

МИНИСТЕРСТВО ОБРАЗОВАНИЯ, НАУКИ И МОЛОДЕЖИ РЕСПУБЛИКИ КРЫМ

ГБОУВО РК «КРЫМСКИЙ ИНЖЕНЕРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ ФЕВЗИ ЯКУБОВА»

Факультет экономики, менеджмента и информационных технологий

Кафедра прикладной информатики

Направление подготовки 09.03.03 Прикладная информатика

Профиль «Прикладная информатика в информационной сфере»

***КУРСОВОЙ ПРОЕКТ***

по дисциплине «Базы данных»

НА ТЕМУ

**Разработка базы данных для продажи и бронирования ж/д билетов через Интернет**

Студента II курса

группы И-1-21

очной формы обучения

Минзатов Э.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Научный руководитель:

преп. Танишева С.С.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

Оценка после защиты

Дата

(Подпись научного руководителя)

Симферополь – 2023

**Аннотация**

**Минзатов Э.С. Разработка базы данных для продажи и бронирования ж/д билетов через Интернет**

В курсовом проекте представлена разработка базы данных для продажи и бронирования железнодорожных билетов через Интернет. Были созданы объекты, таблицы, запросы и формы для обеспечения полноценной работы базы данных. Была реализована система управления базами данных MySQL и интерфейс приложения, включая главное меню. Основное внимание уделено программной оболочке и системе взаимодействия с базой данных, чтобы обеспечить быстрый и удобный доступ к информации о билетах и их продаже.

**Ключевые слова:** ж/д билеты, Интернет, продажа, разработка.

**Abstract**

**Minzatov E. S. Development of a database for the sale and booking of railway tickets via the Internet**

The course project presents the development of a database for the sale and booking of railway tickets via the Internet. Objects, tables, queries and forms have been created to ensure the full operation of the database. The MySQL database management system and the application interface, including the main menu, were implemented. The main attention is paid to the software shell and the system of interaction with the database in order to provide quick and convenient access to information about tickets and their sale.

**Keywords**: railway tickets, Internet, realization, development.

Содержание

[**ВВЕДЕНИЕ 4**](#_Toc133938422)

[**ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ В ОБЛАСТИ РАЗРАБОТКИ БАЗ ДАННЫХ 6**](#_Toc133938423)

[**1.1. Основная терминология теории баз данных 6**](#_Toc133938424)

[**1.2. Анализ предметной области 8**](#_Toc133938425)

[**1.3. Обзор систем управления базами данных 9**](#_Toc133938426)

[**1.4. Инфологическое проектирование 12**](#_Toc133938427)

[**1.5. Логическое проектирование 14**](#_Toc133938428)

[**Выводы к первой главе 18**](#_Toc133938429)

[**ГЛАВА 2. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ В ОБЛАСТИ РАЗРАБОТКИ БАЗ ДАННЫХ 20**](#_Toc133938430)

[**2.1. Разработка проекта и базы данных 20**](#_Toc133938431)

[**2.2. Создание объектов базы данных 21**](#_Toc133938432)

[**2.2.1. Создание таблиц, заполнение их тестовой информацией 21**](#_Toc133938433)

[**2.2.2. Создание запросов 26**](#_Toc133938434)

[**2.3. Создание главного меню приложения 29**](#_Toc133938435)

[**2.4. Разработка системы оперативной справки 30**](#_Toc133938436)

[**Выводы по главе 2 32**](#_Toc133938437)

[**ЗАКЛЮЧЕНИЕ 33**](#_Toc133938438)

# ВВЕДЕНИЕ

**Актуальность темы.** Актуальность темы исследования в данной курсовой работе связана с тем, что в силу развития информационных технологий и интернет-сервисов возрастает спрос на удобные и эффективные способы продажи и бронирования ж/д билетов онлайн. Железнодорожный транспорт является одним из самых доступных и экологичных видов транспорта в России. Он обеспечивает перевозку большого количества пассажиров и грузов на различные расстояния. Однако железнодорожный транспорт сталкивается с рядом проблем, таких как низкая скорость работы, сложность интерфейса, ненадежность хранения данных и защиты от несанкционированного доступа существующих баз данных для продажи и бронирования ж/д билетов. Поэтому необходимо разрабатывать новые подходы к созданию и управлению базами данных. Создание новой базы данных, хранящей информацию о доступных ж/д рейсах, ценах на билеты, датах отправления и прибытия, а также программной оболочки, позволяющей осуществлять управление содержимым базы данных, является актуальной и востребованной задачей. Приложение базы данных позволит быстро модифицировать необходимую информацию, минимизировать избыточность информации, уменьшить вероятность ввода ошибочных данных. База данных упростит реализацию сложных запросов, повысит эффективность использования информационной технологии. Минимизация избыточности и возможность быстрой модификации позволит поддерживать данные на одинаковом уровне актуальности. Кроссплатформенное приложение позволить организовать управление базой данных на всех популярных компьютерных операционных системах.

**Целью курсового проекта** является разработка базы данных для продажи и бронирования ж/д билетов». Для достижения цели были поставлены следующие **задачи**:

* рассмотреть основные понятия теории баз данных;
* проанализировать предметную область;
* выполнить инфологическое, логическое проектирование;
* осуществить обзор современных систем управления базами данных, выбрать наиболее целесообразную для реализации базы данных к заданной предметной области, проанализировав преимущества и недостатки.

**Объектом курсовой работы** является разработка базы данных с использованием конкретной СУБД.

**Предметом** **курсовой работы** является разработка базы данных для продажи и бронирования ж/д билетов через Интернет.

**Структура курсового проекта.** Текстовая часть состоит из введения, двух глав основной части, заключения и списка использованных источников. Общий объем курсового проекта составляет 32 страницы печатного текста, 21 рисунка.

# ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ В ОБЛАСТИ РАЗРАБОТКИ БАЗ ДАННЫХ

## 1.1. Основная терминология теории баз данных

База данных — это организованная структура, предназначенная для хранения, изменения и обработки взаимосвязанной информации. База данных может быть использована для разных целей, например, для учета товаров в магазине, для хранения контактов в телефоне или для создания сайтов в интернете [1]. Для управления базами данных используются системы управления базами данных (СУБД). СУБД появились из-за необходимости эффективно хранить и обрабатывать большие объемы данных в различных приложениях. Первые СУБД были навигационными или сетевыми, то есть данные представлялись в виде записей и связей между ними. Навигационные СУБД были сложны в использовании и не гарантировали целостность данных. Поэтому в 1970 году Эдгар Кодд предложил реляционную модель данных, которая основывалась на математической теории отношений и позволяла работать с данными в виде таблиц. Реляционные СУБД стали доминирующим классом систем баз данных и до сих пор широко используются [2].

