**СОДЕРЖАНИЕ**

[ВВЕДЕНИЕ 4](#_Toc120179548)

[1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ МОДЕЛИ 6](#_Toc120179549)

[1.1 Анализ предметной области и выявление необходимого набора сущностей 6](#_Toc120179550)

[1.2 Обоснование требуемого набора атрибутов для каждой сущности и выделение идентифицирующих атрибутов 7](#_Toc120179551)

[1.3 Определение связей между объектами 10](#_Toc120179552)

[1.4 Описание полученной модели на языке инфологического проектирования 11](#_Toc120179553)

[2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ 12](#_Toc120179554)

[2.1 Построение набора необходимых отношений базы данных 12](#_Toc120179555)

[2.2 Задание первичных и внешних ключей определенных отношений 13](#_Toc120179556)

[2.3 Третья нормальная форма 14](#_Toc120179557)

[2.4 Определение ограничений целостности для внешних ключей отношений и для отношений в целом 14](#_Toc120179558)

[2.5 Графическое представление связей между внешними ключами 15](#_Toc120179559)

[3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ 16](#_Toc120179560)

[4 ЗАПИСЬ ВЫРАЖЕНИЙ, УКАЗАННЫХ В ВАРИАНТЕ ЗАДАНИЯ ТИПОВ ЗАПРОСОВ НА ЯЗЫКЕ SQL 20](#_Toc120179561)

[5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ ПРИЛОЖЕНИЯ 23](#_Toc120179562)

[6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ, РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ 25](#_Toc120179563)

[6.1 Разработка и построение интерфейса главной и рабочих форм 25](#_Toc120179564)

[6.2 Построение главного меню и кнопок панели инструментов 25](#_Toc120179565)

[6.3 Выполнение программного кода на Microsoft Visual C# 26](#_Toc120179566)

[ЗАКЛЮЧЕНИЕ 28](#_Toc120179567)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 29](#_Toc120179568)

[ПРИЛОЖЕНИЕ А (обязательное) Концептуальная схема БД 30](#_Toc120179569)

[ПРИЛОЖЕНИЕ Б (обязательное) Схема реляционной базы данных 31](#_Toc120179570)

[ПРИЛОЖЕНИЕ В (обязательное) Описание задания курсовой работы 32](#_Toc120179571)

[ПРИЛОЖЕНИЕ Г (обязательное) Главная и рабочие формы приложения 34](#_Toc120179572)

# ВВЕДЕНИЕ

Основные идеи современной информационной технологии базируются на концепции, согласно которой данные должны быть организованы в базы данных с целью адекватного отображения изменяющегося реального мира и удовлетворения информационных потребностей пользователей. Эти базы данных создаются и функционируют под управлением специальных программных комплексов, называемых системами управления базами данных (СУБД) [1]. <https://hhrd.ru/referat/267>

База данных – это организованная коллекция данных, обычно хранящаяся и доступная для доступа электронно через компьютерные системы. База данных может быть определена как набор задач или процессов, которые улучшают проектирование, разработку, реализацию и поддержку системы управления данными предприятия.

Современные приложения в наши дни взаимодействуют с некоторым источником данных, либо косвенно, используя веб-сервисы.

Проектирование базы данных – это процесс, который включает в себя создание логических и физических моделей предлагаемой системы базы данных. Хороший дизайн базы данных делит вашу информацию на таблицы, основанные на темах, чтобы уменьшить избыточные данные, предоставляет Access информацию, которая требуется для объединения информации в таблицах по мере необходимости, помогает поддерживать и обеспечивать точность и целостность информации, адаптирует потребности в обработке данных и отчетности.

Важно отметить, что правильное проектирование базы данных существенно влияет на общую производительность любого приложения. Принципы, определенные для базы данных, дают четкое представление о поведении любого приложения и о том, как обрабатываются запросы. Кроме того, время обработки приложения значительно сокращается, если правильно реализованы ограничения по проектированию высокоэффективной базы данных.

Проблема избыточности данных может привести к трудностям в навигации по ним и увеличению ресурсов, необходимых для обработки информации в организациях и предприятиях. Для решения этой проблемы необходимо интегрировать автоматизированные базы данных во все уровни технологических процессов и операций.

Для того, чтобы информация в базах данных была достоверной и полезной, она должна соответствовать определенным требованиям. Одним из таких требований является непротиворечивость данных, то есть отсутствие противоречащих друг другу сведений. Другим требованием является целостность данных, то есть сохранение связей между различными элементами информации.

Ситуация, когда в таблице базы данных имеются неоднократно одни и те же данные, называют избыточностью данных [2]. https://eopearhiiv.edu.ee/e-kursused/eucip/arendus\_vk/213\_\_\_\_.html#:~:text=%D0%A1%D0%B8%D1%82%D1%83%D0%B0%D1%86%D0%B8%D1%8F%2C%20%D0%BA%D0%BE%D0%B3%D0%B4%D0%B0%20%D0%B2%20%D1%82%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B5%20%D0%B1%D0%B0%D0%B7%D1%8B,%D0%B6%D0%B5%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D0%B5%2C%20%D0%BD%D0%B0%D0%B7%D1%8B%D0%B2%D0%B0%D1%8E%D1%82%20%D0%B8%D0%B7%D0%B1%D1%8B%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C%D1%8E%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85.

Одним из ключевых требований к базе данных является отсутствие избыточности, что подразумевает устранение дублирования информации в различных частях базы данных. Базы данных создаются с целью хранения и обеспечения прямого доступа к информации, отражающей конкретную предметную область, то есть область интереса или деятельности пользователя. Глубина детализации данных определяется рядом факторов, включая цель использования информации из базы данных и сложность информационных процессов в рамках предметной области.

В компьютере данные базы данных обычно представлены в форме таблиц, похожих на электронные таблицы. Заголовки таблицы, представленные в виде столбцов, называются именами полей, а сами столбцы – полями. Информация, содержащаяся в этих полях, называется значениями полей.

