**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 | Идентификационный номер произведения в жанре |
| WorkID | FK | Идентификационный номер произведения |
| GenreID | FK | Идентификационный номер жанра |
| Publishers | PublisherID | PK | Идентификационный номер издательства |
| PublisherName |  | Наименование издательства |
| Books\_Publishers | Books\_PublishersID | PK | Идентификационный номер книги в издательствах |
| BookID | FK | Идентификационный номер книги |
| PublisherID | FK | Идентификационный номер издательства |
| Acts | ActID | PK | Идентификационный номер акта |
| LibrarianID | FK | Идентификационный номер библиотекаря |
| SubscriptionID | FK | Идентификационный номер абонемента |
| ActionType |  | Тип действия |
| EventDate |  | Дата оформления акта |
| BooksInventorisation | BookInventoryID | PK | Идентификационный номер копии книги в инвентаризации |
| BookID | FK | Идентификационный номер книги |
| IsAvailable |  | Доступность или списанность |
| CopyNumber |  | Уникальный номер копии |
| Acts\_Books | Acts\_BooksID | PK | Идентификационный номер  книги в акте |
| BookInventoryID | FK | Идентификационный номер инвентаризации книги |
| ActID | FK | Идентификационный номер акта |

# 1.3 Определение связей между объектами

Сущность определяется как некоторый объект рассматриваемой предметной области, информация о котором должна быть отражена в базе данных [3]. https://al.cs.msu.ru/system/files/ER\_method.pdf

Связь создается с помощью внешних ключей, которые являются атрибутами или наборами атрибутов, ссылающихся на первичный ключ или уникальный ключ другой таблицы.

Связи в отображении устроены по тому же принципу, что и грамматические структуры: сущности выполняют роль существительных, а связи – глаголов.

Связи делятся на многие ко многим, один ко многим и один к одному.

Связь один к одному образуется, когда ключевой столбец (идентификатор) присутствует в другой таблице, в которой тоже является ключом либо свойствами столбца задана его уникальность (одно и тоже значение не может повторяться в разных строках).

В типе связей один ко многим одной записи первой таблицы соответствует несколько записей в другой таблице.

Если нескольким записям из одной таблицы соответствует несколько записей из другой таблицы, то такая связь называется «многие ко многим» и организовывается посредством связывающей таблицы.

На практике связь «один к одному» наблюдается не часто. Например, она может возникнуть, когда требуется разделить данных одной таблицы на несколько отдельных таблиц с целью безопасности.

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 высокоуровневого представления 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.

Описание инфологической модели включает следующие формы (спецификации) и диаграммы: для статического компонента - P1 и P2, диаграмма инфологической структуры; для динамического компонента - P3 и P4, диаграмма, отображающая систему доступа к субъектам; для функционального компонента - P5 и P6, диаграмма, отображающая систему манипуляций субъекта.

Важно отметить, что правильное проектирование инфологической модели 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.

Концептуальное проектирование включает в себя построение концептуальной модели, основанной на ранее определенных требованиях, но более близкой к окончательной физической модели. Обычно используется концептуальная модель, называемая моделью сущность-связь. Термины «семантическая модель», «концептуальная модель» и «инфологическая модель» являются синонимами. Обычно используются графические нотации, подобные ER-диаграммам.

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

Процесс проектирования схемы реляционной базы данных включает определение набора отношений, которые составляют базу данных. Этот набор отношений должен вместить всю информацию, которая будет храниться в базе данных. Концептуальная модель – это первый шаг в процессе проектирования базы данных. Эта модель используется для определения требований к базе данных, прежде чем разработчики начнут создавать конкретную базу данных.

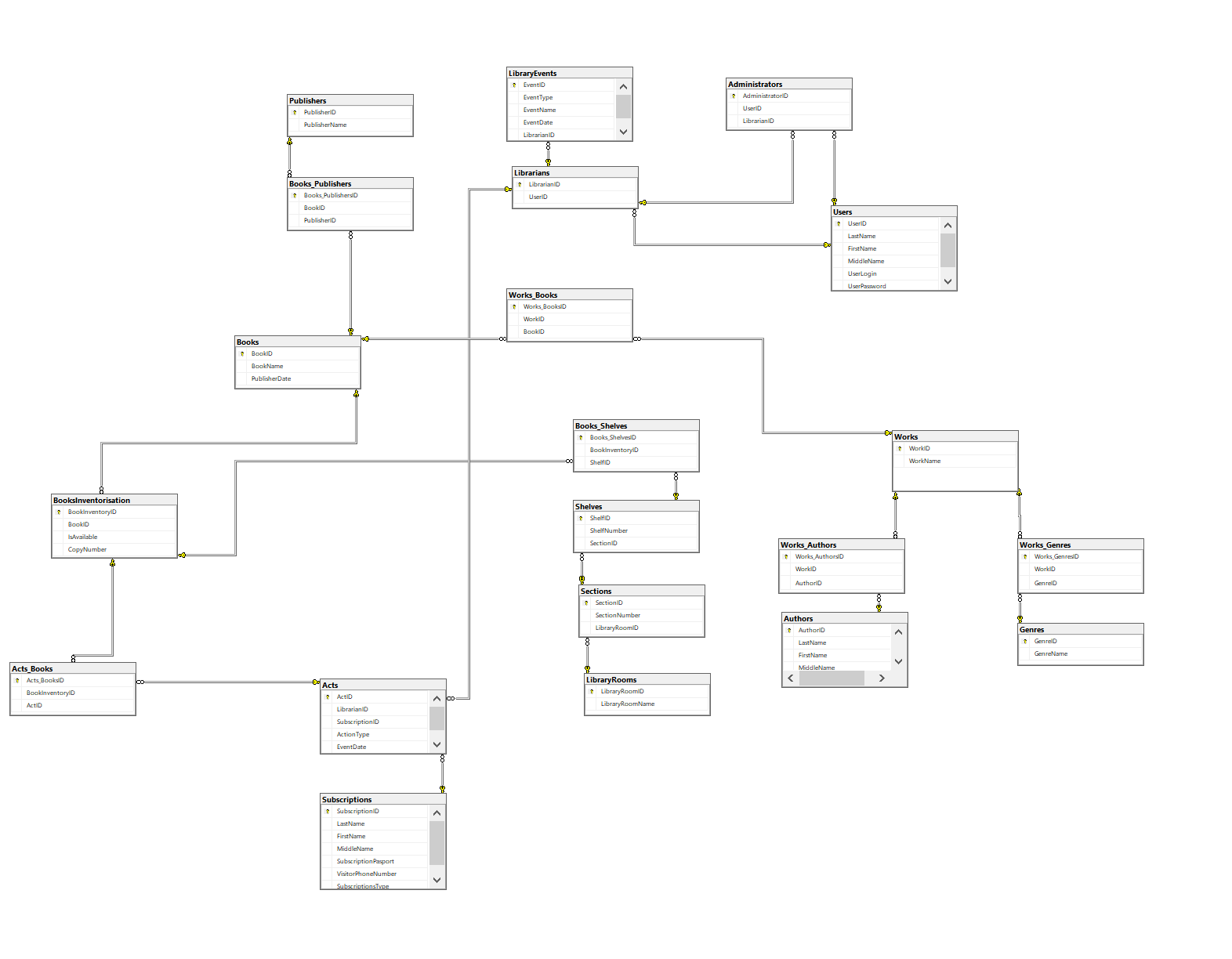
На основе полученной концептуальной модели следует определить набор необходимых отношений. На рисунке 2.1 представлены отношения базы данных. Каждое отношение в базе данных имеет два конца. Конец связи, обозначенный ключом, указывает на то, что сущность на этом конце является главной. Это означает, что эта сущность может быть связана с одной или несколькими другими сущностями. Если все отношения между сущностями установлены как «один ко многим», то конец связи с ключом подразумевает, что данная сущность может быть использована несколько раз в контексте зависимой сущности.

