – Концептуальная модель проектируемой базы данных

ПРИЛОЖЕНИЕ Б
(обязательное)
Схема реляционной базы данных

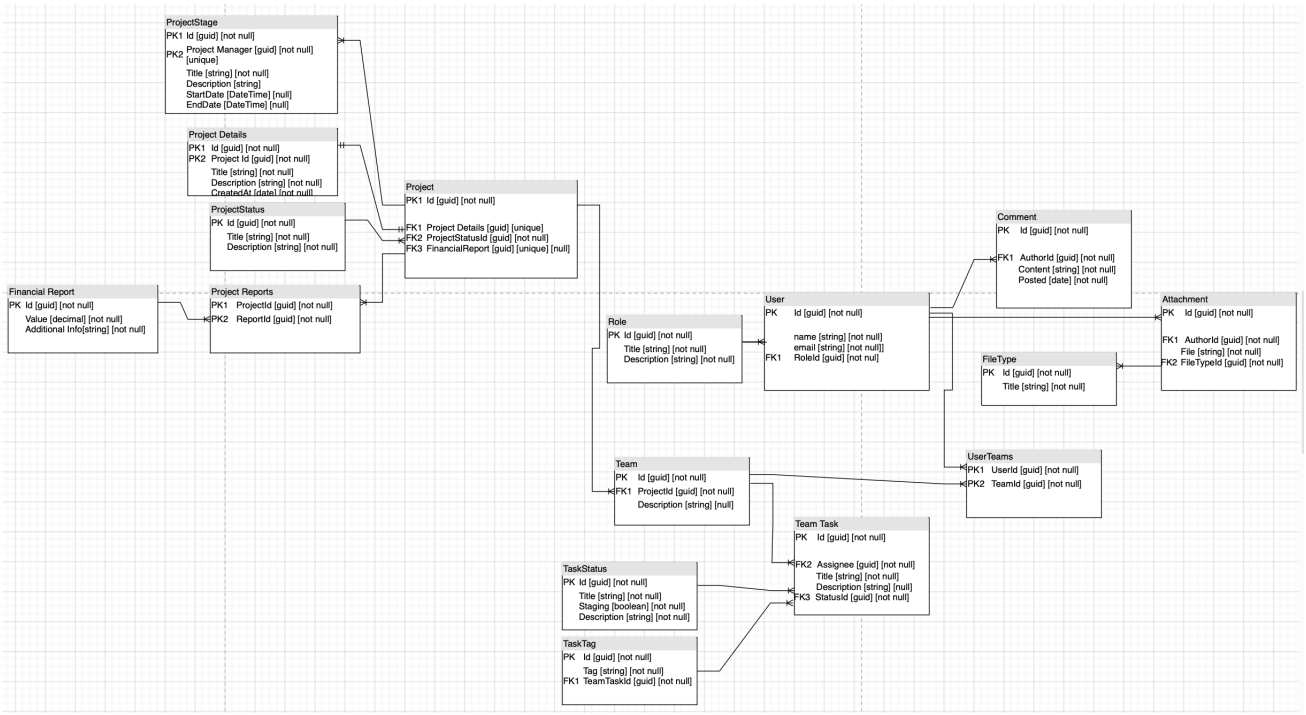


Рисунок Б.1 – Схема реляционной базы данных

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. (ДАННЫЙ АБЗАЦ ДОЛЖНО БЫТЬ У КАЖДОГО СТУДЕНТА).

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 общее число отмененных рейсов полностью, в указанном направлении, по указанному маршруту.

– Получить перечень и общее число задержанных рейсов полностью, по указанной причине, по указанному маршруту, и количество сданных билетов за время задержки.

– 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.

– Получить общее число сданных билетов на указанный рейс, день, маршрут.

Необходимо предусмотреть возможность выдачи пассажиру билета, в котором указано локомотив, маршрут, дата и время отправления, а также наличия багажа.