Разработка современного программного обеспечения является сложным и трудоемким процессом, который включает в себя несколько ключевых этапов. Специалисты в этой области должны обладать глубокими знаниями и пониманием методов анализа, проектирования, реализации и тестирования программного продукта.

На этапе анализа специалисты определяют требования к программному продукту, изучают потребности пользователей и определяют основные функции, которые должно выполнять программное обеспечение.

Этап проектирования включает в себя создание детальной схемы того, как будет работать программа. Это может включать в себя создание диаграмм, которые показывают, как различные компоненты программы будут взаимодействовать друг с другом.

На этапе реализации специалисты преобразуют проект в работающий код, используя подходящий язык программирования.

Наконец, на этапе тестирования программное обеспечение проверяется на наличие ошибок и проблем. Это важный этап, который помогает обеспечить, что программа работает правильно и безопасно перед тем, как она будет запущена.В данной курсовой работе поставлена задача базы данных и создания информационной системы кафедры (преподаватели и предметы).

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

# 1 ПОСТРОЕНИЕ ИНФОЛОГИЧЕСКОЙ КОНЦЕПТУАЛЬНОЙ

**МОДЕЛИ**

# 1.1 Анализ предметной области и выявление необходимого набора

**сущностей**

В качестве идеи для проектирования и разработки была выбрана тема «Информационная система учета выдачи книг в библиотеке», которая предназначена для хранения, обработки и предоставления данных о деятельности библиотеки, включающей в себя не только обслуживание посетителей (читателей), но и работу с информацией о мероприятиях, проводимых библиотекой: встречи, выставки и тому подобное.

Система содержит следующее описание исходя из выданного задания к курсовой работе: в таблице пользователей находятся данные о фамилии, имени и отчестве, логине и пароле. Пользователи подразделяются на два типа: администраторов и библиотекарей. Администратор системы может вносить изменения в любые таблицы, а также владеет теми же правами, что и библиотекарь.

Библиотекарь имеет ограниченный, в сравнении с администратором, функционал, и помимо составления актов выдачи и возврата для читателей может только заводить новые абонементы и регистрировать мероприятия. В акт выдачи-возврата входит тип действия (взял ли читатель книгу или вернул), а также сама книга. Книга характеризуется не только названием, но и инвентаризацией, обуславливающей наличие нескольких копий книги, а также характеризуется состоянием: наличие или списание из фонда библиотеки. В каждой книге содержатся произведения, обладающие своим жанром или несколькими жанрами. Каждая книга может представлять из себя сборник произведений, или произведение может быть циклом книг. Книги находятся в различных залах, в которых находятся секции и полки.

Кроме закрепления за определенным абонементом акта выдачи и возврата, в информационной системе также присутствует возможность для вывода определенной информации через запросы.

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

# 1.2 Обоснование требуемого набора атрибутов для каждой сущности и

**выделение идентифицирующих атрибутов**

Определим нужный набор атрибутов для каждой сущности с целью построения инфологической концептуальной модели.

Концептуальная модель данных в базе данных – это высокоуровневое представление структуры базы данных, которое помогает понять потребности или требования к базе данных. Эта модель используется на этапе сбора требований, то есть до того, как разработчики баз данных начнут создавать конкретную базу данных.

Концептуальная модель данных обычно представляет собой упрощенную диаграмму окончательной базы данных, где детали специально игнорируются, чтобы можно было понять общую картину. Она дает разработчику возможность получить обзор системы, которую предстоит спроектировать, не заботясь о деталях ее реализации.

Концептуальная модель данных является наименее технической из трех стадий моделирования данных (концептуальной, логической и физической), но именно на этой стадии у разработчика есть свобода экспериментировать и думать о дизайне, который будет наилучшим образом соответствовать проекту.

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

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

Таблица 1.1 – Описание сущностей

| Таблица | Поле | Ключ | Описание |
| --- | --- | --- | --- |
| Users | UserID | PK | Идентификационный номер пользователя |
| LastName |  | Фамилия пользователя |
| FirstName |  | Имя пользователя |
| MiddleName |  | Отчество (при наличии) |
| UserLogin |  | Логин |
| UserPassword |  | Пароль |
| Librarians | LibrarianID | PK | Идентификационный номер библиотекаря |
| UserID | FK | Идентификационный номер пользователя |
| Administrators | AdministratorID | PK | Идентификационный номер администратора |
| UserID | FK | Идентификационный номер пользователя |
| LibrarianID | PK | Идентификационный номер библиотекаря |

|  |  |  |  |
| --- | --- | --- | --- |
| Таблица | Поле | Ключ | Описание |
| LibraryEvents | EventID | PK | Идентификационный номер мероприятия |
| EventType |  | Тип мероприятия |
| EventName |  | Название мероприятия |
| EventDate |  | Дата мероприятия |
| LibrarianID | FK | Идентификационный номер библиотекаря |
| Subscriptions | SubscriptionID | PK | Идентификационный номер абонемента |
| LastName |  | Фамилия читателя |
| FirstName |  | Имя читателя |
| MiddleName |  | Отчество (при наличии) |
| SubscriptionPasport |  | Номер паспорта читателя |
| VisitorPhoneNumber |  | Номер телефона (при наличии) |
| SubscriptionsType |  | Тип абонемента |
| LibraryRooms | LibraryRoomID | PK | Идентификационный номер библиотечного зала |
| LibraryRoomName |  | Название библиотечного зала |
| Sections | SectionID | PK | Идентификационный номер секции |
| SectionNumber |  | Номер секции |
| LibraryRoomID | FK | Идентификационный номер библиотечного зала |
| Shelves | ShelfID | PK | Идентификационный номер полки |
| ShelfNumber |  | Номер полки |
| SectionID | FK | Идентификационный номер секции |
| Works | WorksID | PK | Идентификационный номер произведения |
| WorkName |  | Название произведения |
| Books | BookID | PK | Идентификационный номер книги |
| BookName |  | Название книги |
| PublisherDate |  | Дата публикации |
| Works\_Books | Works\_BooksID | PK | Идентификационный номер произведений в книгах |
| WorkID | FK | Название произведения |
| BookID | FK | Название книги |
| Books\_Shelves | Books\_ShelvesID | PK | Идентификационный номер книги на полке |
| BookInventoryID | FK | Идентификационный номер инвентаризации книги |
| ShelfID | FK | Идентификационный номер полки |
| Authors | AuthorID | PK | Идентификационный номер автора |
| LastName |  | Фамилия автора |
| FirstName |  | Имя автора |
| MiddleName |  | Отчество (при наличии) |
| Pseudonym |  | Псевдоним (при наличии) |
| Works\_Authors | Works\_AuthorsID | PK | Идентификационный номер работы у автора |
| WorkID | FK | Идентификационный номер работы |
| AuthorID | FK | Идентификационный номер автора |
| Genres | GenreID | PK | Идентификационный номер жанра |
| GenreName |  | Название жанра |
| Works\_Genres | Works\_GenresID | PK |  |
| WorkID | FK |  |
| GenreID | FK |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | btigada | FK | Идентификационный номер таблицы Brigadi |