СУБД — это комплекс программных и лингвистических средств, позволяющих создать базу данных и манипулировать данными. С помощью СУБД можно вставлять, обновлять, удалять и выбирать данные в базе данных. СУБД также обеспечивает защиту и целостность данных, а также поддержку разных моделей данных [3].

БД должна хранить все основные данные, связанные с продажей и бронированием ж/д билетов через Интернет. При этом не должно быть повторения данных, то есть информация о каждом поезде, станции или пассажире должна находиться только в одной таблице. Для построения БД используется определенная модель данных. Модель данных – это набор правил и способов организации и обработки данных. Существуют разные типы моделей данных в зависимости от того, как устанавливаются связи между данными: сетевая, иерархическая, объектно-ориентированная и реляционная. В реляционной модели данные представлены таблицами, где каждая строка содержит свой набор атрибутов. В этой модели связи между данными устанавливаются с помощью общих атрибутов или ключей. На верхнем уровне в этой модели находятся таблицы без внешних ключей, на следующем уровне – таблицы, ссылающиеся на эти таблицы, затем таблицы, ссылающиеся на таблицы предыдущего уровня и т.д. При этом каждая строка может иметь несколько потомков и предков.

Чтобы найти данные в иерархической системе, нужно идти от корня к ветвям дерева до нужного уровня. Ссылки помогают переходить между записями. Иерархическая модель хорошо подходит для отображения реальных иерархий, запросы к таким базам данных быстро обрабатываются. Но не всегда удобно начинать с корня каждый раз, а других путей по базе нет.

Одним из преимуществ использования иерархической и сетевой моделей данных является повышение скорости доступа к информации в базе данных. Однако для эффективной работы таких моделей необходимы значительные ресурсы как дисковой, так и основной памяти компьютера, поскольку каждый элемент данных должен содержать ссылки на другие элементы. В случае недостаточности основной памяти скорость обработки данных может снизиться. Кроме того, для таких моделей характерна сложность реализации системы управления базами данных, которая должна обеспечивать целостность и безопасность данных.

Реляционные базы данных — это базы данных, основанные на реляционной модели данных, которая представляет данные в виде таблиц с записями и атрибутами. Концепция реляционной модели данных была сформулирована в 1969-1970 годах Эдгаром Коддом, который работал в IBM и искал более эффективный способ работы с данными [4]. Он опубликовал свою знаменитую статью «A Relational Model of Data for Large Shared Data Banks» в 1970 году, в которой изложил основные принципы реляционной модели, такие как нормализация, целостность и операции над отношениями.

После публикации статьи Кодда был запущен ряд проектов по созданию реляционных СУБД, реализующих такую модель. Одним из первых был проект System R в IBM Research в 1974 году, который привел к разработке языка запросов SQL (Structured Query Language).

С тех пор реляционные базы данных стали доминирующей парадигмой управления данными и используются для хранения и обработки большого количества информации в различных областях, таких как бизнес, наука, образование и т.д.

Особенности реляционных баз данных можно сформулировать следующим образом:

* Целостность сведений. Информация, которая помещается в реляционную базу данных, отличается повышенной точностью, однообразностью и максимальной полнотой [5].
* Нормализация — это процесс разбиения таблиц на более мелкие и простые отношения, чтобы избежать дублирования и аномалий данных. Нормализация помогает повысить целостность и эффективность реляционных баз данных.
* Реляционная алгебра. Реляционная алгебра — это набор операций над отношениями (таблицами), которые позволяют извлекать, модифицировать и комбинировать данные из реляционной базы данных. Примерами таких операций являются выборка (selection), проекция (projection), соединение (join), объединение (union) и т.д.
* Гибкость. Реляционные базы данных демонстрируют чрезвычайную гибкость в построении различных приложений и работы с большими объемами данных. Они также позволяют легко изменять структуру и свойства таблиц без потери информации.

## 1.2. Анализ предметной области

Разработка базы данных для продажи и бронирования ж/д билетов через интернет является важной задачей, которая в настоящее время является актуальной для многих компаний в сфере транспортных услуг. Данный проект имеет целью упростить процесс продажи билетов для пассажиров и оптимизировать работу транспортной компании.

При анализе предметной области следует учитывать несколько факторов. Прежде всего, необходимо учесть особенности работы транспортной компании, ее масштаб и географию. Также следует учесть специфику работы с клиентами, в том числе организацию продажи и бронирования билетов, оформление заказов, обработку платежей и т.д.

Одним из главных аспектов является безопасность и конфиденциальность персональных данных клиентов, которые будут храниться в базе данных. Для обеспечения безопасности следует использовать соответствующие технологии и методы защиты данных.

Кроме того, при разработке базы данных следует учесть возможность интеграции с другими системами, такими как системы электронного платежа, системы управления бронированием, системы управления контактными центрами и т.д.

В целом, разработка базы данных для продажи и бронирования ж/д билетов через интернет – это сложный и многогранный проект, требующий тщательного анализа и проектирования. Но благодаря современным технологиям и методам, его реализация может принести значительную пользу и упростить жизнь многих пассажиров.

## 1.3. Обзор систем управления базами данных

Правильный выбор СУБД может значительно повлиять на эффективность и производительность работы приложения, а также на безопасность и защиту данных. Поэтому необходимо провести тщательный анализ требований и возможностей перед выбором СУБД.

Основные критерии по выбору системы оправления базами данных:

1. Производительность - время выполнения запросов и операций, скорость работы на больших объемах данных, уровень нагрузки, возможности кэширования.
2. Функциональность - набор функций, которые предоставляет СУБД, такие как поддержка транзакций, хранимых процедур, триггеров, полнотекстового поиска, поддержка географических данных, масштабируемость и др.
3. Надежность и безопасность - возможности резервного копирования, механизмы репликации и восстановления данных, уровень защиты данных, включая доступ к базе данных, шифрование, авторизацию и аудит.
4. Управление и мониторинг - наличие инструментов для управления и мониторинга базы данных, удобство использования этих инструментов, возможность масштабирования и настройки параметров.
5. Стоимость - стоимость лицензий и поддержки, затраты на оборудование и ресурсы для управления базой данных.
6. Сообщество и поддержка - наличие сообщества разработчиков и пользователей, доступность документации и руководств, наличие онлайн-форумов, возможность получения поддержки от разработчиков СУБД и других сторонних организаций.

