МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ  
Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ   
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Базы данных»

Тема «Реализация базы данных железнодорожного вокзала с использованием средств мониторинга состояния СУБД»

**Исполнитель**

студент 2 курса 7 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. В. Тимошенко

подпись, дата

**Руководитель**

Ст. преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Кантарович В. С.

должность, учен. степень, ученое звание подпись, дата

Допущен к защите \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

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

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Кантарович В. С.

подпись дата инициалы и фамилия

Минск 2022

**Содержание**

[**Введение** 4](#_Toc134476276)

[**1. Аналитический обзор литературы по теме проекта** 5](#_Toc134476277)

[**1.1 Сравнительный анализ теоретических и эвристических методов решения поставленных задач и существующих по данной тематике технических научных решений** 5](#_Toc134476278)

[**1.2 Пример отечественных и зарубежных аналогов** 6](#_Toc134476279)

[**2. Изучение требований, определение вариантов использования** 7](#_Toc134476280)

[**2.1 Определение основных функциональных требований к базе данных** 7](#_Toc134476281)

[**2.2 Определение нефункциональных требований к базе данных** 7](#_Toc134476282)

[**2.3 Определение вариантов использования** 8](#_Toc134476283)

[**2.4 Вывод** 8](#_Toc134476284)

[**3. Анализ модели базы данных. Описание информационных объектов и ограничений целостности** 9](#_Toc134476285)

[**3.1 Проектирование базы данных** 9](#_Toc134476286)

[**3.1.1 Таблица Stations** 9](#_Toc134476287)

[**3.1.2 Таблица Routes** 10](#_Toc134476288)

[**3.1.3 Таблица Schedule** 10](#_Toc134476289)

[**3.1.4 Таблица Trains** 10](#_Toc134476290)

[**3.1.5 Таблица Vans** 11](#_Toc134476291)

[**3.1.6 Таблица Passengers** 11](#_Toc134476292)

[**3.1.7 Таблица Payments** 11](#_Toc134476293)

[**3.1.8 Таблица Tickets** 11](#_Toc134476294)

[**3.1.9 Таблица Stations\_routes** 12](#_Toc134476295)

[**3.2 Вывод** 12](#_Toc134476296)

[**4. Разработка модели базы данных** 13](#_Toc134476297)

[**4.1 Создание необходимых объектов** 13](#_Toc134476298)

[**4.1.1 Функции** 13](#_Toc134476299)

[**4.1.2 Процедуры** 14](#_Toc134476300)

[**4.1.3 Представления** 14](#_Toc134476301)

[**4.1.4 Триггеры** 15](#_Toc134476302)

[**4.2 Описание процедур импорта и экспорта** 16](#_Toc134476303)

[**4.3 Описание технологии** 17](#_Toc134476304)

[**4.4 Вывод** 18](#_Toc134476305)

[**5. Тестирование, проверка работоспособности и анализ полученных результатов.** 19](#_Toc134476306)

[**5.1. Тестирование клиентской части.** 19](#_Toc134476307)

[**5.2. Тестирование части менеджера** 20](#_Toc134476308)

[**5.3. Вывод.** 22](#_Toc134476309)

[**6. Руководство по использованию программного продукта** 23](#_Toc134476310)

[**6.1. Работа с приложением** 23](#_Toc134476311)

[**6.3. Вывод.** 26](#_Toc134476312)

[**Заключение** 27](#_Toc134476313)

[**Список использованных литературных источников** 28](#_Toc134476314)

**Введение**

В нашем современном мире имеется огромный спрос пользователей на хранилища информации, также называемые базами данных, которые могли бы постоянно работать и позволять удобно пользоваться хранимой инфромацией, также необходимо учитывать необходимость в использовании средств мониторинга состояния хранилища данных. Особенно важным становится использование баз данных в железнодорожной отрасли для обработки и хранения данных о состоянии железнодорожных вокзалов и поездов.

В связи с актуальностью данной работы целью курсового проекта является разработка и реализация базы данных Oracle и интерфейса для неё.

К задачам курсового проекта относится: аналтический обзор литературы по теме проекта, изучение требований; определение вариантов использования; анализ и проектирование модели данных; описание информационных объектов и ограничений целостности; создание необходимых объектов; импорт и экспорт данных; описание требуемой технологии; тестирование производительности; формирование вывода по каждому разделу; заключение, включающее вывод по проделанной работе.

Для базы данных обычно требуется комплексное программное обеспечение, которое называется системой управления базами данных (СУБД). СУБД служит интерфейсом между базой данных и пользователями или программами, предоставляя пользователям возможность получать и обновлять информацию, а также управлять ее упорядочением и оптимизацией. СУБД обеспечивает контроль и управление данными, позволяя выполнять различные административные операции, такие как мониторинг производительности, настройка, а также резервное копирование и восстановление.

Спектр применения систем управления базами данных на сегодняшний день практически необъятен – базы данных используются в интернете, в производстве, в промышленности, в маркетинге, в мобильных устройствах, в финансовой и банковской сферах, на телевидении, в телекоммуникациях и рекламе.

**1. Аналитический обзор литературы по теме проекта**

**1.1 Сравнительный анализ теоретических и эвристических методов решения поставленных задач и существующих по данной тематике технических научных решений**

Существует множество методов решения этой задачи, которые могут быть классифицированы по следующим параметрам:

1. Производительность: скорость выполнения запросов и эффективность использования ресурсов.
2. Надежность: способность системы работать без сбоев и сбоев, а также ее способность восстановиться после сбоя.
3. Гибкость: способность системы адаптироваться к изменениям требований и удовлетворять различным потребностям пользователей.
4. Расширяемость: возможность системы расширяться и добавлять новые функциональные возможности.