# 1.3 Определение связей между объектами

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

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

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

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

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

# 1.4 Описание полученной модели на языке инфологического

**проектирования**

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

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

Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Кроме того, в этом контексте равноправно могут использоваться слова «модель базы данных» и «модель предметной области», поскольку такая модель является как образом реальности, так и образом проектируемой базы данных для этой реальности. Конкретный вид и содержание концептуальной модели базы данных определяется выбранным для этого формальным аппаратом. Обычно используются графические нотации, подобные ER-диаграммам.

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

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

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

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

# 2 ПОСТРОЕНИЕ СХЕМЫ РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ

# 2.1 Построение набора необходимых отношений базы данных

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

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

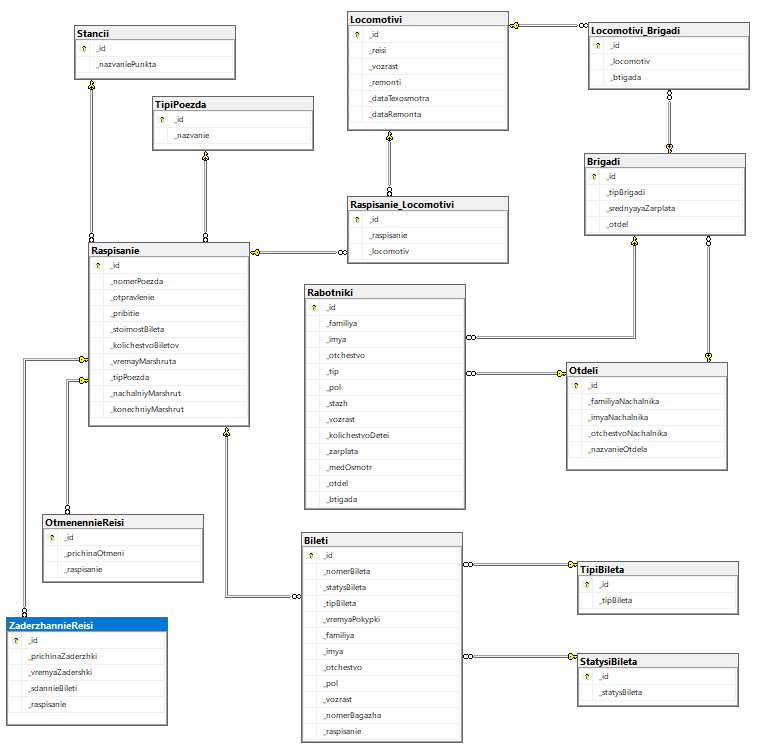


Рисунок 2.1 – Набор необходимых отношений базы данных

# 2.2 Задание первичных и внешних ключей определенных отношений

В каждой таблице базы данных может существовать первичный ключ. Основными требования к первичным ключам являются уникальность и минимальность [3]. <https://cyberleninka.ru/article/n/algoritmy-naznacheniya-pervichnyh-klyuchey-v-zapolnennyh-tablitsah/viewer>

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

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

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

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

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

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

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

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

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

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

Правило целостности внешних ключей. Внешние ключи не должны быть несогласованными, т.е. для каждого значения внешнего ключа должно существовать соответствующее значение первичного ключа в родительском отношении[4]. https://kazedu.com/referat/71716/3

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

Первичные и вторичные ключи представлены в таблице 1.1.

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

# 2.3 Третья нормальная форма

Нормальная форма — требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами (полями таблиц)[5]. https://habr.com/ru/articles/254773/

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

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

- Она должна соответствовать второй нормальной форме.

- Никакой не первичный атрибут не должен транзитивно зависеть от первичного ключа. (ОФОРМИ КАК СПИСОК)

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

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

Повторяющимися являются поля, содержащие одинаковые по смыслу значения. Например, если требуется получить статистику сдачи экзаменов по предметам, можно создать поля для хранения данных об оценке по каждому предмету. Однако в этом случае мы имеем дело с повторяющимися группами.