Рисунок 2.1 – Набор необходимых отношений базы данных

# 

# 2.2 Задание первичных и внешних ключей определенных отношений

Каждая таблица в базе данных может иметь первичный ключ. Главными характеристиками первичных ключей являются их уникальность и минимальность.

Минимальность рассматривается с двух точек зрения: первая связана с объемом памяти, выделенным для хранения значений атрибутов, входящих в первичный ключ. Вторая точка зрения на минимальность первичного ключа подразумевает, что в состав ключа не включаются атрибуты с уникальными значениями.

В связанных таблицах первичный ключ одной (родительской) таблицы становится внешним ключом в другой (дочерней) таблице. Внешний ключ в дочерней таблице ссылается на данные родительской таблицы.

Первичный ключ служит уникальным идентификатором записи в реляционной модели данных. Он также используется для связывания таблиц. В связанных таблицах первичный ключ одной (родительской) таблицы становится внешним ключом в другой (дочерней) таблице. Внешний ключ в дочерней таблице ссылается на данные 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».

Выбор первичного ключа в базе данных – это важный шаг, который требует учета нескольких принципов:

Уникальность: все записи в первичном ключе должны быть уникальными. Это означает, что ни одна другая строка в таблице не имеет такое же значение (или комбинацию значений) в этом столбце (или комбинации столбцов).

Не допускаются значения NULL: первичный ключ не может содержать значений NULL. Это гарантирует, что каждая строка в таблице может быть однозначно идентифицирована.

Минимальность: если используется составной первичный ключ, ни один столбец или меньшая комбинация столбцов не должны уникально идентифицировать каждую строку. Другими словами, если убрать любой столбец из PK, комбинация перестанет быть уникальной.

Неизменность: значения первичного ключа не должны меняться. Изменение значения столбца первичного ключа потребует обновления всех ссылок на этот ключ.

Ограничения на размер: составной первичный ключ не может превышать 16 столбцов и длину ключа 900 байт.

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 – это такое требование, которое предъявляется к структуре таблиц в теории реляционных баз данных для устранения из базы избыточных функциональных зависимостей между атрибутами.

Третья нормальная форма. Таблица находится в третьей нормальной форме, если она удовлетворяет определению второй нормальной формы и ни одно из ее не ключевых полей не зависит функционально от любого другого не ключевого поля [5]. https://works.doklad.ru/view/6jPlKZ4MEYw/all.html

Для того, чтобы таблица соответствовала третьей нормальной форме, она должна удовлетворять следующим условиям: она должна соответствовать второй нормальной форме. А также никакой не первичный атрибут не должен транзитивно зависеть от первичного ключа.

Третья нормальная форма (3НФ) играет ключевую роль в проектировании баз данных, поскольку она способствует уменьшению дублирования данных, предотвращает аномалии данных и обеспечивает целостность данных. Это также облегчает управление данными и улучшает производительность базы данных.

Первая нормальная форма (1НФ) требует, чтобы все атрибуты были простыми, а все используемые домены содержали только скалярные значения. Не должно быть повторяющихся строк в таблице. Повторяющимися считаются поля, содержащие одинаковые по смыслу значения.

Вторая нормальная форма (2НФ) требует, чтобы отношение находилось в 1НФ и каждый не ключевой атрибут неприводимо зависел от первичного ключа (ПК).

3НФ 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.

Учитывая, что все атрибуты наших отношений являются атомарными, каждое отношение имеет первичный ключ, в отношениях базы данных отсутствуют зависимости между неключевыми атрибутами, а также отсутствуют зависимости неключевых атрибутов от части составного ключа, мы можем сделать вывод, что отношения базы данных соответствуют третьей нормальной форме.

# 2.4 Определение ограничений целостности для внешних ключей

**отношений и для отношений в целом**

Ограничения целостности в базах данных представляют собой специализированные инструменты, основная цель которых – предотвращение внесения недопустимых данных в базу данных. Ограничение целостности отношений подразумевает, что в любом отношении не должно быть записей с одинаковыми значениями первичного ключа. Другими словами, каждая запись в отношении должна быть уникальной по сравнению с любой другой записью в том же отношении. Это требование автоматически выполняется, если в системе соблюдаются основные 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 в базе данных – это такой набор правил, которые ограничивают все возможные состояния базы данных и переходы между этими состояниями. Они определяют, какие значения могут быть вставлены в таблицы базы данных, включая ограничения для внешних ключей отношений и для отношений в целом. Ограничения целостности для внешних ключей гарантируют, что значения внешнего ключа всегда присутствуют в таблице, на которую он ссылается. Ограничения целостности для отношений в целом гарантируют, что значения в таблице уникальны и не дублируются.

Для автоматического обновления связанных полей при изменении в основной таблице, необходимо настроить обеспечение целостности данных и каскадное обновление связанных полей. Чтобы соответствовать требованиям, необходимо обеспечить соответствие между типами вводимых данных и типами столбцов в таблицах, а также гарантировать, что все обязательные поля в таблицах заполнены, 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 существуют различные инструменты, которые помогают разработчику минимизировать возможность нарушения целостности данных базы, такие как триггеры, проверки, уникальность и другие.

# 2.5 Графическое представление связей между внешними

**первичными ключами**

Графическое представление связей между внешними и первичными ключами – это важный элемент проектирования баз данных. Оно помогает понять, как данные связаны между различными таблицами в базе данных.

Внешние ключи в таблице базы данных обычно указывают на первичные ключи в другой таблице, создавая таким образом связь между двумя таблицами. Это позволяет избежать дублирования данных и обеспечивает целостность данных.

Графическое представление этих связей обычно осуществляется с помощью ER-диаграмм (Entity-Relationship diagrams), которые показывают сущности (таблицы) и связи между ними.

На такой диаграмме отображаются определенным образом таблицы и связи между ними. На концах соединительных линий обычно находятся символы, показывающие тип связи (один ко многим, многие ко многим и т.д.).

Дополнительно стоит отметить, что ER-диаграммы создаются на основе трех основных концепций: сущностей, атрибутов и отношений.

Важно понимать, что такая диаграмма включает в себя множество специализированных символов, и их значения делают эту модель уникальной. Ее цель – представить инфраструктуру структуры объекта.

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 2019 была реализована спроектированная ранее база данных станции. Данная система была выбрана по ряду весомых причин: широкое распространение, наличие свободно распространяемых сборок, наличие высококачественных программных средств разработки, позволяющих создавать разного вида приложения, которые, в свою очередь, смогут использовать базы данных Microsoft SQL Server.