В теоретической литературе активно обсуждаются различные модели данных, в том числе реляционная, иерархическая и сетевая модели. Однако в контексте реализации базы данных железнодорожного вокзала реляционная модель является наиболее предпочтительной, так как она хорошо подходит для описания связей между различными элементами информационной системы, такими как станции, поезда, расписания, билеты и т.д.

Реляционная модель базы данных – это стандарт, который используется для описания и управления данными в большинстве современных баз данных. Для реализации базы данных железнодорожного вокзала можно использовать различные СУБД, такие как Oracle, Microsoft SQL Server, PostgreSQL и другие.

В литературе также описываются различные методы обработки данных, такие как SQL-запросы и NoSQL-базы данных. В контексте данной темы SQL-запросы являются более предпочтительными, так как они хорошо подходят для организации транзакций, которые важны для системы бронирования билетов и работы с расписаниями.

Для реализации базы данных железнодорожного вокзала с использованием средств мониторинга состояния СУБД можно использовать специализированные программные средства, такие как системы мониторинга и управления базами данных. Одним из примеров такой системы является Nagios, которая позволяет мониторить состояние базы данных, отслеживать ошибки и предупреждать об их возникновении.

Другой подход - использование инструментов профилирования кода, которые позволяют выявить узкие места в работе приложения и оптимизировать его.

Среди существующих технических решений можно выделить информационную систему "Железнодорожный транспорт", разработанную компанией "ООО Интерконтиненталь". Система предназначена для автоматизации работы железнодорожного транспорта и включает в себя модуль управления железнодорожными станциями и вокзалами, а также модуль мониторинга и управления базами данных.

Другим примером является система управления железнодорожным транспортом "ТРАНСДИЗЕЛЬ", разработанная ООО "Трансдизель". Система включает в себя модули управления поездами, грузами, вагонами и другими ресурсами железнодорожного транспорта. В системе используется СУБД PostgreSQL, а для мониторинга состояния базы данных применяется средство мониторинга pgBadger.

**1.2 Пример отечественных и зарубежных аналогов**

Из примеров успешной реализации мониторинга СУБД для железнодорожных систем является проект "Система мониторинга энергопотребления на железнодорожном транспорте" разработанный в России на базе БД Oracle[1]. В данном проекте были решены задачи сбора, хранения и анализа данных по энергопотреблению на железнодорожных объектах, таких как вокзалы, станции, электроподстанции и др. Реализация системы осуществляется через использование сети сбора данных, оборудования для их сбора и передачи на сервера базы данных, а также средств анализа и визуализации полученных данных.

Еще одним примером является проект "Управление железнодорожным транспортом на основе мониторинга движения поездов" разработанный в Китае. В рамках данного проекта была разработана система мониторинга движения поездов, которая позволяет отслеживать расписание движения, скорость движения, статус оборудования на поездах и многое другое. Система использует технологию GPS для определения местоположения поездов, а также средства мониторинга состояния СУБД для обработки и анализа полученных данных.

**1.3 Вывод**

В заключение можно отметить, что реализация базы данных железнодорожного вокзала с использованием средств мониторинга состояния СУБД является актуальной и перспективной задачей, которая может быть решена с помощью различных технических решений и инструментов, такие как Oracle и PostgreSQL, а для мониторинга состояния базы данных – различные средства мониторинга, такие как Oracle Enterprise Manager, Nagios и pgBadger. Однако при выборе конкретного подхода необходимо учитывать особенности конкретной задачи и требования к системе, такие как надежность, масштабируемость и производительность, а также проводить сравнительный анализ существующих решений и аналогов.

**2. Изучение требований, определение вариантов использования**

**2.1 Определение основных функциональных требований к базе данных**

В ходе изучения требований было выявлено, что база данных железнодорожного вокзала должна включать информацию о поездах, вагонах, расписании движения, билетах и о пассажирах. Были определены следующие варианты использования:

1. Хранение информации о расписании прибытия и отправления поездов на определенную дату и время.
2. Предоставление информации о расписании движения поездов пассажирам.
3. Возможность просмотра и редактирования информации о поездах, включая название, маршрут, время отправления и прибытия, количество свободных мест.
4. Хранение информации о билетах на поезда, включая стоимость, маршрут, дату и время отправления, место в вагоне.
5. Возможность продажи билетов на поезда через систему онлайн-бронирования.
6. Ведение учета проданных билетов и свободных мест на каждый поезд.
7. Хранение информации о пассажирах, включая имя, фамилию, номер паспорта, контактную информацию и предоставляемые льготы.
8. Обеспечение защиты данных и доступа только авторизованных пользователей к базе данных.
9. Мониторинг состояния базы данных и обнаружение проблем в работе СУБД, включая информацию о возникающих ошибках и неполадках, о времени отклика системы, а также о загрузке процессора и объеме свободного места на диске.

**2.2 Определение нефункциональных требований к базе данных**

В качестве нефункциональных требований, можно выделить:

1. Безопасность: База данных должна обеспечивать безопасность данных и защиту от несанкционированного доступа, в том числе использование механизмов авторизации и аутентификации.
2. Производительность: База данных должна иметь высокую производительность и обеспечивать быстрый доступ к данным для обеспечения эффективной работы вокзала.
3. Надежность: База данных должна быть надежной и стабильной, обеспечивать целостность данных и иметь возможность быстрого восстановления после сбоев.
4. Масштабируемость: База данных должна быть масштабируемой и гибкой, чтобы обеспечивать эффективное управление растущим объемом данных вокзала.
5. Удобство использования: База данных должна быть удобной и простой в использовании для обеспечения эффективной работы пользователей.