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

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

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

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

# 2.4 Определение ограничений целостности для внешних ключей

**отношений и для отношений в целом**

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

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

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

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

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

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

# 2.5 Графическое представление связей между внешними

**первичными ключами**

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

# 3 СОЗДАНИЕ СПРОЕКТИРОВАННОЙ БАЗЫ ДАННЫХ

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

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

Таблица 3.1 – Характеристики атрибутов таблицы TipiPoezda

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| id | INT | Идентификационный номер таблицы TipiPoezda |
| nazvanie | NVARCHAR(25) | Название типа поезда |

Таблица 3.2 – Характеристики атрибутов таблицы Stancii

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы Stancii |
| \_nazvaniePunkta | NVARCHAR(25) | Название остановочного пункта |

Таблица 3.3 – Характеристики атрибутов таблицы Raspisanie

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы Raspisanie |
| \_nomerPoezda | INT | Номер поезда (bigint) |
| \_otpravlenie | DATETIME | Время и дата отправления |
| \_pribitie | MONEY | Время и дата прибытия |
| \_stoimostBileta | INT | Стоимость билета |
| \_kolichestvoBiletov | TIME | Количество билетов на данный маршрут |
| \_vremayMarshruta | TIME | Время в пути |
| \_tipPoezda | INT | Идентификационный номер таблицы TipiPoezda |
| \_nachalniyMarshrut | INT | Идентификационный номер таблицы Stancii |
| \_konechniyMarshrut | INT | Идентификационный номер таблицы Stancii |

Таблица 3.4 – Характеристики атрибутов таблицы OtmenennieReisi

| Поле | Тип данных | Описание |
| --- | --- | --- |
| \_id | INT | Идентификационный номер таблицы OtmenennieReisi |
| \_prichinaOtmeni | NVARCHAR(25) | Причина отмени |
| \_raspisanie | INT | Идентификационный номер таблицы Raspisanie |

Таблица 3.5 – Характеристики атрибутов таблицы ZaderzhannieReisi

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы ZaderzhannieReisi |
| \_prichinaZaderzhki | NVARCHAR(25) | Причина задержки |
| \_vremyaZadershki | TIME | Время задержки |
| \_sdannieBileti | INT | Количество сданных билетов |
| \_raspisanie | INT | Идентификационный номер таблицы Raspisanie |

Таблица 3.6 – Характеристики атрибутов таблицы StatysiBileta

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы StatysiBileta |
| \_statysBileta | NVARCHAR(25) | Название статуса билета |

Таблица 3.7 – Характеристики атрибутов таблицы TipiBileta

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы TipiBileta |
| \_tipBileta | NVARCHAR(25) | Название типа билета |

Таблица 3.8 – Характеристики атрибутов таблицы Bileti

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы Bileti |
| \_nomerBileta | BIGINT | Номер билета |
| \_statysBileta | INT | Идентификационный номер таблицы StatysiBileta |
| \_tipBileta | INT | Идентификационный номер таблицы TipiBileta |
| \_vremyaPokypki | TIME | Время покупки билета |
| \_familiya | NVARCHAR(25) | Фамилия покупателя |
| \_imya | NVARCHAR(25) | Имя покупателя |
| \_otchestvo | NVARCHAR(25) | Отчество покупателя |
| \_pol | NVARCHAR(25) | Пол покупателя |
| \_vozrast | INT | Возраст покупателя |
| \_nomerBagazha | INT | Номер багажа (если имеется) |
| \_raspisanie | INT | Идентификационный номер таблицы Raspisanie |

Таблица 3.9 – Характеристики атрибутов таблицы Locomotivi

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы Locomotivi |

Продолжение таблицы 3.9

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_reisi | INT | Количество рейсов |
| \_vozrast | INT | Возраст локомотива |
| \_remonti | INT | Количество ремонтов |
| \_dataTexosmotra | DATE | Дата запланированного техосмотра |
| \_dataRemonta | DATE | Дата последнего ремонта |

Таблица 3.10 – Характеристики атрибутов таблицы Raspisanie\_Locomotivi

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы Raspisanie-Locomotivi |
| \_raspisanie | INT | Идентификационный номер таблицы Raspisanie |
| \_locomotiv | INT | Идентификационный номер таблицы Locomotivi |

Таблица 3.11 – Характеристики атрибутов таблицы Otdeli

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы Otdeli |
| \_familiyaNachalnika | NVARCHAR(25) | Фамилия начальника отдела |
| \_imyaNachalnika | NVARCHAR(25) | Имя начальника отдела |
| \_otchestvoNachalnika | NVARCHAR(25) | Отчество начальника отдела |
| \_nazvanieOtdela | NVARCHAR(25) | Название отдела |

Таблица 3.12 – Характеристики атрибутов таблицы Brigadi

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы Brigadi |
| \_tipBrigadi | NVARCHAR(25) | Типа бригады |
| \_srednyayaZarplata | MONEY | Средняя зарплата в бригаде |
| \_otdel | INT | Идентификационный номер таблицы Otdeli |

Таблица 3.13 – Характеристики атрибутов таблицы Locomotivi\_Brigadi

| Поле | Тип данных | Описание |
| --- | --- | --- |
| \_id | INT | Идентификационный номер таблицы Locomotivi-Brigadi |
| \_locomotiv | INT | Идентификационный номер таблицы Locomotivi |
| \_btigada | INT | Идентификационный номер таблицы Brigadi |

Таблица 3.14 – Характеристики атрибутов таблицы Rabotniki

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| \_id | INT | Идентификационный номер таблицы Rabotniki |
| \_familiya | NVARCHAR(25) | Фамилия работника |
| \_imya | NVARCHAR(25) | Имя работника |
| \_otchestvo | NVARCHAR(25) | Отчество работника |
| \_tip | NVARCHAR(25) | Занимаемая должность |
| \_pol | NVARCHAR(25) | Пол |
| \_stazh | INT | Стаж работы |
| \_vozrast | INT | Возраст |
| \_kolichestvoDetei | INT | Количество детей |
| \_zarplata | MONEY | Зарплата |
| \_medOsmotr | DATE | Дата прохождения медосмотра |
| \_otdel | INT | Идентификационный номер таблицы Otdeli |
| \_btigada | INT | Идентификационный номер таблицы Brigadi |

# 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 \* FROM Rabotniki

SELECT \* FROM Otdeli

SELECT \* FROM Rabotniki

WHERE Rabotniki.\_otdel = 1

SELECT \* FROM Rabotniki

WHERE \_stazh > 20

SELECT \* FROM Rabotniki

WHERE \_pol = 'жен'