Были рассмотрены следующие СУБД: MySQL, PostgreSQL, Microsoft SQL Server, Oracle, MongoDB.

Все выбранные СУБД подходят для анализа и сравнения, т.к. реализуют реляционную модель:

1. Производительность:

* MySQL и PostgreSQL показывают хорошую производительность на тяжелых нагрузках, включая крупномасштабные операции с базой данных.
* MongoDB и Microsoft SQL Server предназначены для работы с нереляционными и реляционными данными соответственно и могут быть медленными при выполнении сложных запросов на больших объемах данных.
* Oracle имеет высокую производительность и широкий набор функций для работы с большими объемами данных.

1. Функциональность:

* PostgreSQL и Oracle имеют широкий набор функций, включая поддержку триггеров, процедур, пользовательских типов данных, полнотекстовый поиск и географические данные.
* MySQL и Microsoft SQL Server имеют основные функции для работы с реляционными данными, но могут быть ограничены в поддержке сложных запросов и операций.
* MongoDB предназначен для работы с нереляционными данными и имеет функции для хранения и обработки документов.

1. Надежность и безопасность:

* Все СУБД имеют механизмы резервного копирования и восстановления данных, а также механизмы безопасности, такие как шифрование данных и аутентификация пользователей.
* Oracle и Microsoft SQL Server имеют функции репликации, которые обеспечивают высокую доступность и масштабируемость.

1. Управление и мониторинг:

* PostgreSQL и Oracle имеют удобные средства управления и мониторинга базы данных, включая инструменты для настройки параметров и мониторинга производительности.
* MySQL и MongoDB имеют менее развитые инструменты управления, но включают сторонние инструменты от сообщества разработчиков.
* Microsoft SQL Server имеет удобный интерфейс управления и мониторинга базы данных в виде SQL Server Management Studio.

1. Стоимость:

* MySQL и PostgreSQL являются бесплатными и имеют открытый исходный код.
* Oracle и Microsoft SQL Server имеют платные лицензии и поддержку.
* MongoDB имеет разные уровни поддержки, включая бесплатную версию и платные опции для крупных проектов.

1. Сообщество и поддержка:

* MySQL, PostgreSQL, Oracle, Microsoft SQL Server имеют большое сообщество пользователей и разработчиков, которые создают инструменты, обновления и исправления.
* MongoDB имеет активное сообщество пользователей и разработчиков, которые создают плагины и обновления для базы данных [6].

В целом, каждая СУБД имеет свои сильные и слабые стороны, и выбор зависит от конкретных потребностей проекта.

MySQL и PostgreSQL подходят для крупномасштабных операций с реляционными данными, MongoDB подходит для работы с нереляционными данными, а Oracle и Microsoft SQL Server имеют большой набор функций и могут быть использованы для обработки больших объемов данных в корпоративной среде. Кроме того, MySQL и PostgreSQL являются бесплатными и имеют открытый исходный код, тогда как Oracle и Microsoft SQL Server имеют платные лицензии и поддержку.

MySQL также поддерживает множество функций, таких как репликация, что позволяет автоматически копировать данные на другие серверы, и шифрование данных, что обеспечивает безопасность данных при передаче и хранении [7].

## 1.4. Инфологическое проектирование

Инфологическое проектирование баз данных — это процесс разработки концептуальной модели данных, которая описывает данные, используемые в организации или бизнесе, и их отношения между собой. Это важный этап в разработке баз данных, который помогает понять, каким образом данные используются в организации, как они связаны друг с другом и какие правила должны быть определены для их использования.

Основной целью ИПБД является определение структуры и организации данных, которые будут использоваться в базе данных, а также определение правил и ограничений на использование этих данных. Это включает в себя определение таблиц, полей, связей между таблицами, индексов, ограничений целостности и т.д.

Инфологическое проектирование баз данных является одним из этапов процесса разработки баз данных, который следует за созданием концептуальной модели данных и предшествует физическому проектированию базы данных.

Конструктивные элементы — это основные компоненты, из которых состоят инфологические модели данных. Они представляют из себя сущности, атрибуты, отношения, ассоциации, иерархии, абстракции и ограничения.

Конструктивные элементы для модели базы данных для продажи и бронирования ж/д билетов через Интернет могут выглядеть следующим образом:

1. Сущности:

* Клиенты (Customers)
* Ж/д рейсы (Routes)
* Станции (Stations)
* Билеты (Tickets)

1. Атрибуты:

* Для клиентов: имя, фамилия, адрес, номер телефона, электронная почта.
* Для ж/д рейсов: номер поезда, дата отправления, время отправления, станция отправления, станция прибытия.
* Для станций: название, адрес, город.
* Для билетов: номер билета, дата покупки, дата отправления, время отправления, станция отправления, станция прибытия, место, цена.

1. Отношения:

* Каждый ж/д рейс имеет множество билетов (One-to-Many Relationship: Routes to Tickets).
* Каждый клиент может иметь несколько билетов (One-to-Many Relationship: Customers to Tickets).
* Каждый ж/д рейс имеет одну станцию отправления (One-to-One Relationship: Routes to Stations).
* Каждый ж/д рейс имеет одну станцию прибытия (One-to-One Relationship: Routes to Stations).

1. Ассоциации:

* Каждый билет связан с конкретным ж/д рейсом (Association: Tickets to Routes).
* Каждый билет связан с конкретным клиентом (Association: Tickets to Customers).

1. Иерархии:

* Ж/д рейсы могут быть упорядочены по городам, которые они связывают, например: Москва - Санкт-Петербург, Москва - Нижний Новгород (Hierarchy: Routes).

1. Абстракции:

* Дата и время отправления и прибытия могут быть описаны с помощью абстракции «время» (Abstraction: Time).
* Цена может быть описана с помощью абстракции «валюта» (Abstraction: Currency).

1. Ограничения:

* Нельзя продать билет на ж/д рейс, если на него не осталось свободных мест (Constraint: Tickets Sold Out).
* Дата отправления должна быть больше текущей даты (Constraint: Departure Date).

## 1.5. Логическое проектирование

Логическое проектирование — это процесс создания модели данных для базы данных, определяющий структуру базы данных, ее компоненты, их связи и правила, которые определяют, как данные должны храниться и связываться друг с другом.