**2.3 Определение вариантов использования**

Вариантами использования базы данных являются:

1. Просмотр расписания отправления и прибытия поездов;
2. Просмотр информации о наличии мест на поездах;
3. Бронирование мест на поезд;
4. Просмотр информации о задержках и отменах поездов;
5. Просмотр истории покупок билетов;

**2.4 Вывод**

Для определения функиональных требований были рассмотрены варианты использования, которые позволяют эффективно применять СУБД для решения задач, схожих с поставленной. Были определены нефункциональные варианты использования, необходимые качественному продукту.

Таким образом, изучение требований и определение вариантов использования позволяет определить основные функциональные и нефункциональные требования к базе данных, а также выявить возможные варианты использования системы.

**3. Анализ модели базы данных. Описание информационных объектов и ограничений целостности**

**3.1 Проектирование базы данных**

Для разработки веб-приложения в рамках курсового проекта понадобилась база данных с необходимой конфигурацией сущностей для хранения всей необходимой информации.

Структуру базу данных, ограничения целостности, связи и поля можно увидеть в приложении A и на рисунке 3.1.

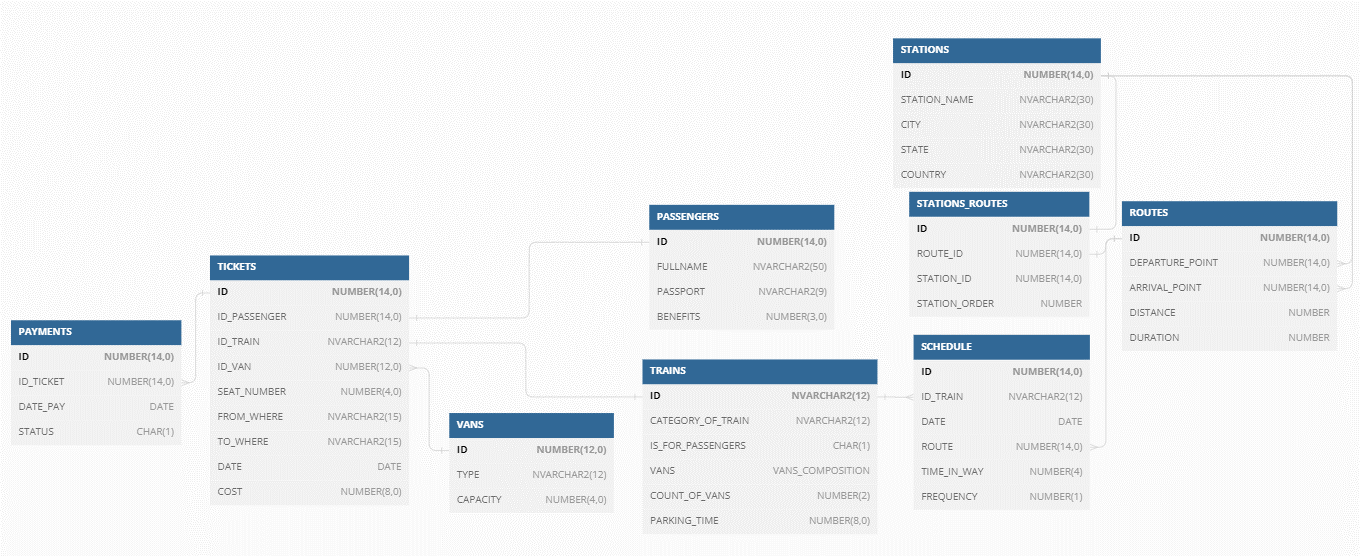


Рисунок 3.1 – Структура базы данных

В базе данных существует 8 таблиц, каждая из которых выполняет свою функцию.

### **3.1.1 Таблица Stations**

Данная таблица используется для хранения информации о станциях остановки составов.

В её состав входят следущие столбцы:

* Столбец ID. Хранит уникальный идентификатор каждой точки остановки.
* Cтолбец STATION\_NAME. Хранит название станции.
* Cтолбец CITY. Хранит название города, в котором расположена станция.
* Cтолбец STATE. Хранит название региона, в котором расположена станция.
* Cтолбец COUNTRY. Хранит название страны, в котором расположена станция.

Данная таблица содержит ограничение перичного ключа для столбца ID, а также все столбцы должны содержать какое-либо значение.

### **3.1.2 Таблица Routes**

Данная таблица используется для хранения информации о маршрутах передвижения составов.

В её состав входят следущие столбцы:

* Столбец ID. Хранит уникальный идентификатор каждого маршрута.
* Cтолбец DEPARTURE\_POINT. Хранит ID начальной точки.
* Cтолбец ARRIVAL\_POINT. Хранит ID конечной точки.
* Cтолбец DISTANCE. Хранит расстояние, проходящее составом.
* Cтолбец DURATION. Хранит время, проводимое в пути.

Данная таблица содержит ограничение перичного ключа для столбца ID. Также для столбцов DEPARTURE\_POINT и ARRIVAL\_POINT существуют ограничения целостности по внешнему ключу к таблице STATIONS.

### **3.1.3 Таблица Schedule**

Данная таблица используется для хранения информации о расписании передвижения составов.