SELECT \* FROM Rabotniki

WHERE \_vozrast < 45

SELECT \* FROM Rabotniki

WHERE \_kolichestvoDetei IS NOT NULL AND \_kolichestvoDetei > 1

SELECT \* FROM Rabotniki

WHERE \_zarplata > 800.5001

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 COUNT(Rabotniki.\_id) AS 'Количество рабочих', Otdeli.\_nazvanieOtdela AS 'Отдел' FROM Rabotniki, Otdeli

WHERE Otdeli.\_id = Rabotniki.\_otdel

GROUP BY Otdeli.\_nazvanieOtdela

SELECT COUNT(\*) AS 'Количество рабочих по всем отделам' FROM Rabotniki

WHERE \_otdel IS NOT NULL

SELECT \* FROM Rabotniki, Brigadi

WHERE Rabotniki.\_btigada = Brigadi.\_id AND Brigadi.\_srednyayaZarplata <2000

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 \* FROM Rabotniki

WHERE Rabotniki.\_tip = 'водитель' AND Rabotniki.\_medOsmotr IS NOT NULL

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

Продолжение листинга 4.4

SELECT Raspisanie.\_nachalniyMarshrut, Raspisanie.\_konechniyMarshrut, Locomotivi.\_id FROM Locomotivi, Raspisanie, Raspisanie\_Locomotivi

WHERE \_pribitie >= '2019-02-01 15:30:00.000' AND \_pribitie<='2019-02-03 15:30:00.000' AND

Raspisanie.\_id = Raspisanie\_Locomotivi.\_raspisanie AND Locomotivi.\_id = Raspisanie\_Locomotivi.\_locomotiv

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 \* FROM Locomotivi

WHERE \_dataTexosmotra>='2022-10-06' AND \_dataTexosmotra <='2022-10-8'

SELECT \* FROM Locomotivi

WHERE \_dataRemonta = '2022-10-06'

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 \* FROM Raspisanie

WHERE \_nachalniyMarshrut = '' AND \_konechniyMarshrut = ''

Листинг 4.7 – Получить перечень и общее число отмененных рейсов полностью, в указанном направлении, по указанному маршруту

SELECT \* FROM OtmenennieReisi, Raspisanie, Stancii

WHERE Raspisanie.\_id = OtmenennieReisi.\_raspisanie AND Stancii.\_nazvaniePunkta = 'Туда' AND Raspisanie.\_konechniyMarshrut = Stancii.\_id

Листинг 4.8 – Получить перечень и общее число задержанных рейсов полностью, по указанной причине, по указанному маршруту, и количество сданных билетов за время задержки

SELECT \* FROM ZaderzhannieReisi

WHERE \_prichinaZaderzhki = 'Погодные условия'

Листинг 4.9 – Получить перечень и среднее количество проданных билетов за указанный интервал времени на определённые маршруты, по длительности маршрута, по цене билета

SELECT Bileti.\_nomerBileta FROM Bileti, Raspisanie

WHERE \_vremyaPokypki IS NULL AND Bileti.\_raspisanie = Raspisanie.\_id AND Raspisanie.\_konechniyMarshrut =3

Листинг 4.10 – Получить перечень и общее число маршрутов указанной категории, следующих в определенном напpавлении

SELECT \* FROM Raspisanie

WHERE \_tipPoezda = 1 AND \_konechniyMarshrut = 3

Листинг 4.11 – 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 \* FROM Raspisanie, Bileti

Продолжение листинга 4.11

WHERE Raspisanie.\_otpravlenie<='2022-10-06 03:00:00.000' AND Raspisanie.\_pribitie >='2022-10-06 04:00:00.000' AND Bileti.\_raspisanie = Raspisanie.\_id AND Bileti.\_statysBileta != 1

Листинг 4.12 – Получить перечень и общее число невыкупленных билетов на указанном рейс, день, некоторый маршрут

SELECT FROM Bileti

WHERE \_statysBileta != ''

Листинг 4.13 – Получить общее число сданных билетов на указанный рейс, день, маршрут

SELECT COUNT(\*) AS 'Count' FROM Bileti

WHERE \_statysBileta = 1 AND \_id = 1)

# 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:

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

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

Для объединения базы данных с интерфейсом приложения использовался SqlClient.

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

Для работы был использован Microsoft SQL Server.

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

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

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

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

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

# 6 РЕАЛИЗАЦИЯ ЗАКОНЧЕННОГО ПРИЛОЖЕНИЯ,

**РАБОТАЮЩЕГО С СОЗДАННОЙ БАЗОЙ ДАННЫХ**

# 6.1 Разработка и построение интерфейса главной и рабочих форм

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

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

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

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

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

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

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

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

# 6.2 Построение главного меню и кнопок панели инструментов

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

# 6.3 Выполнение программного кода на Microsoft Visual C#

Далее следует описание работы программы с базой данных. Все необходимые методы для 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

Продолжение листинга 6.1

{

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 2019.

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

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

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

В результате выполнения курсовой работы были спроектированы и реализованы: база данных информационной системы 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**