Основной целью логического проектирования является создание оптимальной структуры базы данных, обеспечивающей эффективное хранение данных, быстрый доступ к ним и сохранение целостности данных. Логическое проектирование является важным этапом в процессе создания базы данных и предшествует физическому проектированию, которое определяет, как данные будут храниться на диске.

В результате логического проектирования баз данных создается концептуальная модель данных, которая описывает структуру и связи данных в базе данных без учета конкретных технологий и платформ, на которых будет реализована база данных. Эта модель служит основой для физического проектирования базы данных, где определяется, как эта модель будет реализована на конкретной платформе [8].

Каскадное обновление и каскадное удаление являются механизмами, которые определяют, как база данных будет обрабатывать изменения в данных и поддерживать целостность базы данных при внесении изменений. Эти механизмы являются частью логического проектирования базы данных и используются для определения ограничений целостности и взаимосвязей между таблицами.

При проектировании базы данных важно учитывать не только структуру таблиц и связи между ними, но и механизмы обработки изменений в данных, чтобы гарантировать целостность и надежность базы данных [9].

При логическом проектировании базы данных для продажи и бронирования ж/д билетов через Интернет следует учесть следующие аспекты:

1. Нормализация данных — это процесс разбиения таблицы на более мелкие и связанные между собой таблицы. Нормализация позволяет устранить избыточность и неоднозначность данных, что повышает эффективность хранения и обработки данных.
2. Индексация — это процесс создания индексов для быстрого доступа к данным. Индексы позволяют ускорить выполнение запросов к базе данных и повысить ее производительность.
3. Разграничение доступа — это ограничение доступа к данным на уровне пользователя или роли. Это позволяет управлять доступом к данным и обеспечить безопасность базы данных.
4. Резервное копирование — это процесс создания резервной копии базы данных. Резервное копирование позволяет защитить данные от случайного или намеренного их удаления, а также от сбоев в оборудовании.
5. Обеспечение целостности данных — это процесс обеспечения точности и согласованности данных в базе данных. Это достигается с помощью ограничений и проверок, которые предотвращают добавление неверных данных в базу данных.
6. Оптимизация запросов — это процесс оптимизации запросов к базе данных с целью ускорения их выполнения. Оптимизация запросов позволяет повысить производительность базы данных и уменьшить нагрузку на систему.
7. Масштабируемость — это способность базы данных масштабироваться в соответствии с растущим объемом данных и нагрузкой на систему. Масштабируемость позволяет обеспечить эффективность и доступность базы данных в будущем.

В данной схеме есть одна явная проблема, которые мешает ее привести к 3 нормальной форме.

В таблицах customers и users есть поле boughtTicketId и userboughtTicketsId соответственно, которое ссылается на таблицу routes. Однако, у каждого билета может быть только один покупатель и один пользователь, а в таблице routes может быть несколько билетов. Таким образом, в таблицах customers и users есть множественная связь с таблицей routes. Это нарушает требования 2 нормальной формы.

Чтобы исправить это можно создать дополнительную таблицу, которая будет связывать таблицы customers, routes и users и иметь первичный ключ из двух полей: customer\_id и route\_id. Таким образом, в таблицах customers и users нужно удалить поля boughtTicketId и userboughtTicketsId, а вместо них использовать ссылку на созданную дополнительную таблицу. Это позволит избежать множественной связи и соответственно нарушения 2 нормальной формы.

Для третьей нормальной формы (3НФ) необходимо следовать следующим правилам:

1. Каждый неключевой атрибут должен зависеть только от первичного ключа.
2. Не должно быть транзитивных зависимостей.

Исходя из этого, можно проанализировать каждую таблицу в схеме БД:

* Таблица customers уже находится в 3НФ, так как имеет только первичный ключ и неключевой атрибут email зависит только от idCustomers.
* Таблица stations также находится в 3НФ, так как имеет только первичный ключ и неключевые атрибуты name и city зависят только от idStations.
* Таблица routes находится в 3НФ, так как departure\_station и arrival\_station зависят только от первичного ключа idRoutes
* route\_id и customer\_id зависят только от первичного ключа idTickets. Нет транзитивных зависимостей, поэтому таблица уже находится в 3НФ.
* Таблица users уже находится в 3НФ, так как имеет только первичный ключ и неключевые атрибуты userEmail, userPassword и isAdmin зависят только от idUsers.
* Таблица user\_bought\_tickets имеет следующие зависимости:
  + - * + user\_id зависит только от первичного ключа idUsers.
        + route\_id и ticket\_id зависят только от первичного ключа user\_id.

Нет транзитивных зависимостей, поэтому таблица уже находится в 3НФ.

Таким образом, все таблицы находятся в 3НФ.

С конечной моделью разрабатываемой базы данных можно ознакомиться на рисунке 1.1.

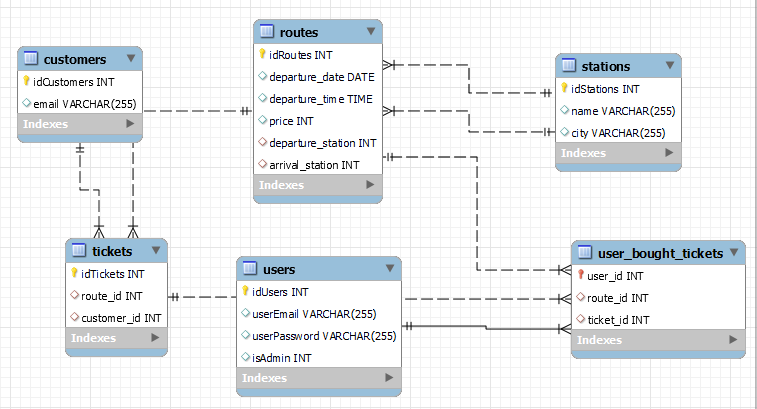


Рис. 1.1. Логически отредактированная информационная модель базы данных

## Выводы к первой главе

В данной работе были исследованы теоретические аспекты разработки баз данных, с учетом анализа понятий «база данных» и «система управления базами данных». Были рассмотрены три основные модели баз данных: иерархическая, реляционная и объектно-ориентированная, и определены их положительные и отрицательные стороны.

На этапе инфологического проектирования были установлены сущности, необходимые для построения ER-диаграммы, позволяющей наиболее естественно представить информацию, которую предполагается хранить в базе данных.