В её состав входят следущие столбцы:

* cтолбец ID. Хранит уникальный идентификатор каждого маршрута.
* Cтолбец ID\_TRAIN. Хранит ID поезда, зарезервированного для маршрута.
* Cтолбец DATE. Хранит дату начала выдвижения состава по маршруту.
* Cтолбец ROUTE. Хранит ID маршрута передвижения состава.
* Cтолбец TIME\_IN\_WAY. Хранит время состава в пути.
* Cтолбец FREQUENCY. Хранит частоту использования данной ячейки расписания.

Данная таблица содержит ограничение перичного ключа для столбца ID. Также для столбцов ID\_TRAIN и ROUTE существуют ограничения целостности по внешнему ключу к таблице TRAINS и ROUTES соответсвенно.

### **3.1.4 Таблица Trains**

Данная таблица используется для хранения информации о поездах(составах).

В её состав входят следущие столбцы:

* Столбец ID. Хранит уникальный идентификатор каждого состава.
* Cтолбец CATEGORY\_OF\_TRAIN. Хранит категорию данного состава.
* Cтолбец IS\_FOR\_PASSENGERS. Хранит символ, обозначающий возможность использования данного состава для перевозки пассажиров.
* Cтолбец VANS. Хранит массив ID вагонов, из которых состоит поезд.
* Cтолбец COUNTS\_OF\_VANS. Хранит количество вагонов в составе.
* Cтолбец PARKING\_TIME. Хранит время стоянки состава.

Данная таблица содержит ограничение перичного ключа для столбца ID.

### **3.1.5 Таблица Vans**

Данная таблица используется для хранения информации о вагонах.

В её состав входят следущие столбцы:

* Столбец ID. Хранит уникальный идентификатор каждого вагона.
* Cтолбец TYPE. Хранит тип вагона.
* Cтолбец CAPACITY. Хранит вместимость вагона.

Данная таблица содержит ограничение перичного ключа для столбца ID.

### **3.1.6 Таблица Passengers**

Данная таблица используется для хранения информации о пассажирах.

В её состав входят следущие столбцы:

* Столбец ID. Хранит уникальный идентификатор каждого пассажира.
* Cтолбец FULLNAME. Хранит ФИО пассажира.
* Cтолбец PASSPORT. Хранит паспортные данные пассажира.
* Cтолбец BENEFITS. Хранит льготы, применяемые к пассажиру.

Данная таблица содержит ограничение перичного ключа для столбца ID.

### **3.1.7 Таблица Payments**

Данная таблица используется для хранения информации о платежах.

В её состав входят следущие столбцы:

* Столбец ID. Хранит уникальный идентификатор каждого платежа.
* Cтолбец ID\_TICKET. Хранит ID билета, по которому произошла оплата.
* Cтолбец DATE\_PAY. Хранит дату оплаты.
* Cтолбец STATUS. Хранит статус оплаты.

Данная таблица содержит ограничение перичного ключа для столбца ID. Также для столбцов ID\_TICKET существует ограничение целостности по внешнему ключу к таблице TICKETS.

### **3.1.8 Таблица Tickets**

Данная таблица используется для хранения информации о билетах.

В её состав входят следущие столбцы:

* Столбец ID. Хранит уникальный идентификатор каждого билета.
* Cтолбец ID\_PASSENGER. Хранит ID пассажира, которому был выдан билет.
* Cтолбец ID\_TRAIN. Хранит ID поезда, на который забронирован билет.
* Cтолбец ID\_VAN. Хранит ID вагона, в котором находится арендованное место.
* Cтолбец SEAT\_NUMBER. Хранит номер арендованного места.
* Cтолбец FROM\_WHERE. Хранит место посадки на поезд.
* Cтолбец TO\_WHERE. Хранит место выхода из поезда.
* Cтолбец DATE. Хранит дату поездки.
* Cтолбец COST. Хранит стоимость билета.

Данная таблица содержит ограничение перичного ключа для столбца ID. Также для столбцов ID\_PASSENGER, ID\_TRAIN и ID\_VAN существуют ограничения целостности по внешнему ключу к таблице PASSENGERS, TRAINS и VANS соответсвенно.

### **3.1.9 Таблица Stations\_routes**

Данная таблица используется для хранения информации о билетах.

В её состав входят следущие столбцы:

* Столбец ID. Хранит уникальный идентификатор каждой станции в маршруте.
* Cтолбец ROUTE\_ID. Хранит ID маршрута, в котором привязана станция.
* Cтолбец STATION\_ID. Хранит ID станции, которая используется в маршруте.
* Cтолбец STATION\_ORDER. Хранит номер станции в маршруте.

Данная таблица содержит ограничение перичного ключа для столбца ID. Также для столбцов ROUTE\_ID, STATION\_ID существуют ограничения целостности по внешнему ключу к таблице ROUTES, STATIONS соответсвенно.

**3.2 Вывод**

В ходе анализа базы данных необходимо была составлена UML диаграмма базы данных, а также описана каждая таблица в диаграмме.

Таким образом, изучение требований и определение вариантов использования позволяет определить основные функциональные и нефункциональные требования к базе данных, а также выявить возможные варианты использования системы.

**4. Разработка модели базы данных**

Модель базы данных – это структура, которая определяет, как данные будут храниться, организовываться и управляться в базе данных. Она описывает типы данных, связи между таблицами, ограничения целостности данных и другие аспекты, необходимые для правильного функционирования базы данных.