ПРИЛОЖЕНИЕ Г
(обязательное)
Главная и рабочие формы приложения

В ходе проектирования было определено, что при запуске приложения будет открываться главное окно с общим элементом для вывода содержимого таблиц. Окно входа в систему представлено на рисунке Г.1.

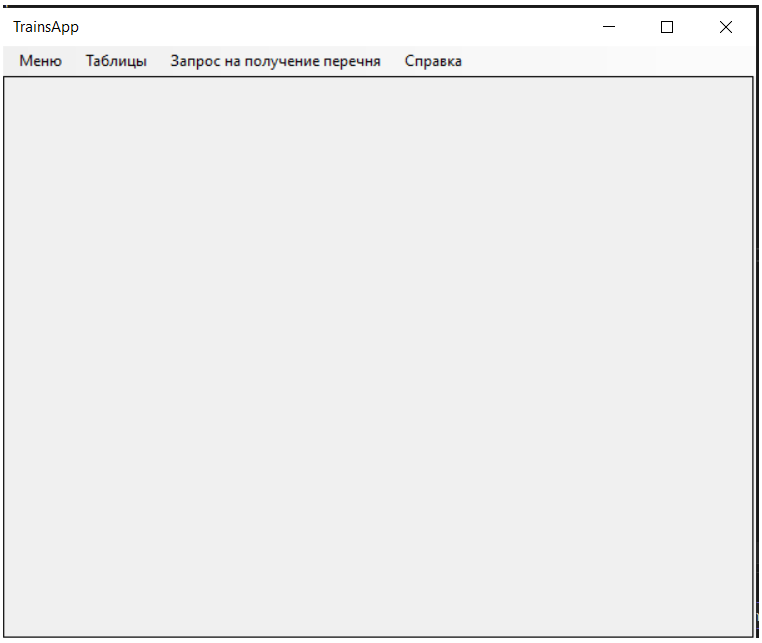


Рисунок Г.1 – Главное окно приложения

Главное окно при нажатии на пункт «Отделы» выпадающего меню «Таблицы» представлено на рисунке Г.2.

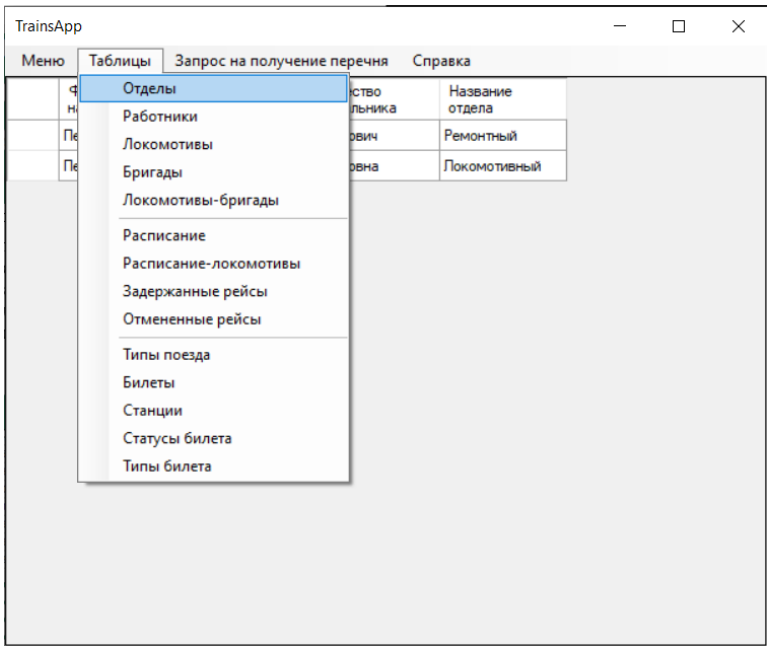


Рисунок Г.2 – Выбор пункта «Отделы»

При выборе из какой таблицы произвести общую выборку приложение определяет какое окно добавления или редактирования открывать после нажатия на пункты «Добавить» и «Редактировать» выпадающего меню «Меню».