Далее был проведен обзор систем управления базами данных, используя метод анализа иерархий. В результате обзора было определено, что наиболее оптимальным вариантом является СУБД MySQL, поскольку она распространяется бесплатно для некоммерческого пользования, поддерживается наибольшим количеством операционных систем и обладает наименьшими требованиями к рабочему окружению.

Наконец, в результате логического проектирования исходные отношения, выделенные на этапе инфологического проектирования, были разложены на более простые. В целом, на основе проведенных исследований была разработана оптимальная база данных для использования в заданной предметной области.

# ГЛАВА 2. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ В ОБЛАСТИ РАЗРАБОТКИ БАЗ ДАННЫХ

## 2.1. Разработка проекта и базы данных

Для создания проекта по продаже и бронированию ж/д билетов через Интернет используется среда MySQL Workbench. Для создания нового проекта пользователю необходимо выбрать иконку  рядом с секцией Models в нижней части страницы домашнего экрана приложения (рис. 2.1.), либо выбрать опцию New Model в меню File (рис. 2.2.).

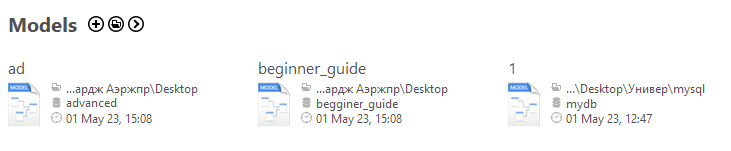


Рис. 2.1. Секция Models

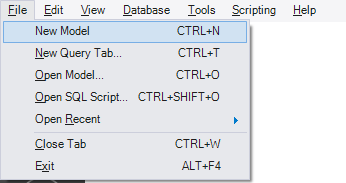


Рис. 2.2. Домашний экран приложения MySQL Workbench

Чтобы добавить новую схему в проект, нужно нажать на кнопку «Create a new schema» в правой части панели инструментов Physical Schemas. По умолчанию схема будет названа «new\_schema1», но можно изменить ее имя, отредактировав поле «Name». В данном случае схема была переименована в «railway\_database».

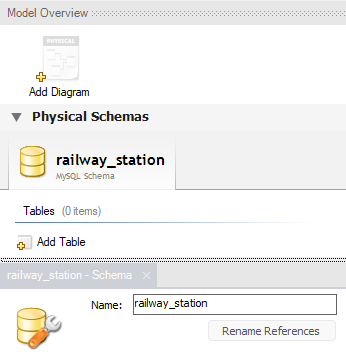


Рис. 2.3. Новая схема

## 2.2. Создание объектов базы данных

### 2.2.1. Создание таблиц, заполнение их тестовой информацией

Для создания пустой Enhanced Entity-Relationship (EER) диаграммы в MySQL Workbench необходимо выполнить следующие шаги: открыть окно Model Overview и кликнуть на иконку Add Diagram (рис. 2.4), после чего появится пустая диаграмма. Затем можно добавлять таблицы и связи между ними, используя панель инструментов, расположенную слева в окне Diagram. В данном проекте была создана пустая EER диаграмма, после чего была добавлена новая таблица при помощи этой панели инструментов.

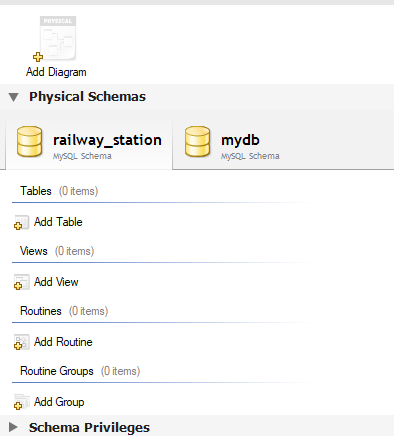


Рис. 2.4. Создание EER-диаграммы



Рис. 2.5. Панель инструментов

После добавления таблицы на EER-диаграмму, производится открытие редактора таблиц с именем таблицы по умолчанию «table1». Для изменения имени таблицы на «routes», необходимо отредактировать поле «Table Name». Далее, необходимо добавить поля в таблицу. Для этого можно дважды кликнуть по ячейке колонки «Column Name». Первое поле таблицы по умолчанию называется «idroutes», поскольку MySQL Workbench добавляет префикс «id» к имени таблицы для начального значения поля. Для выбора типа данных INT, а также для выбора флажков PK (PRIMARY KEY), NN (NOT NULL) и AI (AUTO\_INCREMENT), необходимо выполнить соответствующие действия в редакторе таблицы. Аналогичным образом можно создать поля «departure\_date», «departure\_time», «departure\_station», «arrival\_station» и «price» для таблицы «routes».

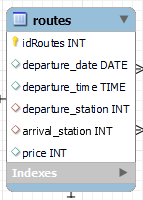


Рис 2.6. Таблица «routes»

После создания всех таблиц необходимо установить связи между ними. Для этого можно использовать панель инструментов, представленную на рисунке 2.5. Для создания связи «один-к-одному» нужно нажать на цифру 1, для создания связи «один-ко-многим» – на цифру 2, а для создания связи «многие-ко-многим» – на цифру 5 [10]. При создании связей внешние ключи устанавливаются автоматически. Если нужно отредактировать созданную связь, можно воспользоваться пунктом Edit Relationship контекстного меню связи [11].

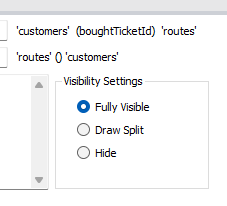


Рис. 2.7. Окно редактирования связей

Данный пункт позволяет изменять название, тип связи, а также класс принадлежности. EER диаграмма схемы railway представлена на рисунке 2.8.

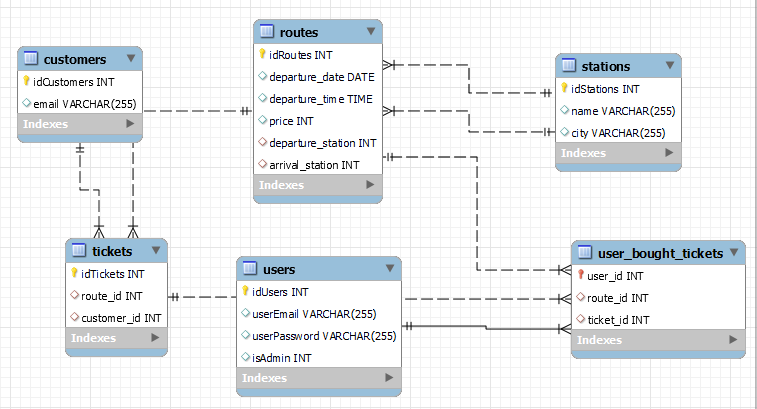


Рис. 2.8. EER-диаграмма базы данных

После создания модели базы данных необходимо выполнить синхронизацию с MySQL сервером. Для этого нужно выбрать в меню Database пункт Synchronize Model, который запустит мастер Synchronize Model with Database. На странице Connection Options можно выбрать соединение и задать дополнительные параметры, но если настройки уже определены, можно перейти к следующей странице, нажав кнопку Next.