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

Таблица 3.1 – Характеристики атрибутов таблицы Users

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| UserID | INT IDENTITY | Идентификационный номер пользователя |
| LastName | NVARCHAR(50) | Фамилия пользователя |
| FirstName | NVARCHAR(50) | Имя пользователя |
| MiddleName | NVARCHAR(50) | Отчество (при наличии) |
| UserLogin | NVARCHAR(50) | Логин |
| UserPassword | NVARCHAR(50) | Пароль |

Таблица 3.2 – Характеристики атрибутов таблицы Administrators

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| AdministratorID | INT IDENTITY | Идентификационный номер администратора |
| UserID | INT, FOREIGN KEY | Идентификационный номер пользователя |
| LibrarianID | INT, FOREIGN KEY | Идентификационный номер библиотекаря |

Таблица 3.3 – Характеристики атрибутов таблицы Librarians

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| LibrarianID | INT IDENTITY | Идентификационный номер библиотекаря |
| UserID | INT, FOREIGN KEY | Идентификационный номер пользователя |

Таблица 3.4 – Характеристики атрибутов таблицы LibraryEvents

| Поле | Тип данных | Описание |
| --- | --- | --- |
| EventID | INT | Идентификационный номер мероприятия |
| EventType | NVARCHAR(50) | Тип мероприятия |
| EventName | NVARCHAR(50) | Наименование мероприятия |
| EventDate | DATE | Дата мероприятия |
| LibrarianID | INT, FOREIGN KEY | Идентификационный номер библиотекаря |

Таблица 3.5 – Характеристики атрибутов таблицы Subscriptions

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| SubscriptionID | INT | Идентификационный номер абонемента |
| LastName | NVARCHAR(50) | Фамилия читателя |
| FirstName | NVARCHAR(50) | Имя читателя |
| MiddleName | NVARCHAR(50) | Отчество (при наличии) |
| SubscriptionPasport | NVARCHAR(50) | Данные паспорта |
| VisitorPhoneNumber | NVARCHAR(20) | Телефон (при наличии) |
| SubscriptionsType | NVARCHAR(20) | Тип абонемента |

Таблица 3.6 – Характеристики атрибутов таблицы LibraryRooms

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| LibraryRoomID | INT | Идентификационный номер библиотечного зала |
| LibraryRoomName | NVARCHAR(50) | Наименование зала |

Таблица 3.7 – Характеристики атрибутов таблицы Sections

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| SectionID | INT | Идентификационный номер секции |
| SectionNumber | INT | Номер секции |
| LibraryRoomID | INT, FOREIGN KEY | Идентификационный номер зала |

Таблица 3.8 – Характеристики атрибутов таблицы Shelves

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| ShelfID | INT | Идентификационный номер полки |
| ShelfNumber | INT | Номер полки |
| SectionID | INT, FOREIGN KEY | Идентификационный номер секции |

Таблица 3.9 – Характеристики атрибутов таблицы Works

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| WorkID | INT | Идентификационный номер произведения |
| WorkName | NVARCHAR(100) | Название произведения |

Таблица 3.10 – Характеристики атрибутов таблицы Books

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| BookID | INT | Идентификационный номер книги |
| BookName | NVARCHAR(100) | Название книги |
| PublisherDate | DATE | Дата публикации |

Таблица 3.11 – Характеристики атрибутов таблицы Works\_Books

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Works\_BooksID | INT | Идентификационный номер произведения в книге или книги в произведении |
| WorkID | INT, FOREIGN KEY | Идентификационный номер произведения |
| BookID | INT, FOREIGN KEY | Идентификационный номер книги |

Таблица 3.12 – Характеристики атрибутов таблицы Books\_Shelves

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Books\_ShelvesID | INT | Идентификационный номер книги на полке |
| BookInventoryID | INT, FOREIGN KEY | Идентификационный номер инвентаризации |
| ShelfID | INT, FOREIGN KEY | Идентификационный номер полки |

Таблица 3.13 – Характеристики атрибутов таблицы Authors

| Поле | Тип данных | Описание |
| --- | --- | --- |
| AuthorID | INT | Идентификационный номер автора |
| LastName | NVARCHAR(50) | Фамилия автора |
| FirstName | NVARCHAR(50) | Имя автора |
| MiddleName | NVARCHAR(50) | Отчество (при наличии) |
| Pseudonym | NVARCHAR(50) | Псевдоним (при наличии) |

Таблица 3.14 – Характеристики атрибутов таблицы Works\_Authors

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Works\_AuthorsID | INT | Идентификационный номер произведения у автора |
| WorkID | INT, FOREIGN KEY | Идентификационный номер произведения |
| AuthorID | INT, FOREIGN KEY | Идентификационный номер автора |

Таблица 3.15 – Характеристики атрибутов таблицы Works\_Genres

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Works\_GenresID | INT | Идентификационный номер произведения в жанре или же жанров в произведениях |
| WorkID | INT, FOREIGN KEY | Идентификационный номер произведения |
| GenreID | INT, FOREIGN KEY | Идентификационный номер жанра |

Таблица 3.16 – Характеристики атрибутов таблицы Publishers

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| PublisherID | INT | Идентификационный номер издательства |
| PublisherName | NVARCHAR(50) | Наименование издательства |

Таблица 3.17 – Характеристики атрибутов таблицы Books\_Publishers

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Books\_PublishersID | INT | Идентификационный номер книги у издательства |
| BookID | INT, FOREIGN KEY | Идентификационный номер книги |
| PublisherID | INT, FOREIGN KEY | Идентификационный номер издательства |

Таблица 3.18 – Характеристики атрибутов таблицы Acts

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| ActID | INT | Идентификационный номер акта |
| LibrarianID | INT, FOREIGN KEY | Идентификационный номер библиотекаря |
| SubscriptionID | INT, FOREIGN KEY | Идентификационный номер абонемента |
| ActionType | NVARCHAR(20) | Тип действия |
| EventDate | DATE | Дата регистрации акта |

Таблица 3.19 – Характеристики атрибутов таблицы BooksInventorisation

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| BookInventoryID | INT | Идентификационный номер инвентаризации книги |
| BookID | INT, FOREIGN KEY | Идентификационный номер книги |
| IsAvailable | BIT | Доступность или списанность копии книги в фонде библиотеки |
| CopyNumber | INT | Уникальный номер копии книги |

Таблица 3.20 – Характеристики атрибутов таблицы Acts\_Books

|  |  |  |
| --- | --- | --- |
| Поле | Тип данных | Описание |
| Acts\_BooksID | INT | Идентификационный номер актов для книг |
| BookInventoryID | INT, FOREIGN KEY | Идентификационный номер инвентаризации книги |
| ActID | INT, FOREIGN KEY | Идентификационный номер акта |

# 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 – Список книг, опубликованных в указанном году

CREATE PROCEDURE GetBooksByYear

@Year INT

AS

BEGIN

SELECT BookName

FROM Books

WHERE YEAR(PublisherDate) = @Year;

END

Листинг 4.2 – Список книг по указанному жанру, к которому подходят произведения, содержащиеся в этих книгах

CREATE PROCEDURE GetBooksByGenre

@Genre NVARCHAR(50)

AS

BEGIN

SELECT DISTINCT B.BookName

FROM Books B

JOIN Works\_Books WB ON B.BookID = WB.BookID

JOIN Works W ON WB.WorkID = W.WorkID

JOIN Works\_Genres WG ON W.WorkID = WG.WorkID

JOIN Genres G ON WG.GenreID = G.GenreID

WHERE G.GenreName = @Genre;

END

Листинг 4.3 – Перечень обслуженных читателей указанным библиотекарем в указанный промежуток дат

CREATE PROCEDURE GetReadersByLibrarianAndPeriod

@LibrarianID INT,

@StartDate DATE,

@EndDate DATE

AS

BEGIN

SELECT DISTINCT S.LastName, S.FirstName

FROM Subscriptions S

JOIN Acts A ON S.SubscriptionID = A.SubscriptionID

WHERE A.LibrarianID = @LibrarianID AND A.EventDate BETWEEN @StartDate AND @EndDate;