Правильная модель базы данных обеспечивает эффективное хранение и управление данными, а также обеспечивает защиту данных и целостность базы данных.

**4.1 Создание необходимых объектов**

Для реализации базы данных железнодорожного вокзала необходимо создать набор объектов базы данных, которые представляют собой таблицы, представления, индексы, ограничения и хранимые процедуры. Все эти объекты должны быть созданы в соответствии с требованиями к производительности, безопасности и надежности.

В проектируемой базе данных были созданы 9 таблиц, описание которых приведено выше.

Для взаимодействия с этими таблицами были разработаны такие объекты базы данных, как функции, процедуры, представления и триггеры.

### **4.1.1 Функции**

Функция в Oracle – это объект базы данных, который возвращает значение на основе переданных в нее аргументов. Функции могут быть созданы с помощью оператора CREATE FUNCTION. Пример функции, которая удаляет из массива VARRAY определенное значение – листинг 4.1.

|  |
| --- |
| CREATE OR REPLACE FUNCTION *delete\_entry*(p\_record IN VANS\_COMPOSITION, p\_val IN NUMBER) RETURN VANS\_COMPOSITION IS  v\_ret VANS\_COMPOSITION := VANS\_COMPOSITION(); BEGIN  FOR n IN p\_record.FIRST..p\_record.LAST LOOP  IF p\_record(n) != p\_val THEN  v\_ret.EXTEND;  v\_ret(v\_ret.LAST) := p\_record(n);  END IF;  END LOOP;  RETURN v\_ret; END; |

Листинг 4.1 – Функция работы с массивом

Здесь мы создаем функцию delete\_entry, которая принимает массив чисел и одно число и возвращает новый массив без этого числа.

### **4.1.2 Процедуры**

Процедура в Oracle – это объект базы данных, который представляет собой набор SQL-инструкций, которые могут быть вызваны для выполнения определенной задачи. Процедуры могут быть созданы с помощью оператора CREATE PROCEDURE. Пример определения процедуры, которая обновляет вместимость вагонов, по прошествии даты поездки в билете – листинг 4.2.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE *UPDATE\_VANS\_CAPACITY* AS BEGIN DECLARE  CURSOR cur IS SELECT ID\_VAN FROM TICKETS WHERE TICKETS."DATE" < *SYSDATE*;  TYPE id\_tab\_type IS TABLE OF TICKETS.ID\_VAN%TYPE INDEX BY PLS\_INTEGER;  id\_tab id\_tab\_type; BEGIN  OPEN cur;  LOOP  FETCH cur BULK COLLECT INTO id\_tab LIMIT 1000;  EXIT WHEN id\_tab.COUNT = 0;  FORALL i IN 1..id\_tab.COUNT  UPDATE VANS SET CAPACITY = CAPACITY + 1  WHERE ID = id\_tab(i);  END LOOP;  CLOSE cur; END; end; |

Листинг 4.2 – Процедура UPDATE\_VANS\_CAPACITY

Данный код представляет собой определение процедуры UPDATE\_VANS\_CAPACITY, которая использует курсор для выборки идентификаторов вагонов из таблицы TICKETS и обновляет информацию о вместимости вагонов в таблице VANS.

Процедура позволяет эффективно обновлять информацию о вместимости вагонов на железнодорожных станциях, используя массовое обновление с помощью курсора и массива.

### **4.1.3 Представления**

Представление в Oracle – это объект базы данных, который представляет собой виртуальную таблицу, основанную на результатах выполнения SQL-запроса. Представления могут быть созданы с помощью оператора CREATE VIEW. Пример определения представления, которое используется для отображения расписания – листинг 4.3.

|  |
| --- |
| CREATE OR REPLACE VIEW TAKE\_SCHEDULE AS  SELECT SCHEDULE."ID", ID\_TRAIN, TRAINS.CATEGORY\_OF\_TRAIN,  ROUTES.DEPARTURE\_POINT, ROUTES.ARRIVAL\_POINT,  ROUTES.DISTANCE, ROUTES.DURATION,  "DATE", FREQUENCY, TRAINS.IS\_FOR\_PASSENGERS  FROM SCHEDULE  JOIN TRAINS ON SCHEDULE.ID\_TRAIN = TRAINS.ID  JOIN ROUTES ON SCHEDULE.ROUTE = ROUTES.ID; |

Листинг 4.3 – Представление TAKE\_SCHEDULE

Данный код создает представление, которое объединяет таблицы "SCHEDULE", "TRAINS" и "ROUTES", чтобы получить информацию о графике движения поездов. В результате выполнения данного кода можно получить доступ к перечню поездов, их категории, маршруту, дате, частоте и другой информации, связанной с графиком движения.

### **4.1.4 Триггеры**

Триггер в Oracle – это объект базы данных, который автоматически запускается при определенном событии в базе данных, таком как вставка, обновление или удаление данных. Триггеры могут быть созданы с помощью оператора CREATE TRIGGER. Пример определения триггера, который проверяет вагоны в поезде – листинг 4.4.