На рисунке Г.3 представлено окно добавления новых отделов, так как до этого мы запросили отобразить все отделы базы данных, обращаясь к таблице «Отдели».

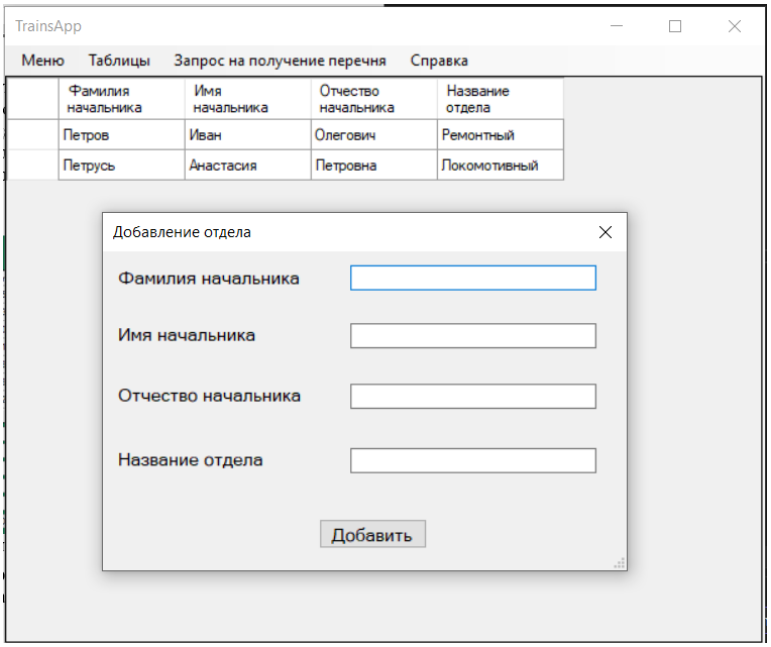


Рисунок Г.3 – Окно добавления записи

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

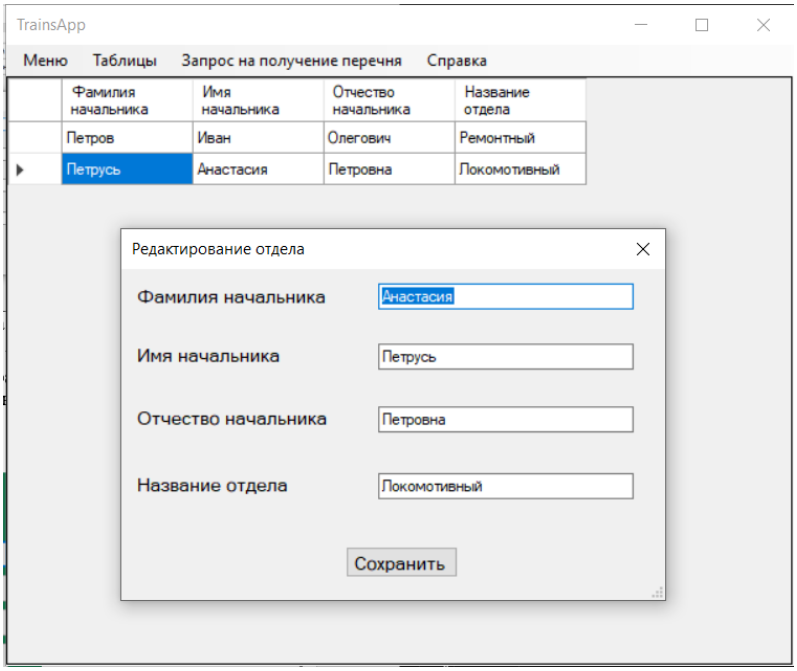


Рисунок Г.4 – Окно редактирования записи

После выбора пункта меню «Удалить» нам будет отображено окно подтверждения, если перед этим мы указали какую запись мы собираемся удалять. После чего, в зависимости от выбора, будет произведено удаление или же закрытие окна подтверждения. Подтверждающее окно представлено на рисунке Г.5.