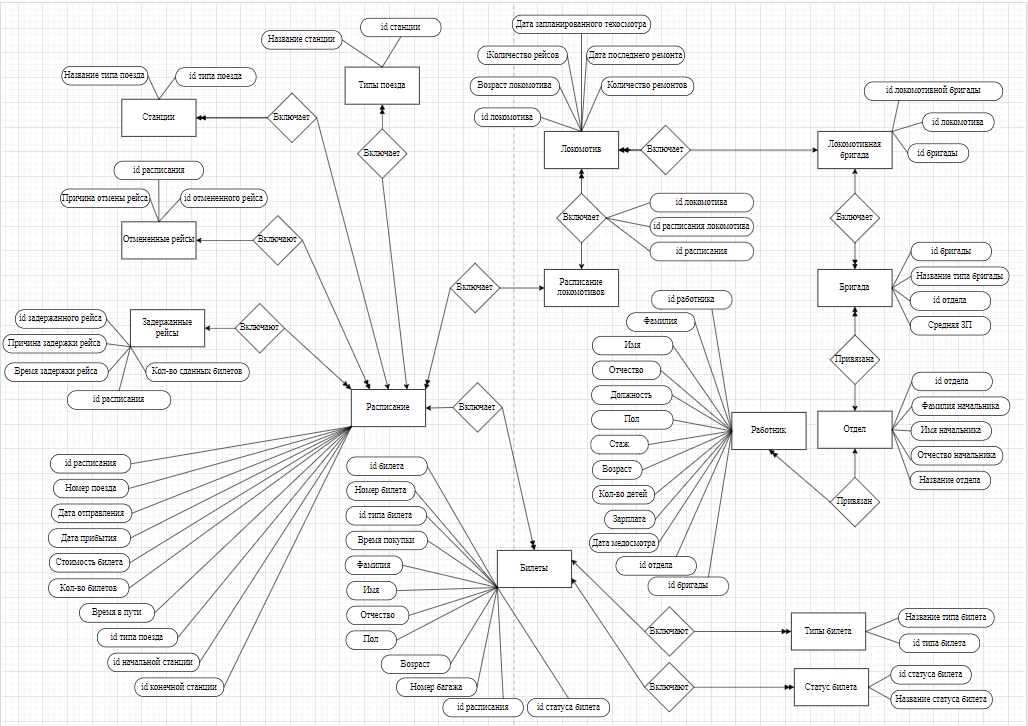


Рисунок А.1 – Концептуальная модель проектируемой базы данных

# ПРИЛОЖЕНИЕ Б

(обязательное)

**Схема реляционной базы данных**

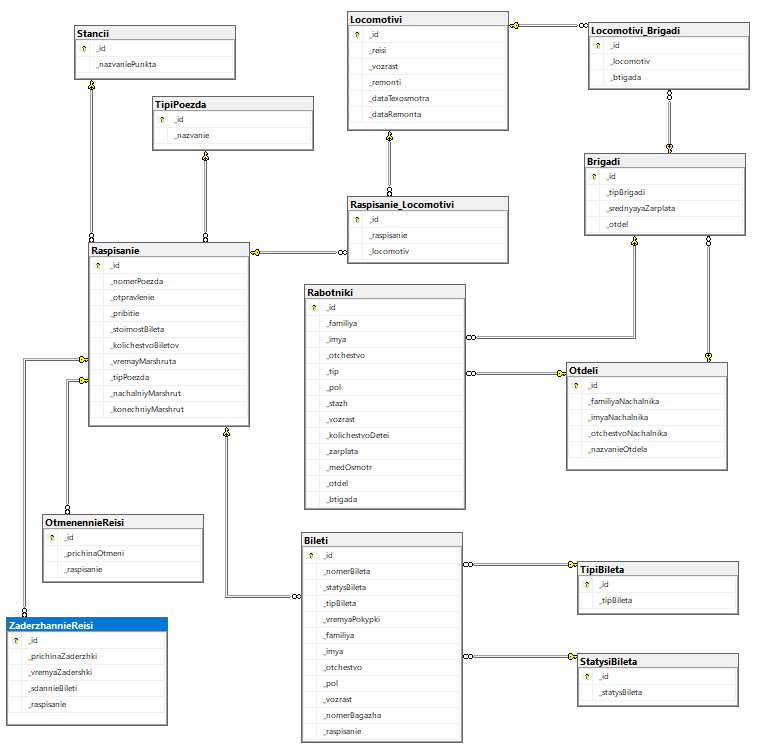


Рисунок Б.1 – Схема реляционной базы данных

# ПРИЛОЖЕНИЕ В

(обязательное)

**Описание задания курсовой работы**

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

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.