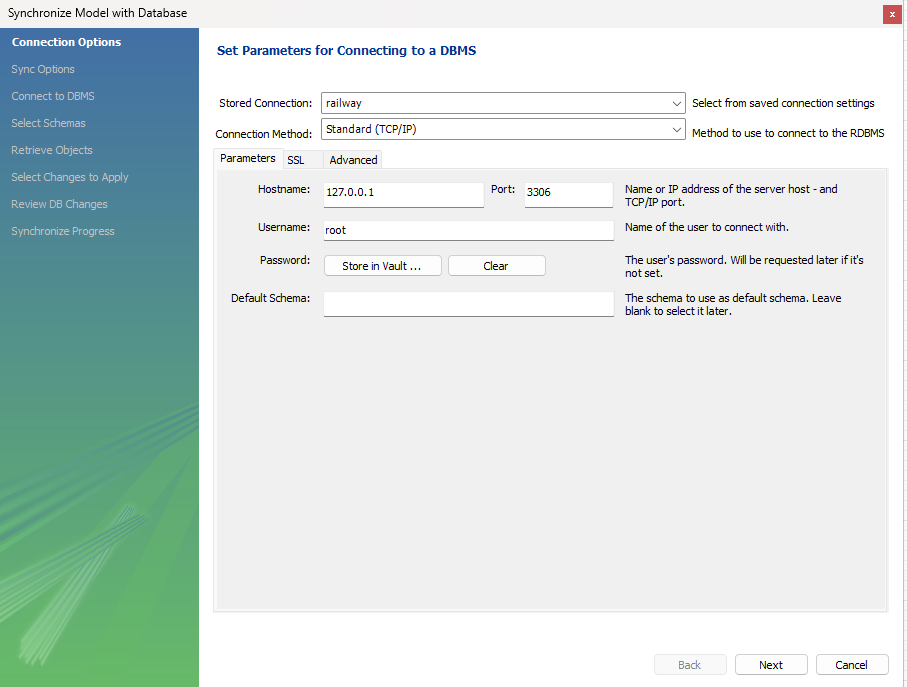


Рис. 2.9. Страница Connection Options мастера Synchronize Model with Database

Страница Sync Options мастера содержит расширенные настройки, которые не являются обязательными для данной базы данных. После этого переходим к следующей странице, где можно выбрать объекты базы данных для синхронизации с сервером. В данном случае необходимо синхронизировать 4 объекта, которые являются таблицами.

Окно Review DB Changes отображает SQL-сценарий, который будет выполнен на сервере для создания схемы «railway». Сценарий содержит запросы для создания таблиц. Кнопки Save To File… и Copy to Clipboard позволяют сохранить сценарий в файле и буфере обмена соответственно. На этом этапе можно внести необходимые изменения в SQL-сценарий.

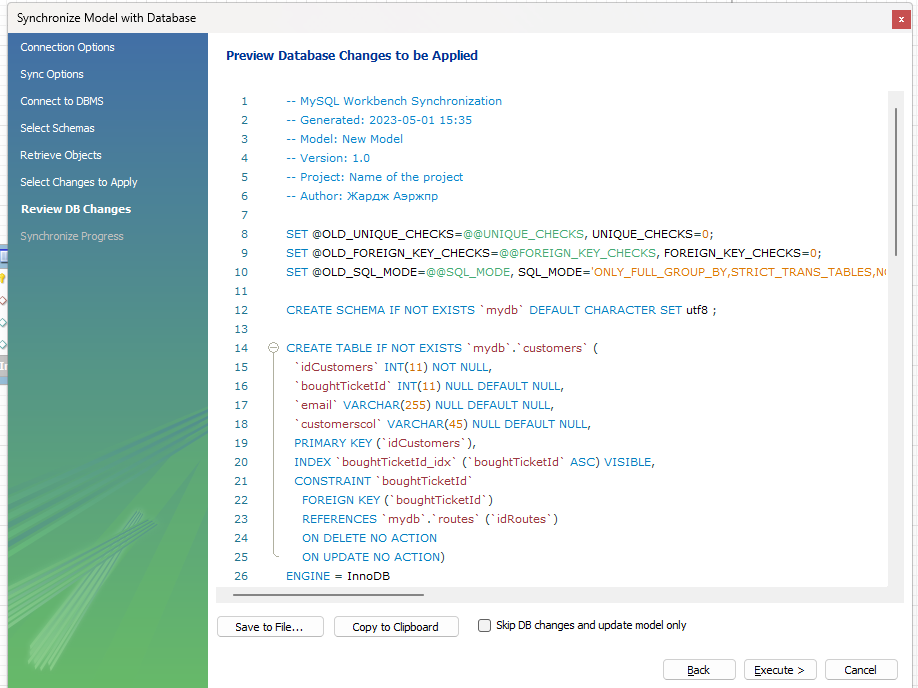


Рис. 2.10. SQL-запрос, создающий таблицы на стороне сервера

Для заполнения каждой из таблиц были созданы и выполнены определённые запросы. Все запросы заполнения расположены на  
рисунках 2.11. – 2.16.