END

Листинг 4.4 – Перечень общей нагрузки работы библиотекарей в числе обслуженных уникальных читателей за указанный промежуток времени

CREATE PROCEDURE GetLibrarianWorkload

@StartDate DATE,

@EndDate DATE

AS

BEGIN

SELECT L.LibrarianID, COUNT(DISTINCT A.SubscriptionID) AS NumberOfReaders

FROM Librarians L

JOIN Acts A ON L.LibrarianID = A.LibrarianID

WHERE A.EventDate BETWEEN @StartDate AND @EndDate

GROUP BY L.LibrarianID;

END

Листинг 4.5 – Перечень авторов, произведений которых в фонде библиотеки находится больше, чем указанное количество

CREATE PROCEDURE GetAuthorsWithMoreWorksThan(@WorkCount INT)

AS

BEGIN

SELECT a.LastName, a.FirstName, a.MiddleName

FROM Authors a

JOIN Works\_Authors wa ON a.AuthorID = wa.AuthorID

GROUP BY a.LastName, a.FirstName, a.MiddleName

HAVING COUNT(wa.WorkID) > @WorkCount

END

Листинг 4.6 – Отсортированные списки публикованных книг от старых к новым или от новых к старым по выбранному варианту сортировки

CREATE PROCEDURE GetBooksSortedByDate(@SortOrder NVARCHAR(4))

AS

BEGIN

IF @SortOrder = 'ASC'

BEGIN

SELECT BookName, PublisherDate

FROM Books

ORDER BY PublisherDate ASC

END

ELSE IF @SortOrder = 'DESC'

BEGIN

SELECT BookName, PublisherDate

FROM Books

ORDER BY PublisherDate DESC

END

ELSE

BEGIN

PRINT 'Неверный параметр сортировки. Используйте ASC для сортировки от старых к новым или DESC для сортировки от новых к старым.'

END

END

Листинг 4.7 – Список читателей, которые не брали и не возвращали книги после определенной даты

CREATE PROCEDURE GetUsersWithNoActsAfterMonth(@Month INT)

AS

BEGIN

SELECT s.LastName, s.FirstName, s.MiddleName

FROM Subscriptions s

WHERE NOT EXISTS (

SELECT 1

FROM Acts a

WHERE a.SubscriptionID = s.SubscriptionID AND MONTH(a.EventDate) > @Month

)

# 5 ВЫБОР И ОСНОВАНИЕ СРЕДСТВ РАЗРАБОТКИ

**ПРИЛОЖЕНИЯ**

Для создания приложения была выбрана среда разработки Microsoft Visual Studio 2022, а в качестве языка программирования был выбран C#. Этот язык программирования является одним из наиболее популярных языков, 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.

C# 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. Для объединения базы данных с интерфейсом приложения использовался SqlClient1.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.

В ходе работы была использована система управления базами данных Microsoft SQL Server. Это мощная система, которая предлагает ряд значительных преимуществ, включая безопасность данных, простоту настройки, масштабируемость, поддержку, производительность и надежность. Microsoft SQL Server гарантирует безопасность базы данных, используя разнообразные методы, включая шифрование данных и контроль доступа, что делает его идеальным выбором для хранения конфиденциальной информации. Установка и настройка Microsoft SQL Server проще, чем у многих других систем управления базами данных, что позволяет быстро начать работу с базой данных и сосредоточиться на разработке приложения. Microsoft SQL Server может масштабироваться в соответствии с потребностью разработчика, а также обладает большим сообществом пользователей и разработчиков и предлагает широкий спектр инструментов и ресурсов, которые помогут создавать приложения, использующие базу данных SQL. Microsoft SQL Server известен своей высокой производительностью и надежностью, обеспечивает быстрый доступ к данным и устойчивость к сбоям, что делает его отличным выбором для критически важных приложений.

# 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 библиотекарей, пользователь будет авторизирован как библиотекарь и попадает в соответствующее окно, демонстрирующее возможности системы с ограниченным функционалом, соответствующим его полномочиям.

Единственным отличием будет возможность просмотра дополнительной информации, использующий процедуры с запросами и ввод нужной информации в соответствующие поля. Например, библиотекарь может просмотреть выработку сотрудников библиотеки за определенный промежуток времени.

При авторизации как администратор, пользователь попадает в окно работы с данными системы, включающими добавление, редактирование и удаление данных во всех таблицах, а также в том числе регистрацией актов выдачи и возврата книг читателям, потому как администратор системы является также и библиотекарем.

Весь доступный функционал удобно располагается в прокручивающемся списке в левой части окна. Кроме того, все окна приложения разработаны с использованием адаптивного интерфейса, что позволяет приложению хорошо выглядеть в любом разрешении и при любых размерах окон, что может быть удобно для пользователя.

Все основные формы и виды выполнены в отдельных страницах программы с собственным функционалом и с использованием интерактивных полей ввода для взаимодействия с пользователем. Интерфейс содержит необходимые подсказки для пользователя, а также обрабатывает неправильный ввод или возможные ошибочные действия.

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

Использование WPF для разработки интерфейса значительно упрощает процесс создания настольного приложения благодаря встроенному графическому редактору. Этот редактор предоставляет возможность визуализировать создаваемую форму, и в целом облегчает работу с графическими компонентами. Более того, XAML-код, связанный с окном и являющийся partial-классом, позволяет легко вносить изменения в графический интерфейс.

Visual Studio также предлагает связанную функциональность в виде панели графических инструментов. Вместо программного создания элементов управления на C#, мы можем просто перетаскивать их на форму с панели инструментов, используя мышь. Это значительно упрощает процесс разработки интерфейса.

# 6.2 Построение главного меню и кнопок панели инструментов

Навигационная панель программы представлена пунктами таблиц, доступных для редактирования в режиме работы администратора: мероприятия, абонементы, полки, библиотечные залы, секции, жанры, произведения, книги, инвентаризация, а также перекрестные таблицы. Регистрация актов и добавление новых пользователей системы располагаются в соответствующих контролах. Также часть функционала, связанная с редактированием данных, доступна при появлении контекстного меню, открывающегося через клик правой клавиши мыши.

# 6.3 Выполнение программного кода на Microsoft Visual C#

Далее следует описание работы программы с базой данных. Часть функционала была вынесена в отдельный класс для удобства использования, а также чтобы избежать дублирования кода. Данные методы представлены в классе DatabaseOperations. Подключение к базе данных начинается с формирования строки подключения. Код класса DatabaseOperations представлен в листинге 6.1.