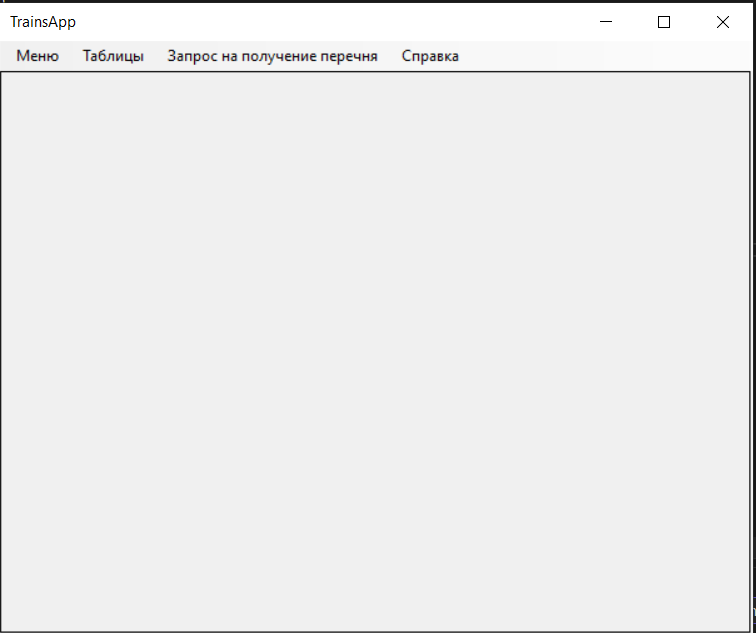


Рисунок Г.1 – Главное окно приложения

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

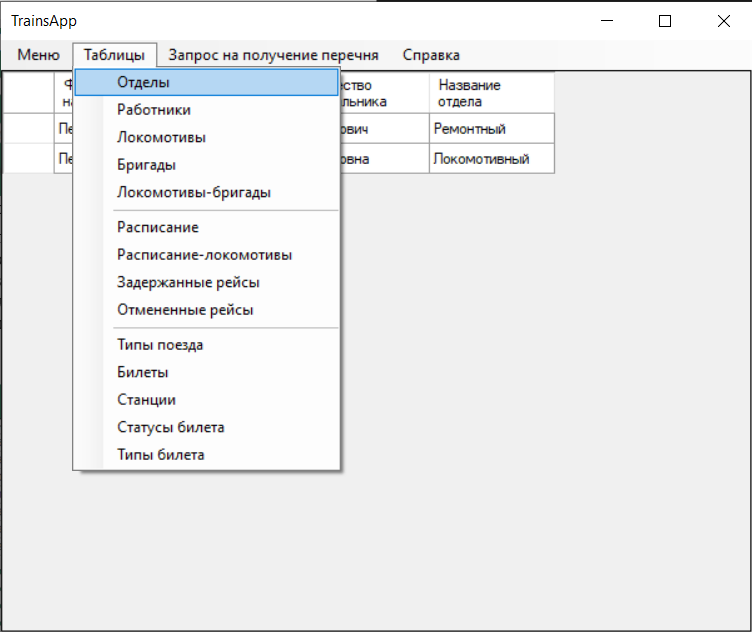


Рисунок Г.2 – Выбор пункта «Отделы»

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».

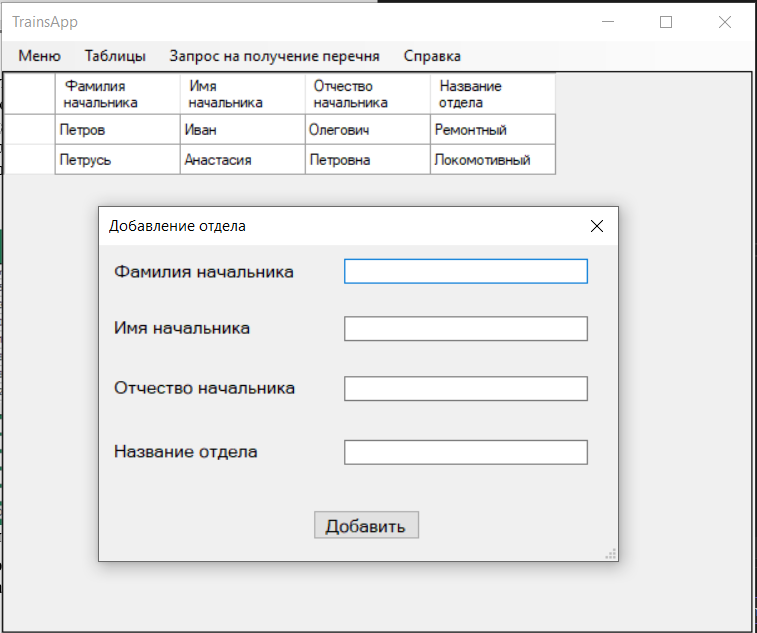


Рисунок Г.3 – Окно добавления записи

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.

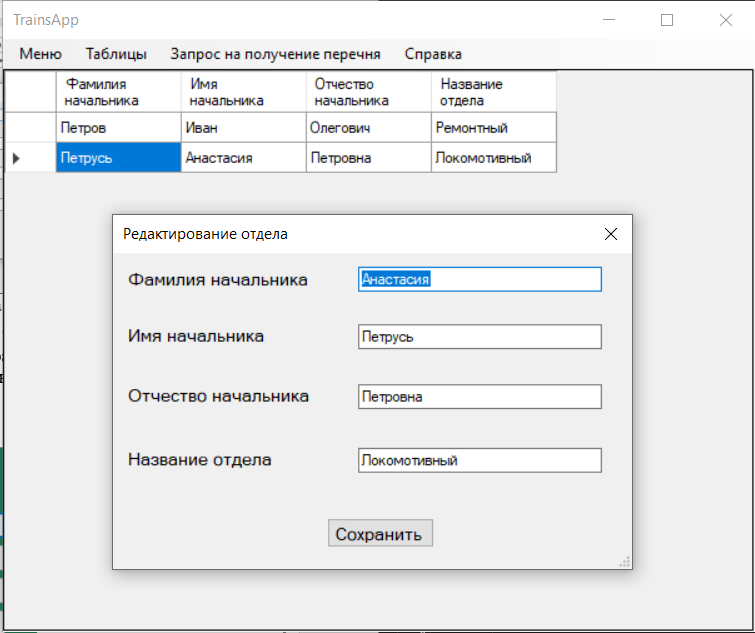


Рисунок Г.4 – Окно редактирования записи

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.

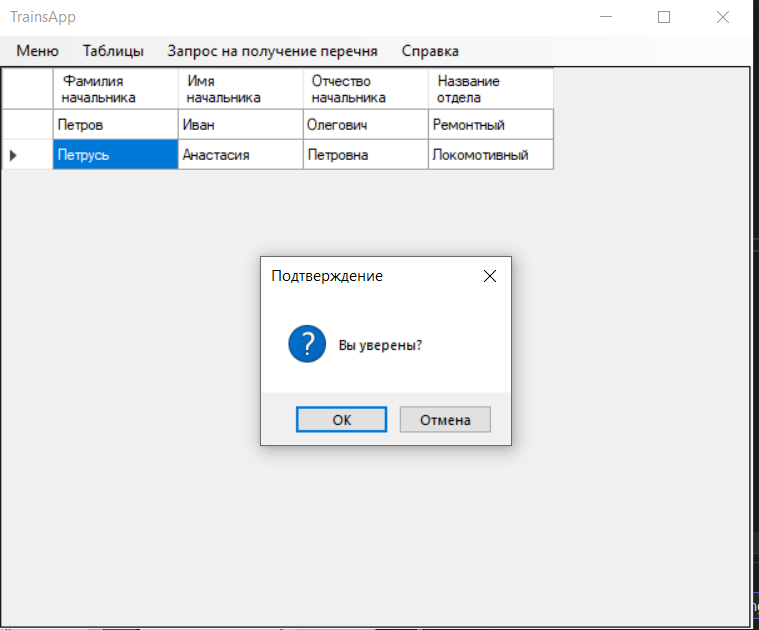


Рисунок Г.5 – Окно подтверждения удаления

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»