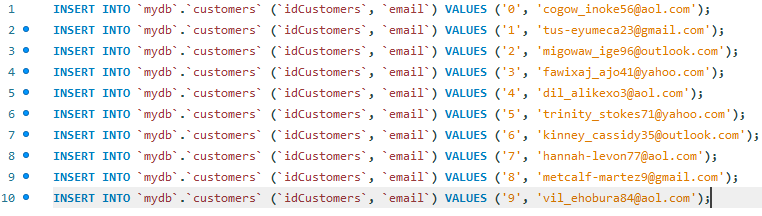


Рис. 2.11. Запрос на добавление информации в таблицу «customers»

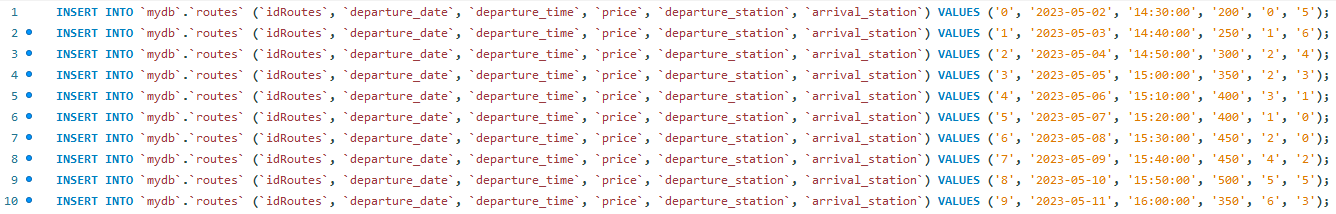


Рис. 2.12. Запрос на добавление информации в таблицу «routes»

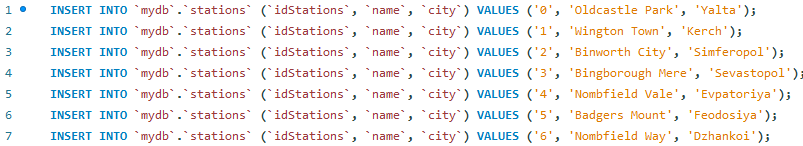


Рис. 2.13. Запрос на добавление информации в таблицу «stations»

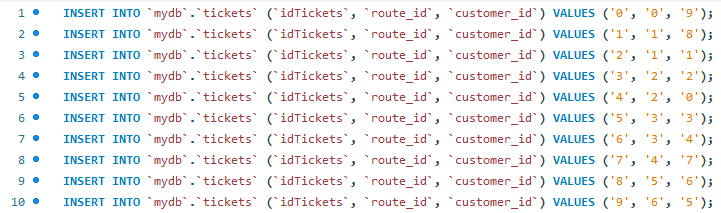


Рис. 2.14. Запрос на добавление информации в таблицу «tickets»

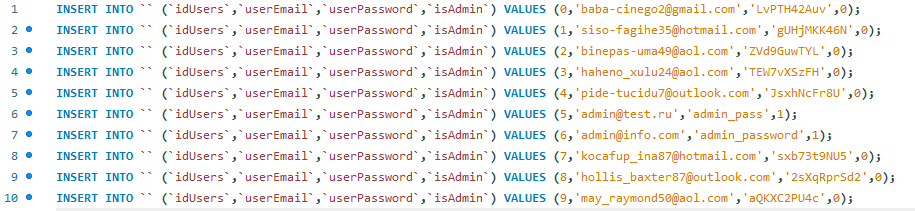


Рис. 2.15. Запрос на добавление информации в таблицу «users»

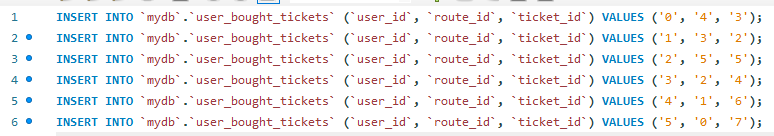


Рис. 2.16. Запрос на добавление информации в таблицу «user\_bought\_tickets»

### 2.2.2. Создание запросов

Для разработки базы данных продажа и бронирование ж/д билетов через Интернет необходимо создать запросы на выборку, которые будут показывать необходимые данные, такие как расписание отправления и станции прибытия. Для получения информации необходимо объединить таблицы routes и stations по нескольким ключам. В запросе необходимо выбрать значения поля departure\_station из таблицы routes и значения поля departure\_station\_city из таблицы stations. SQL-запрос показан на рисунке 2.11.

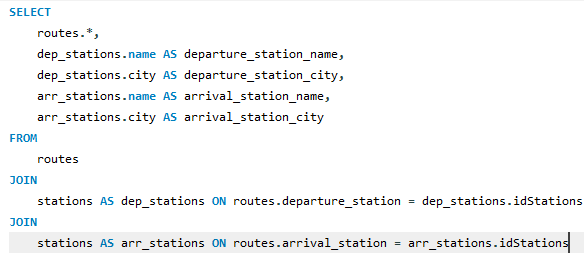


Рис. 2.11. SQL-запрос для объединения routes и stations по внешним ключам

Данный SQL-запрос выполняет выборку из таблицы routes с дополнительной информацией из таблиц stations в двух экземплярах с использованием оператора JOIN.

Первый оператор JOIN объединяет таблицы routes и stations по столбцу departure\_station, который является внешним ключом для связи с таблицей stations. Таким образом, в результирующую таблицу будут включены все записи из таблицы routes, а также дополнительные столбцы name и city из таблицы stations, соответствующие departure\_station.

Аналогичным образом второй оператор JOIN объединяет таблицы routes и stations по столбцу arrival\_station, который также является внешним ключом для связи с таблицей stations. В этом случае будут добавлены в результирующую таблицу столбцы name и city из таблицы stations, соответствующие arrival\_station.

Таким образом, в результате выполнения запроса можно получить таблицу, содержащую все записи из таблицы routes, а также информацию о станциях отправления и прибытия, включающую их названия и города.

Для сохранений изменений в базе данных применяются обновляющие запросы с условием совпадения уникального идентификатора записи таблицы, в которых используется SQL команда «Update». В переменную «id» передаётся первичный ключ, по которому SQL определяет куда записывать новые данные. Также, с «id» передаются и данные полей, которые необходимо обновить. Запрос на обновление строки в таблице «Сотрудники» представлен на рисунке 2.17.



Рис. 2.17. Запрос на обновление полей в таблице «routes»

Запрос использует оператор UPDATE, который позволяет изменять данные в таблице routes. Конкретный маршрут для изменения определяется с помощью условия WHERE idRoutes = ?, где idRoutes - это название столбца в таблице, а ? является параметром, который будет заменен на значение переменной route\_id.

В запросе есть 5 полей, которые должны быть обновлены в таблице: departure\_date, departure\_time, departure\_station, arrival\_station и price. Значения для этих полей берутся из параметров, переданных в POST-запросе: departure\_date, departure\_time, departure\_station, arrival\_station и price.

Вместо конкретных значений полей, в запросе используются плейсхолдеры ?, которые будут заменены на соответствующие значения, введенные пользователем.