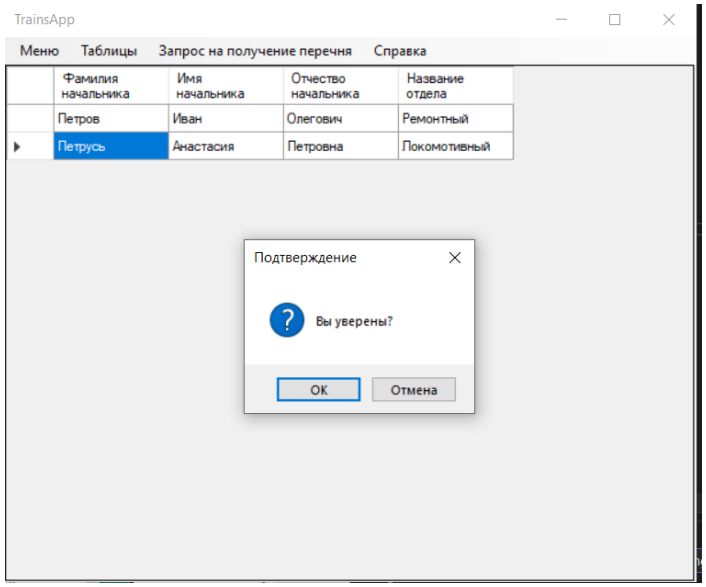


Рисунок Г.5 – Окно подтверждения удаления

Мы имеем возможность произвести выборку по конкретным полям таблицы. На рисунке Г.6 представлено окно, появлению которого предшествует нажатие на пункт «Работников» выпадающего меню «Запрос на получение перечня». На рисунке Г.7 представлен результат ввода данных в текстовое поле и нажатие кнопки «Работники по полу»