|  |
| --- |
| CREATE OR REPLACE TRIGGER TRAINS\_INSERT\_UPDATE\_TRIGGER  BEFORE INSERT OR UPDATE ON TRAINS  FOR EACH ROW  DECLARE  vans\_in VANS\_COMPOSITION := VANS\_COMPOSITION();  vans\_out VANS\_COMPOSITION := VANS\_COMPOSITION();  vansfree NUMBER(1, 0);  BEGIN  IF INSERTING THEN  vans\_in := :NEW.VANS;  vans\_out := :NEW.VANS;  FOR I IN vans\_in.FIRST .. vans\_in.LAST LOOP  SELECT IS\_FREE INTO vansfree FROM VANS WHERE ID = vans\_in(I);  IF vansfree = 0 THEN  vans\_out := DELETE\_ENTRY(vans\_out, vans\_in(I));  ELSE  UPDATE VANS SET IS\_FREE = 0 WHERE ID = vans\_in(I);  END IF;  END LOOP;  :NEW.COUNT\_OF\_VANS := vans\_out.COUNT;  :NEW.VANS := vans\_out;  ELSIF UPDATING('VANS') THEN  vans\_in := :OLD.VANS;  FOR I IN vans\_in.FIRST .. vans\_in.LAST LOOP  SELECT IS\_FREE INTO vansfree FROM VANS WHERE ID = vans\_in(I);  UPDATE VANS SET IS\_FREE = 1 WHERE ID = vans\_in(I);  END LOOP;  vans\_in := :NEW.VANS;  FOR I IN vans\_in.FIRST .. vans\_in.LAST LOOP  SELECT IS\_FREE INTO vansfree FROM VANS WHERE ID = vans\_in(I);  IF vansfree = 0 THEN  vans\_out := DELETE\_ENTRY(vans\_out, vans\_in(I));  ELSE  UPDATE VANS SET IS\_FREE = 0 WHERE ID = vans\_in(I);  END IF;  END LOOP;  :NEW.VANS := vans\_out;  :NEW.COUNT\_OF\_VANS := vans\_out.COUNT;  END IF;  END; |

Листинг 4.4 – Триггер TRAINS\_INSERT\_UPDATE\_TRIGGER

Данный код представляет собой триггер, который запускается перед операциями INSERT или UPDATE на таблице "TRAINS". Он обходит массив VARAAY VANS\_COMPOSITION для поиска плохих данные (повторяющихся и недоступных значений), при нахождении выполняется удаление данных из массива.

В результате выполнения кода, массив значений будет содержать только уникальные и разрешенные значения.

**4.2 Описание процедур импорта и экспорта**

Процедуры импорта и экспорта являются важными инструментами для перемещения данных между различными базами данных или для создания резервных копий базы данных.

JSON (JavaScript Object Notation) – это легкий формат обмена данными, который используется для представления структурированных данных в виде пар "имя/значение". Он широко используется в веб-приложениях и мобильных приложениях, а также в базах данных.

Использование формата JSON для импорта и экспорта данных предоставляет удобный и гибкий способ обмена данными между различными базами данных или веб-приложениями. Он позволяет представлять структурированные данные в виде пар "имя/значение" и легко преобразовывать данные в различные форматы, такие как XML или CSV.

Процедуры импорта и экспорта данных могут быть полезны для создания резервных копий базы данных, для переноса данных между различными окружениями или для обмена данными с другими приложениями. Они также обеспечивают удобный способ загрузки данных из внешних источников и экспорта данных для анализа и обработки в других приложениях – листинг 4.5.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE TO\_JSON(return\_out out int)  IS BEGIN  DECLARE  output\_file UTL\_FILE.FILE\_TYPE;  json\_data CLOB;  CURSOR vans\_cursor IS  SELECT JSON\_OBJECT(  'type' VALUE TYPE,  'capacity' VALUE CAPACITY,  'is\_free' VALUE IS\_FREE  ) AS json\_data  FROM VANS  WHERE ROWNUM < 10;  BEGIN  output\_file := UTL\_FILE.FOPEN('UTL\_DIR', 'VANS.JSON', 'W');  FOR van\_rec IN vans\_cursor LOOP  json\_data := van\_rec.json\_data;  UTL\_FILE.PUT\_LINE(output\_file, json\_data);  END LOOP;  UTL\_FILE.FCLOSE(output\_file);  return\_out := 1;  END;  end; |

Листинг 4.5 – Пример процедуры экспорта из базы данных

Процедура производит экспорт первых 9 строк из таблицы VANS в файл VANS.JSON.

В целом, процедуры импорта и экспорта данных являются важными инструментами для обеспечения эффективной работы базы данных и обеспечения ее надежности.

**4.3 Описание технологии**

Мониторинг состояния СУБД вляется важным инструментом для обеспечения высокой производительности и надежности базы данных. Он позволяет анализировать работу базы данных в реальном времени и выявлять проблемы, которые могут влиять на ее работу.

В данном проекте используется Oracle Enterprise Manager (OEM), который предоставляет средства мониторинга, управления и администрирования баз данных Oracle. OEM обеспечивает централизованное управление всеми аспектами работы базы данных, включая аппаратное обеспечение, операционную систему, СУБД и приложения.

Одной из главных функций OEM является мониторинг производительности базы данных. Он позволяет анализировать данные о нагрузке на базу данных, использовании ресурсов и производительности запросов – рисунок 4.6. OEM также предоставляет возможность установки и настройки предупреждений и автоматических действий в случае возникновения проблем с производительностью.