Листинг 6.1 – Методы для работы с базой данных

public class DatabaseOperations

{

protected string connectionString = "Data Source=ALIZZAVET\\ELIZAVETA;Initial Catalog=Библиотека;Integrated Security=True";

public SqlDataAdapter FillDataGrid(string selectCommand, out DataTable dataTable)

{

SqlDataAdapter dataAdapter = new SqlDataAdapter(selectCommand, connectionString);

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(dataAdapter);

dataTable = new DataTable();

dataAdapter.Fill(dataTable);

dataTable.PrimaryKey = new DataColumn[] { dataTable.Columns[0] };

return dataAdapter;

}

public DataRow AddRow(DataTable dataTable)

{

DataRow row = dataTable.NewRow();

string idColumnName = dataTable.Columns[0].ColumnName;

int maxId = 0;

if (dataTable.Rows.Count > 0)

{

maxId = dataTable.AsEnumerable().Max(r => r.Field<int>(idColumnName));

}

row[idColumnName] = maxId + 1;

return row;

}

public void UpdateRow(DataTable dataTable, SqlDataAdapter dataAdapter)

{

using (SqlCommandBuilder commandBuilder = new SqlCommandBuilder(dataAdapter))

{

dataAdapter.UpdateCommand = commandBuilder.GetUpdateCommand();

dataAdapter.InsertCommand = commandBuilder.GetInsertCommand();

dataAdapter.DeleteCommand = commandBuilder.GetDeleteCommand();

dataAdapter.Update(dataTable);

}

}

public void DeleteRow(DataRow row, DataTable dataTable, SqlDataAdapter dataAdapter, string currentTable, DataGrid dataGrid)

{

// Получаем ID строки, которую нужно удалить

int id = (int)row[0];

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

// Вызываем хранимую процедуру для удаления записи

string query = $"EXEC DeleteFrom{currentTable} {id}";

using (SqlCommand command = new SqlCommand(query, connection))

{

command.ExecuteNonQuery();

}

}

// Обновляем DataGrid

if (currentTable == "Sections")

{

dataAdapter = FillDataGridForDisplay($"EXEC GetSectionsData", out dataTable);

dataTable.TableName = currentTable;

dataGrid.ItemsSource = dataTable.DefaultView;

dataGrid.Columns[0].Visibility = Visibility.Collapsed;

dataGrid.Columns[2].Visibility = Visibility.Collapsed;

}

else if (currentTable == "LibraryEvents")

{

dataAdapter = FillDataGridForDisplay($"EXEC GetLibraryEventsData", out dataTable);

dataTable.TableName = currentTable;

dataGrid.ItemsSource = dataTable.DefaultView;

dataGrid.Columns[0].Visibility = Visibility.Collapsed;

dataGrid.Columns[1].Visibility = Visibility.Collapsed;

dataGrid.Columns[4].Visibility = Visibility.Collapsed;

}

else

{

dataAdapter = FillDataGridForDisplay($"SELECT \* FROM {currentTable}", out dataTable);

dataTable.TableName = currentTable;

dataGrid.ItemsSource = dataTable.DefaultView;

dataGrid.Columns[0].Visibility = Visibility.Collapsed;

}

}

public SqlDataAdapter FillDataGridFromStoredProcedure(string storedProcedureName, out DataTable dataTable)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

using (SqlCommand command = new SqlCommand(storedProcedureName, connection))

{

command.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdapter = new SqlDataAdapter(command);

SqlCommandBuilder commandBuilder = new SqlCommandBuilder(dataAdapter);

dataTable = new DataTable();

dataAdapter.Fill(dataTable);

return dataAdapter;

}

}

}

public SqlDataAdapter FillDataGridForDisplay(string selectCommand, out DataTable dataTable)

{

return FillDataGrid(selectCommand, out dataTable);

}

public SqlDataAdapter FillDataGridForEdit(string selectCommand, out DataTable dataTable)

{

return FillDataGrid(selectCommand, out dataTable);

}

}

Работа с данными (редактирование и добавление новых записей) происходит в специальном окне, которое вызывается после нажатия на соответствующие пункты контекстного меню. Благодаря тому, что окно универсальное, не потребовалось создавать огромное количество окон-копий. Пользовательские контролы для ввода информации генерируются исходя из типа принимаемых аргументов из таблиц. Для того, чтобы реализовать подсказки для пользователя, реализован функционал, позволяющий при наведении на контрол отобразить всплывающую подсказку, содержащую в себе имя поля, которое нужно ввести. Таким образом удалось добиться интуитивно понятного универсального интерфейса окна. Пользовательские контролы представлены текстовыми полями и комбо-боксами, которые облегчают выбор данных. Код окна представлен в листинге 6.2.

Листинг 6.2 – Универсальное окно для работы с данными

public partial class EditWindow : Window

{

private DataRow row;

private Dictionary<string, Control> controls = new Dictionary<string, Control>();

private ComboBoxHelper comboBoxHelper = new ComboBoxHelper();

public EditWindow(DataRow row, bool isNewRecord)

{

InitializeComponent();

this.row = row;

TextBox textBox;

foreach (DataColumn column in row.Table.Columns)

{

if (column.DataType == typeof(DateTime))

{

DatePicker datePicker = new DatePicker();

if (row[column] != DBNull.Value)

{

datePicker.SelectedDate = (DateTime)row[column];

}

stackPanel.Children.Insert(stackPanel.Children.Count - 2, datePicker);

controls.Add(column.ColumnName, datePicker);

}

else

{

switch (column.ColumnName)

{

case "LibrarianID":

if (row.Table.TableName == "Librarians")

{

textBox = CreateTextBox(row, column);

}

else

{

ComboBox comboBoxLibrarians = CreateComboBox("comboBoxLibrarians", row, column);

comboBoxHelper.FillLibrariansComboBox(comboBoxLibrarians);

}

break;

case "ActionType":

ComboBox comboBoxActionType = CreateComboBox("comboBoxActionType", row, column);

comboBoxHelper.FillActionTypeComboBox(comboBoxActionType);

break;

default:

textBox = CreateTextBox(row, column);

break;

}

}

}

}

private ComboBox CreateComboBox(string name, DataRow row, DataColumn column)

{

ComboBox comboBox = new ComboBox

{

Name = name,

HorizontalAlignment = HorizontalAlignment.Left,

Margin = new Thickness(10, 0, 0, 10),

VerticalAlignment = VerticalAlignment.Top,

Width = 200

};

stackPanel.Children.Insert(stackPanel.Children.Count - 2, comboBox);

controls.Add(column.ColumnName, comboBox);

if (int.TryParse(row[column].ToString(), out int id))

{

comboBox.SelectedValue = id;

}

else

{

comboBox.SelectedValue = null;

}

return comboBox;

}

private TextBox CreateTextBox(DataRow row, DataColumn column)

{

TextBox textBox = new TextBox { Text = row[column].ToString() };

textBox.ToolTip = column.ColumnName;

if (column == row.Table.Columns[0])

{

textBox.IsEnabled = false;

}

stackPanel.Children.Insert(stackPanel.Children.Count - 2, textBox);

controls.Add(column.ColumnName, textBox);

return textBox;

}

private void btnOK\_Click(object sender, RoutedEventArgs e)

{

try

{

foreach (DataColumn column in row.Table.Columns)

{

if (column.DataType == typeof(DateTime))

{

DatePicker datePicker = (DatePicker)controls[column.ColumnName];

if (datePicker.SelectedDate.HasValue)

{

row[column] = datePicker.SelectedDate.Value;

}

else

{

MessageBox.Show($"Поле {column.ColumnName} не может быть пустым.");

return;

}

}

else

{

if (controls[column.ColumnName] is TextBox textBox)

{

if (string.IsNullOrWhiteSpace(textBox.Text))

{

MessageBox.Show($"Поле {column.ColumnName} не может быть пустым.");

return;

}

row[column] = textBox.Text;

}

else if (controls[column.ColumnName] is ComboBox comboBox)

{

if (comboBox.SelectedValue != null)

{

KeyValuePair<int, string> selectedPair = (KeyValuePair<int, string>)comboBox.SelectedValue;

row[column] = selectedPair.Key;

}

else

{

MessageBox.Show($"Поле {column.ColumnName} не может быть пустым.");

return;