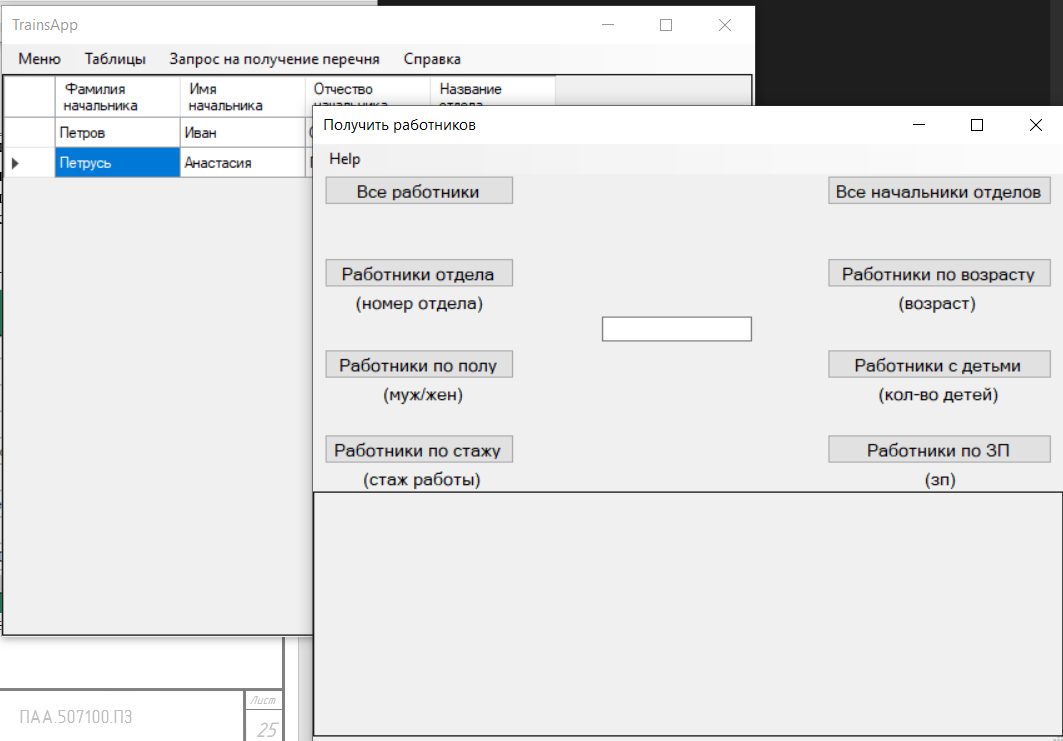


Рисунок Г.6 – Окно для запросов к таблице «Работники»

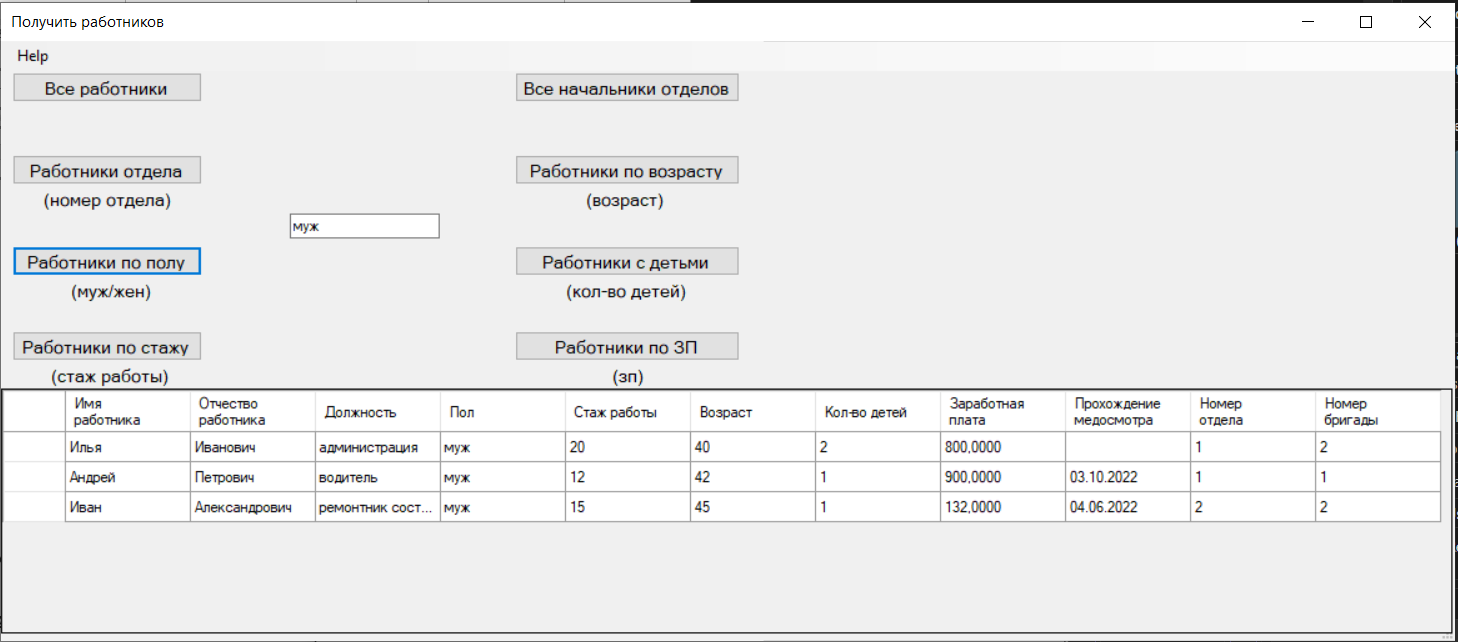


Рисунок Г.7 – Результат выборки по полу «Пол»

Любые изменения будут сохранятся в используемой базе данных.