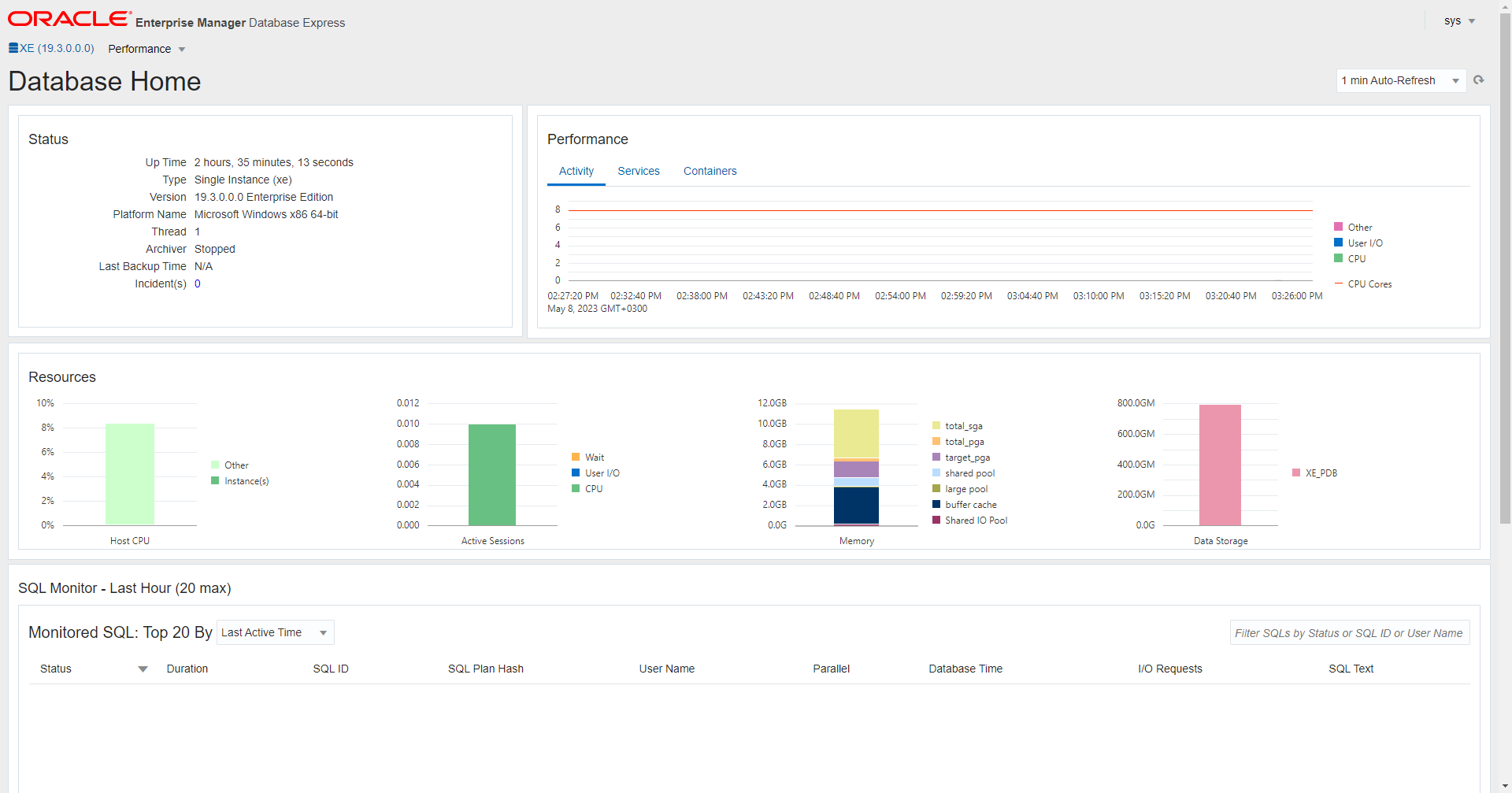


Рисунок 4.6 – Oracle Enterprise Manager

Данное ПО доступно из любого браузера по ссылке, настраиваемой администратором, что значительно упрощает доступ для мониторинга состояния базы данных.

Кроме того, OEM обеспечивает механизмы автоматического обнаружения и устранения проблем в работе базы данных. Он позволяет анализировать журналы ошибок, настраивать систему резервного копирования, мониторить доступность базы данных и многое другое.

Таким образом, использование Oracle Enterprise Manager в проекте обеспечивает высокую производительность и надежность базы данных, а также облегчает ее администрирование и управление.

**4.4 Вывод**

В данном разделе была разработана модель данных. Также были описаны процедуры импорта и экспорта из JSON. Использование формата JSON для импорта и экспорта данных позволяет удобно и гибко обмениваться данными между различными приложениями и базами данных, что может упростить работу с данными и повысить эффективность работы в целом. Была рассмотрена используемая технология, которая значительно упрощает работу по администрированию базы данных.

# **5. Тестирование, проверка работоспособности и анализ полученных результатов.**

## **5.1. Тестирование клиентской части.**

Для клиентской части были написаны процедуры работы с данными из базы данных. А именно добавления, изменения и удаления данных. Процедура INSERT\_TICKETS, которая используется для вставки значений в таблицу TICKETS, рисунок 5.1.





Рисунок 5.1 – Процедура INSERT\_TICKETS

Далее процедура INSERT\_PASSENGERS, которая используется для вставки значений в таблицу PASSENGERS, рисунок 5.2.





Рисунок 5.2 – Процедура INSERT\_PASSENGERS

Далее процедура INSERT\_PAYMENTS, которая используется для вставки значений в таблицу PAYMENTS, рисунок 5.3.





Рисунок 5.3 – Процедура INSERT\_PAYMENTS

Также после вставки (или обновления) строки в таблицу PAYMENTS, должен выполнятся триггер для изменения количества свободных мест в вагоне, связанным с оплаченным билетом, рисунок 5.4.

|  |
| --- |
| а |
| б |

Рисунок 5.4 – Результат применения триггера: а – до, б – после

После этого можно проверить представление TAKE\_SCHEDULE, вывод данных продемонстрирован на рисунке 5.5.