}

}

}

}

this.DialogResult = true;

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка: {ex.Message} Пожалуйста, заполните все данные.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void btnCancel\_Click(object sender, RoutedEventArgs e)

{

DialogResult = false;

}

}

Одной из главных особенностей функциональности данной системы является регистрация актов выдачи и возврата. Окно, на которое попадает пользователь при выборе соответствующей функциональности, представляет из себя окно для отображения уже существующих записей в базе данных в специальном контроле: data grid. А также реализует в себе возможность вызова нужных пунктов работы с актами из контекстного меню по нажатию правой клавиши мыши. Для регистрации актов вызывается отдельно оформленное окно, также как и для добавления или редактирования новых пользователей системы. Потому как заполнение данных таблиц несколько отличается от редактирования данных простых или перекрестных таблиц, также содержащихся в базе данных. Обработка введенных данных и реализация отображения актов выдачи и возврата представлена в классе ActWindow. Для отображения данных использовались процедуры запросов в SQL-коде. Для того, чтобы скрыть отображение лишней информации для пользователя, используется команда dataGrid.Columns[0].Visibility = Visibility.Collapsed;, где в квадратных скобках указывается номер столбца, который мы хотим скрыть при отображении данных в data grid. Данные процедуры вызываются в этом окне. Код окна представлен в листинге 6.3.

Листинг 6.2 – Универсальное окно для работы с данными

public partial class ActWindow : Window

{

protected string connectionString = "Data Source=ALIZZAVET\\ELIZAVETA;Initial Catalog=Библиотека;Integrated Security=True";

public ActWindow()

{

InitializeComponent();

}

private void LoadData\_Click(object sender, RoutedEventArgs e)

{

LoadDataGrid();

}

private void Add\_Click(object sender, RoutedEventArgs e)

{

EditActWindow editActWindow = new EditActWindow();

editActWindow.ShowDialog();

LoadDataGrid();

}

private void Edit\_Click(object sender, RoutedEventArgs e)

{

if (dataGrid.SelectedItem is DataRowView row)

{

int actId = (int)row["ActID"];

EditActWindow editActWindow = new EditActWindow(actId);

editActWindow.ShowDialog();

LoadDataGrid();

}

}

private void Delete\_Click(object sender, RoutedEventArgs e)

{

if (dataGrid.SelectedItem is DataRowView row)

{

int actId = (int)row["ActID"];

MessageBoxResult result = MessageBox.Show("Are you sure you want to delete this item?", "Confirmation", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand("DeleteAct", connection);

command.CommandType = CommandType.StoredProcedure;

command.Parameters.AddWithValue("@ActID", actId);

command.ExecuteNonQuery();

}

LoadDataGrid();

}

}

}

private void LoadDataGrid()

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

connection.Open();

SqlCommand command = new SqlCommand("GetActsAndActsBooksData", connection);

command.CommandType = CommandType.StoredProcedure;

SqlDataAdapter dataAdapter = new SqlDataAdapter(command);

DataTable dataTable = new DataTable("ActsAndActsBooks");

dataAdapter.Fill(dataTable);

dataGrid.ItemsSource = dataTable.DefaultView;

dataGrid.Columns[0].Visibility = Visibility.Collapsed;

dataGrid.Columns[1].Visibility = Visibility.Collapsed;

dataGrid.Columns[3].Visibility = Visibility.Collapsed;

dataGrid.Columns[8].Visibility = Visibility.Collapsed;

dataGrid.Columns[7].Visibility = Visibility.Collapsed;

}

}

private void BackToAdminWindow\_Click(object sender, RoutedEventArgs e)

{

if(MainWindow.IsAdmin == false)

{

LibraryWindow libraryWindow = new LibraryWindow();

libraryWindow.Show();

Close();

}

else

{

AdminWindow adminWindow = new AdminWindow();

adminWindow.Show();

this.Close();

}

}

}

# 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**

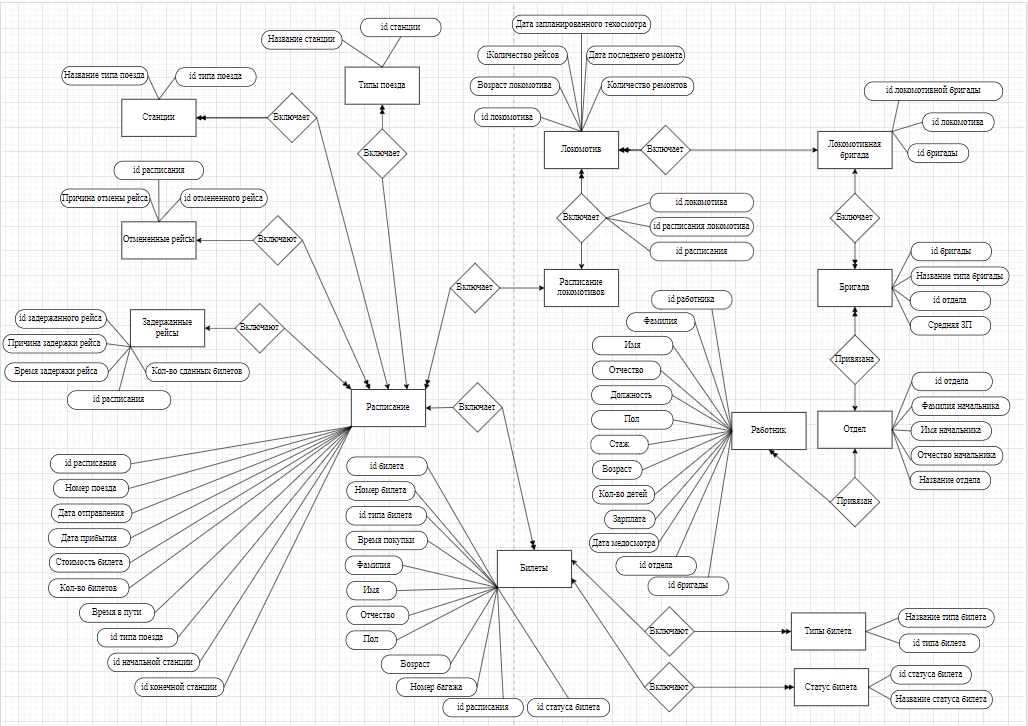


Рисунок А.1 – Концептуальная модель проектируемой базы данных

# ПРИЛОЖЕНИЕ Б

(обязательное)