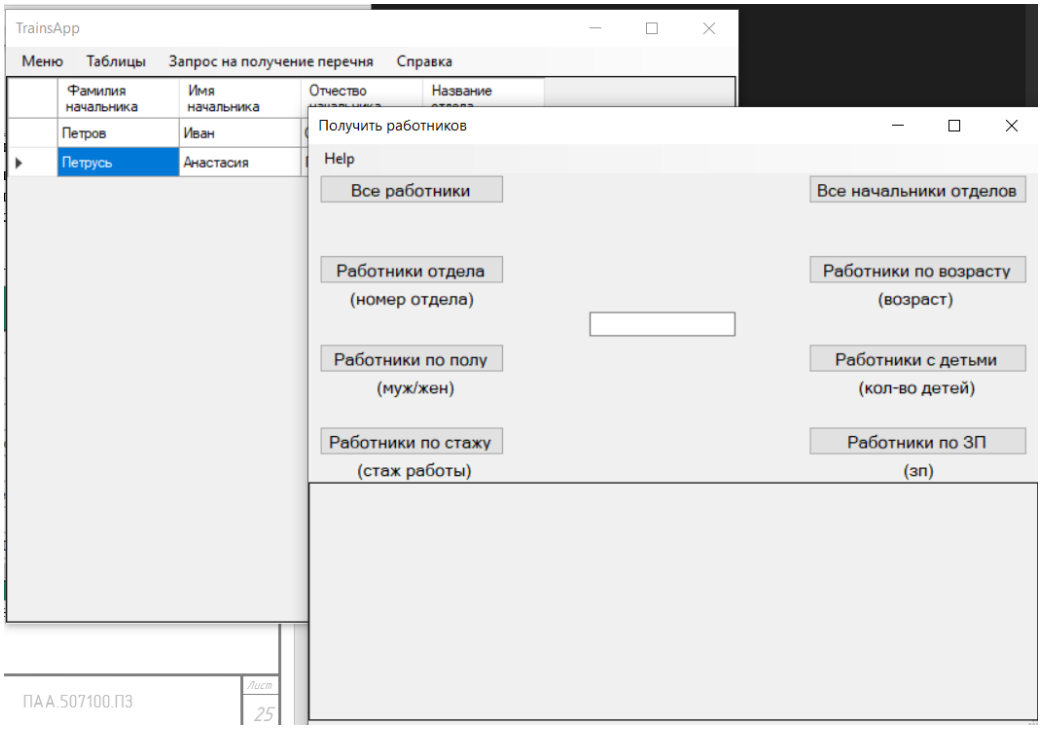
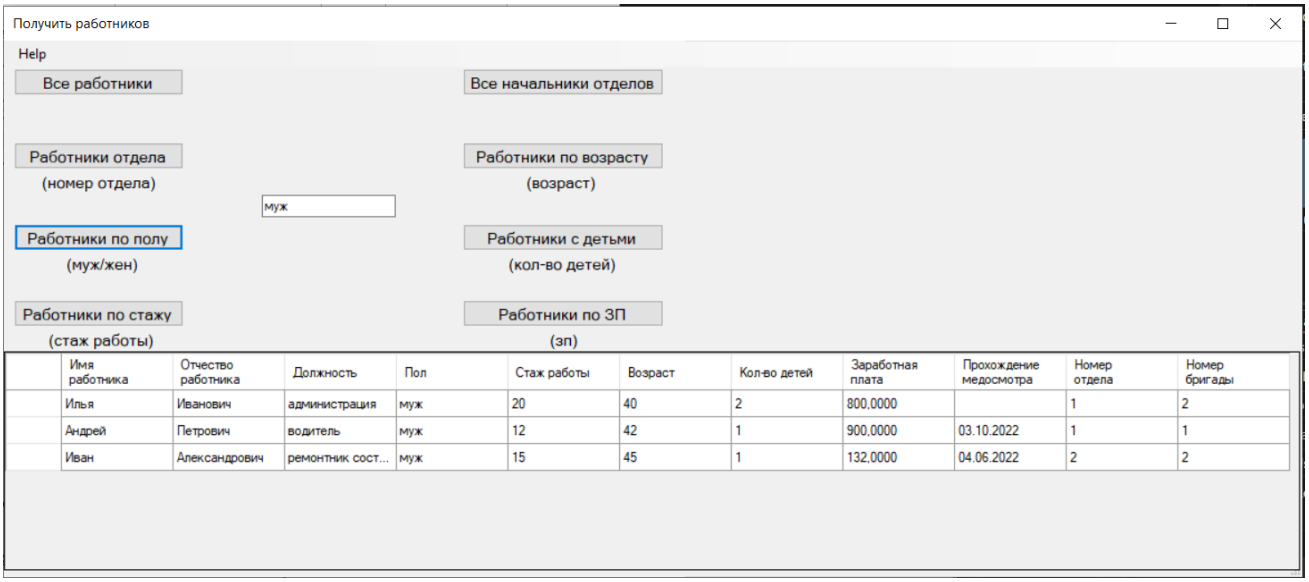


Рисунок Г.6 – Окно для запросов к таблице «Работники»



Имя работника	Отчество работника	Должность	Пол	Стаж работы	Возраст	Кол-во детей	Заработная плата	Прохождение медосмотра	Номер отдела	Номер бригады
Илья	Иванович	администрация	муж	20	40	2	800,0000		1	2
Андрей	Петрович	водитель	муж	12	42	1	900,0000	03.10.2022	1	1
Иван	Александрович	ремонтник сост...	муж	15	45	1	132,0000	04.06.2022	2	2

Рисунок Г.7 – Результат выборки по полу «Пол»

Любые изменения будут сохранятся в используемой базе данных.