## 2.3. Создание главного меню приложения

Главное меню приложения содержит следующие элементы: Главная, о нас, войти как администратор. Меню «Главная» — это главное страница, выбор которой приводит к переходу на главную страницу, где содержатся данные обо всех маршрутах. Меню «О нас» при открытии приводит к открытию вкладки, где содержится описание сайта. Меню «Войти как администратор» позволяет осуществить вход на сайт как администратор.



Рис. 2.18. Главное меню приложения

На рисунке 2.19 представлен код для обработки выбора пунктов главного меню.

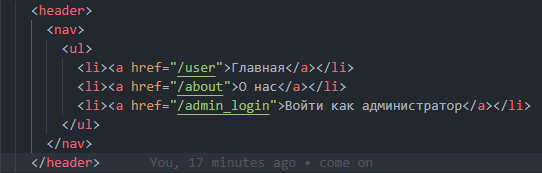


Рис. 2.19. Код для обработки пунктов меню

Главное меню было создано с использованием возможностей языка программирования EJS. В папку views был помещен файл header.ejs, который затем был включён в каждый маршрут проекта.

Главное меню было создано с помощью языков программирования HTML, CSS, EJS и Express.

## 2.4. Разработка системы оперативной справки

Система оперативной справки является важной частью базы данных «Продажа и бронирование ж/д билетов через Интернет». Она предназначена для обеспечения быстрого и качественного ответа на вопросы пользователей, включая информацию об услугах, оплате и других аспектах работы сайта. Реализация системы оперативной справки основана на использовании базы знаний, содержащей подробную информацию о продуктах, услугах и политике компании, которая регулярно обновляется сотрудниками магазина.

Для доступа к оперативной справке пользователи могут перейти на сайт и выбрать раздел «О нас», где им доступны функции поиска и просмотра информации, а также возможность задать свой вопрос через форму обратной связи или написать на почту для получения ответа [12]. Благодаря системе оперативной справки, количество обращений пользователей в службу поддержки снижается, что ускоряет процесс получения необходимой информации, повышает удовлетворенность пользователей работой и способствует увеличению продаж.



Рис. 2.20. Раздел оперативной справки

После заполнения пунктов соответствующей информацией, был создан файл, содержащий скомпилированные страницы.

## Выводы по главе 2

В настоящей главе была выполнена разработка базы данных, а также создан графический интерфейс, позволяющий управлять базой данных без знания языка запросов. Создание базы данных происходило в среде MySQL Workbench. Были созданы 4 таблицы, установлены связи между ними. Далее были созданы запросы на выборку, необходимые для формирования документов. Была разработана система оперативной справки, которая помогает снизить количество обращений пользователей в службу поддержки и ускорить процесс получения необходимой информации. В целом, создание онлайн магазина для продажи Ж/Д билетов — это сложный процесс, который требует знаний и умений в различных областях, таких как базы данных, программирование, веб-дизайн и другие.

# ЗАКЛЮЧЕНИЕ

Из результатов курсового проекта по разработке базы данных «Продажа и бронирование ж/д билетов через Интернет» можно сделать вывод о том, что создание подобного проекта требует серьезной работы в нескольких направлениях. Необходимо провести анализ рынка магазинов по продаже и бронированию ж/д билетов и конкурентов, чтобы определить уникальные особенности и преимущества, которые позволят привлечь и удержать целевую аудиторию.

Важно также уделить внимание дизайну сайта, чтобы он был привлекательным и удобным в использовании. Не менее важно создать удобную систему поиска и бронирования билетов, а также обеспечить безопасность платежей и доставку заказов. В ходе работы была разработана база данных, которая основана на MySQL. Были проанализированы существующие требования к подобного рода веб-приложениям, изучены основные принципы проектирования и создания баз данных, а также рассмотрены наиболее популярные методы для создания и управления онлайн магазинами.

В результате была разработана база данных, которая позволяет хранить информацию о ж/д билетах, клиентах, заказах и других важных аспектах функционирования магазина. Также была создана веб-страница, которая использует данные из базы данных для отображения информации о билетах, бронировании и управлении клиентскими профилями.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Статья «Что такое База Данных (БД)» [Электронный ресурс]/Хабр– Режим доступа: <https://habr.com/ru/post/555760/>.
2. Статья «Что нужно знать о развитии СУБД» [Электронный ресурс]/cnews.ru– Режим доступа: <https://www.cnews.ru/articles/2019-08-26_chto_nuzhno_znat_o_razvitii_subd>.
3. Статья «Что такое СУБД» [Электронный ресурс]/nic.ru – Режим доступа: <https://www.nic.ru/help/chto-takoe-subd_8580.html>.
4. Статья «Знакомство с реляционными базами данных» [Электронный ресурс]/DigitalOcean – Режим доступа: <https://www.digitalocean.com/community/tutorials/understanding-relational-databases-ru>.
5. Статья «Что такое реляционная база данных» [Электронный ресурс]/Oracle – Режим доступа: <https://www.oracle.com/cis/database/what-is-a-relational-database/>.
6. Post, Gerald V. Comparison of database management systems / Post, Gerald V. // Journal of Computer Information Systems. – 2001. – Vol. 41. – p. 43
7. Корнеева Е.В. Методика проектирования данных на основе слабоструктурированной информации / Е.В. Корнеева // Сервис в России и за рубежом. – 2009. – №2. – С. 64–76.
8. Статья «Логическое проектирование баз данных» [Электронный ресурс]/Oracle. Режим доступа: <https://oracle-patches.com/db/логическое-проектирование-баз-данных>.
9. Каскадное удаление [Электронный ресурс]/metanit. Режим доступа: <https://metanit.com/sharp/efcore/3.2.php>.
10. Инфологическое моделирование [Электронный ресурс]. Режим доступа: <https://studfile.net/preview/720097/>.
11. Елашкин М. Производительность СУБД и тесты ТРС. / М. Елашкин – 2014. – URL: <http://www.bytemag.ru/articles/detail.php?ID=8571>
12. [Территория новых возможностей. Вестник Владивостокского государственного](http://cyberleninka.ru/journal/n/territoriya-novyh-vozmozhnostey-vestnik-vladivostokskogo-gosudarstvennogo-universiteta-ekonomiki-i-servisa) [университета экономики и сервиса](http://cyberleninka.ru/journal/n/territoriya-novyh-vozmozhnostey-vestnik-vladivostokskogo-gosudarstvennogo-universiteta-ekonomiki-i-servisa). – 2011. – №2. – С. 101-105