**Схема реляционной базы данных**

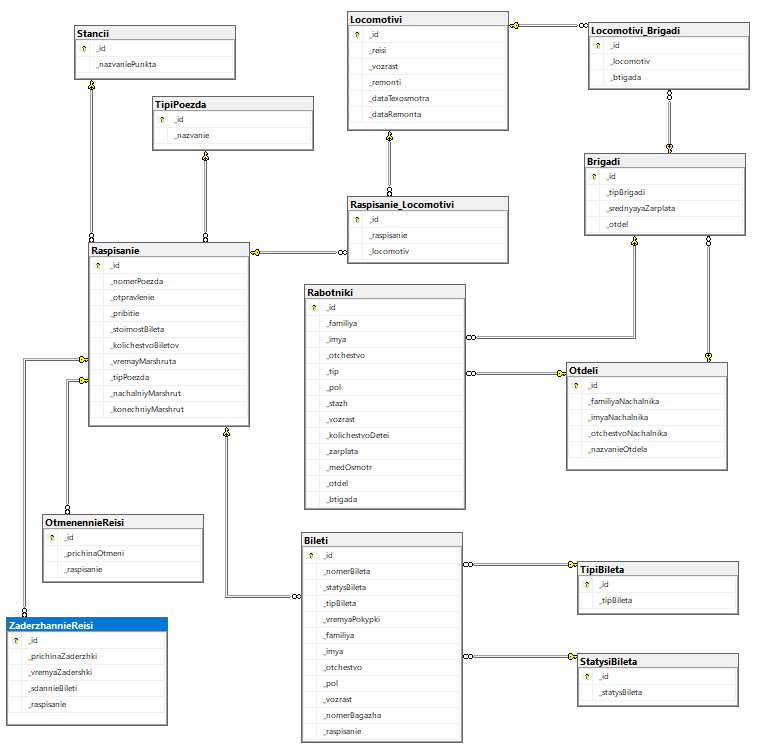


Рисунок Б.1 – Схема реляционной базы данных

# 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.

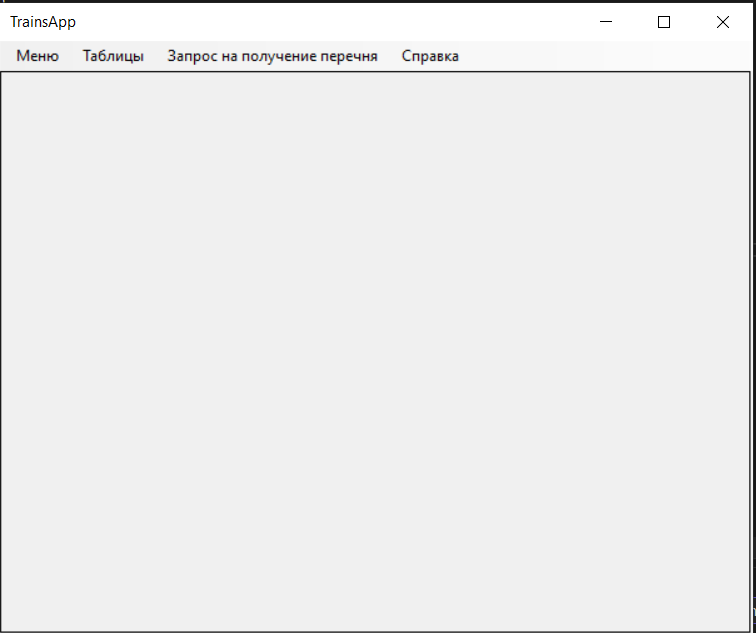


Рисунок Г.1 – Главное окно приложения

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

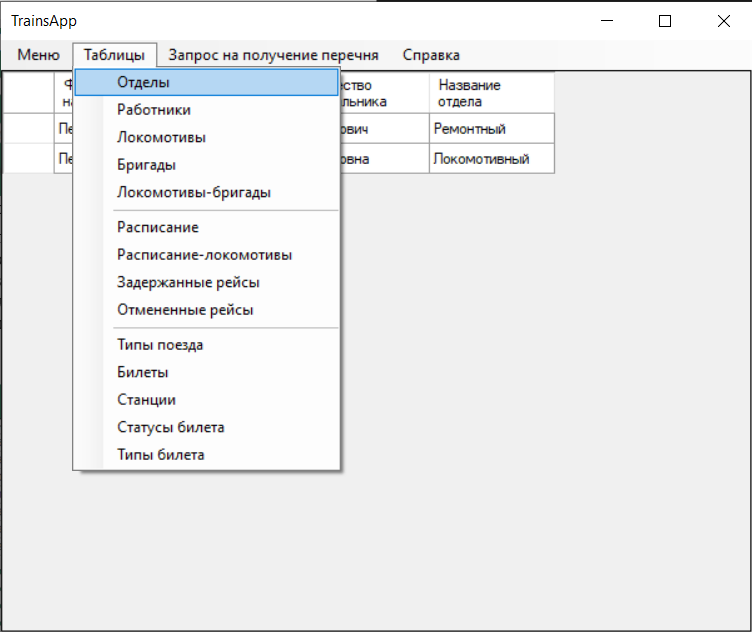


Рисунок Г.2 – Выбор пункта «Отделы»

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».

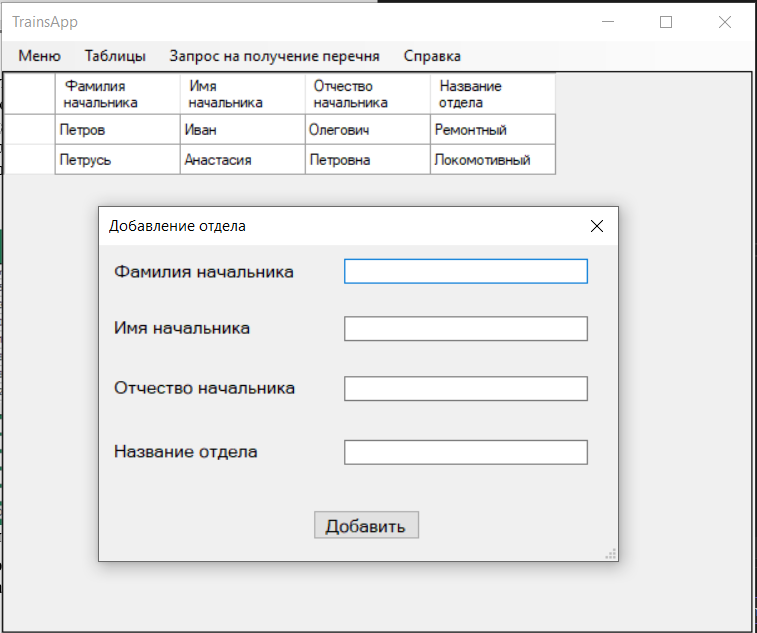


Рисунок Г.3 – Окно добавления записи

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.

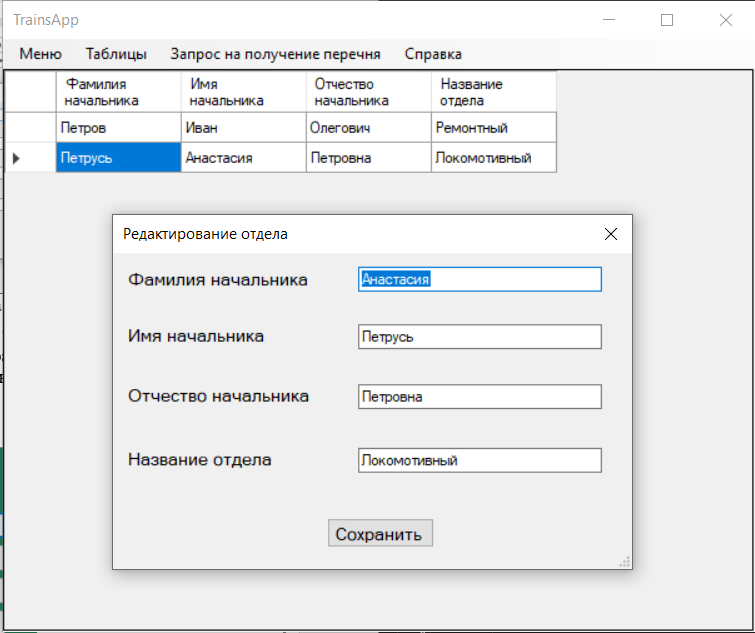


Рисунок Г.4 – Окно редактирования записи

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.

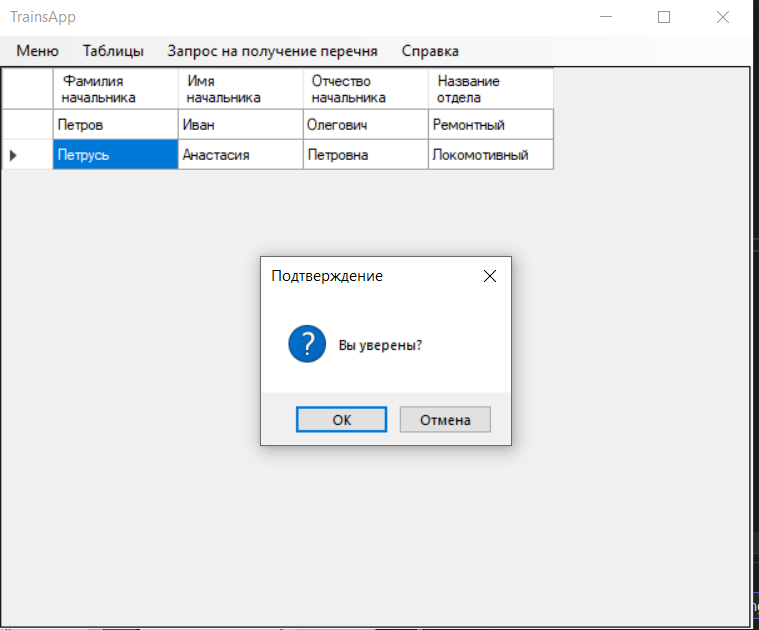


Рисунок Г.5 – Окно подтверждения удаления

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»

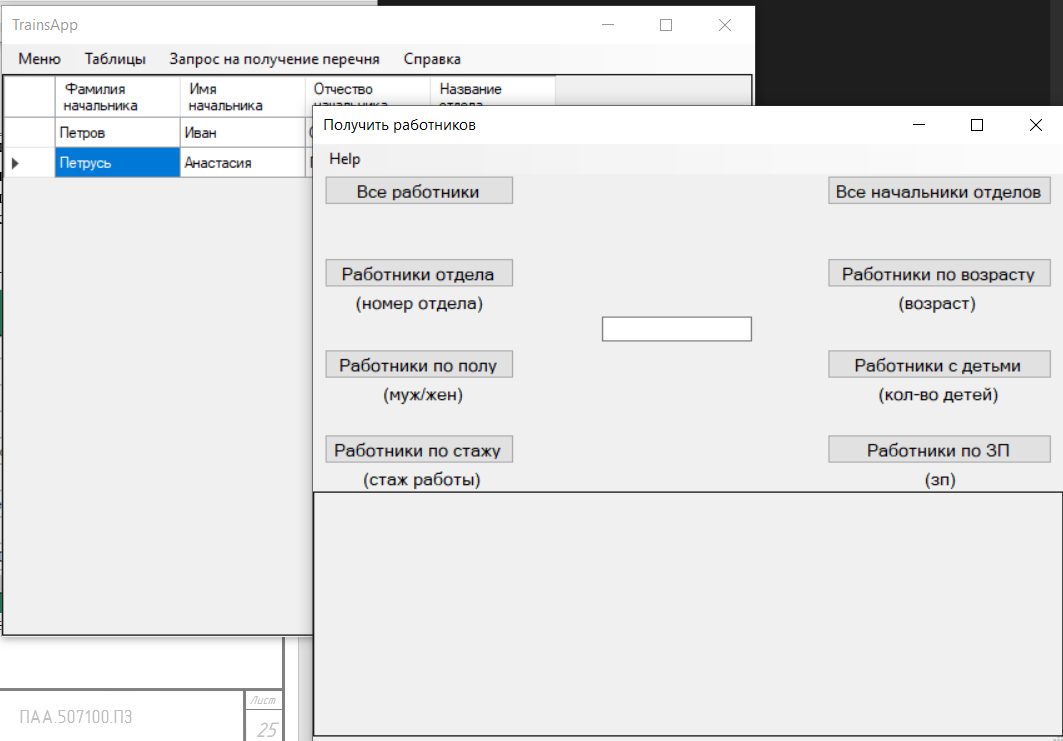


Рисунок Г.6 – Окно для запросов к таблице «Работники»

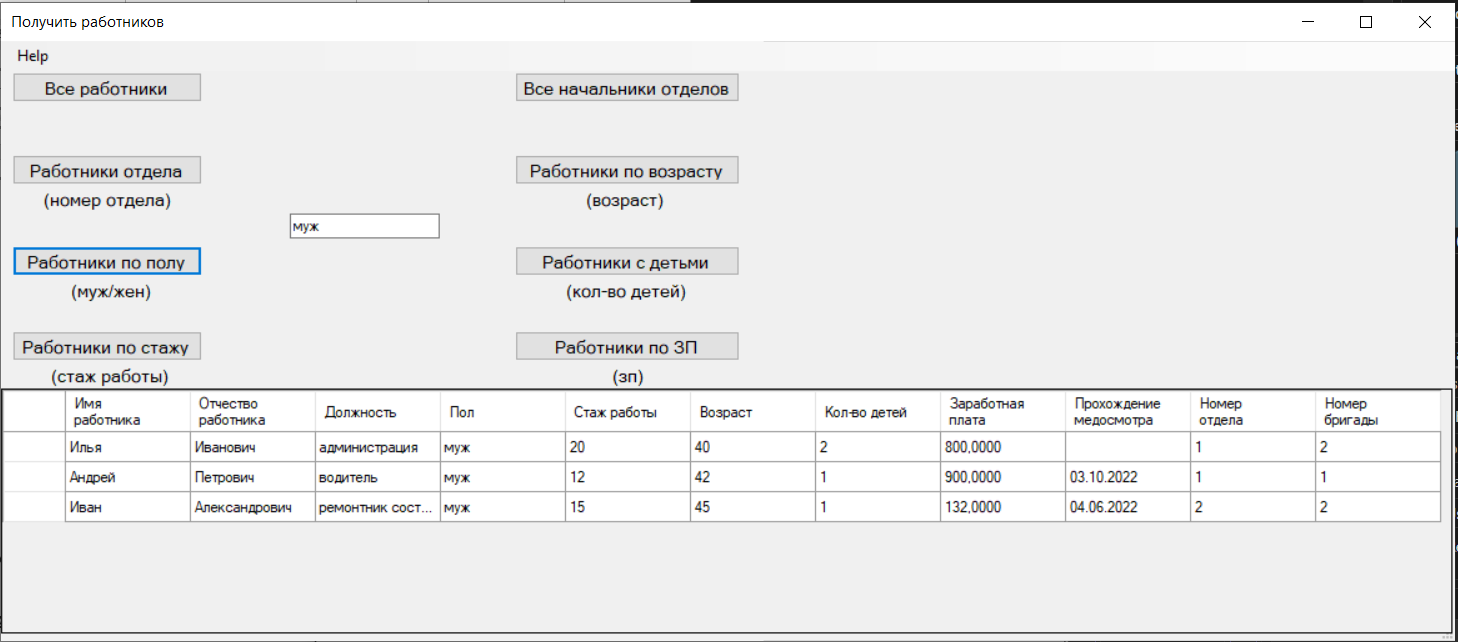


Рисунок Г.7 – Результат выборки по полу «Пол»

Любые изменения будут сохранятся в используемой базе данных.