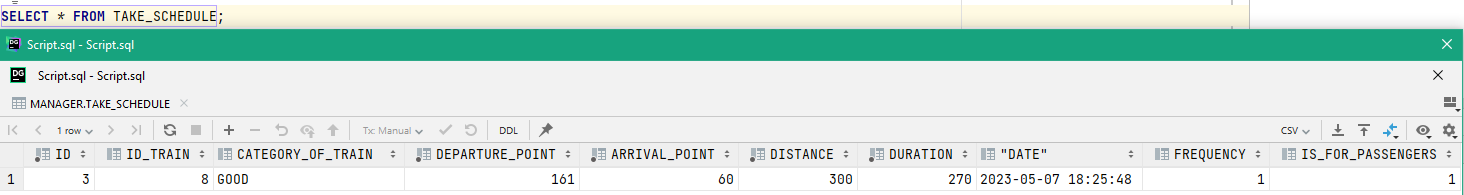


Рисунок 5.5 – представление TAKE\_SCHEDULE

Проверка представления TAKE\_TICKET, рисунок 5.6.



Рисунок 5.6 – Представление TAKE\_TICKET

А также процедуры INSERT\_TAKE\_TICKET для него, рисунок 5.7.





Рисунок 5.7 – Процедура INSERT\_TAKE\_TICKET

## **5.2. Тестирование части менеджера**

Для начала тестирования части менеджера необходимо дозаполнить таблицы данными. Процедура INSERT\_VANS продемонстрирована на рисунке 5.6.





Рисунок 5.6 – Процедура INSERT\_VANS

Далее процедура INSERT\_TRAINS, которая используется для вставки значений в таблицу TRAINS, рисунок 5.7.





Рисунок 5.7 – Процедура INSERT\_TRAINS

Далее процедура INSERT\_ROUTES, которая используется для вставки значений в таблицу ROUTES, рисунок 5.8.





Рисунок 5.8 – Процедура INSERT\_ROUTES

Далее процедура INSERT\_STATIONS, которая используется для вставки значений в таблицу STATIONS, рисунок 5.9.





Рисунок 5.9 – Процедура INSERT\_STATIONS

Далее процедура INSERT\_STATIONS\_ROUTES, которая используется для вставки значений в таблицу STATION\_ROUTES, рисунок 5.10.





Рисунок 5.10 – Процедура INSERT\_STATIONS\_ROUTES

Далее процедура INSERT\_SCHEDULE, которая используется для вставки значений в таблицу SCHEDULE, рисунок 5.11.





Рисунок 5.11 – Процедура INSERT\_SCHEDULE

Аналогичны этим процедурам, процедуры изменения и удаления из базы данных.

Далее протестируем работоспособность триггера TRAINS\_INSERT\_UPDATE\_TRIGGER, рисунок 5.12.



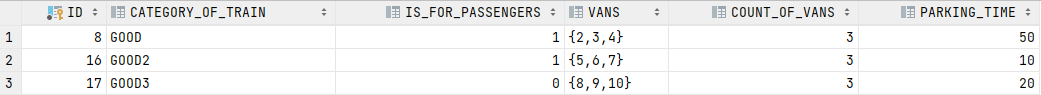
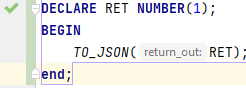


Рисунок 5.12 – Проверка триггера TRAINS\_INSERT\_UPDATE\_TRIGGER

Далее протестируем работоспособность процедур экспорта и импорта данных из JSON, рисунок 5.13, 5.14.



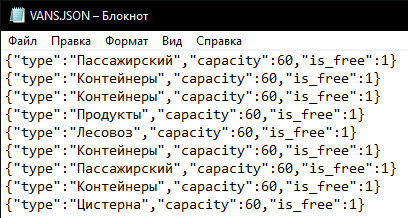


Рисунок 5.13 – Проверка процедуры экспорта

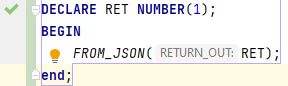




Рисунок 5.14 – Проверка процедуры импорта

**5.3. Вывод.**

В данном разделе были протестированы возможности клиентов и менеджеров. По итогам тестирования можно сказать, что все работает корректно.

# **6. Руководство по использованию программного продукта**

**6.1. Работа с приложением**

Первое, что видит пользователь приложения – это страница авторизации. Она представлена на рисунке 6.1.

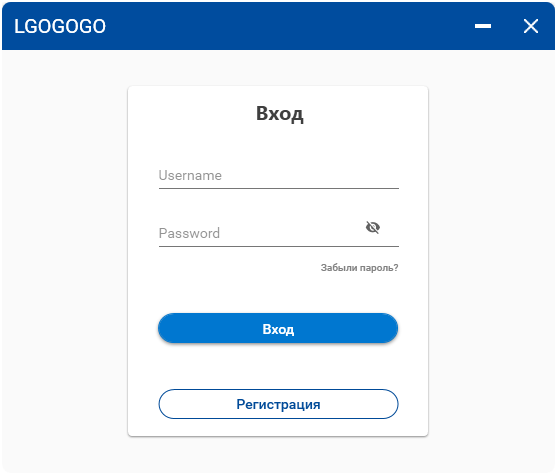


Рисунок 6.1 – Страница авторизации

Для дальнейшей работы с приложением, на этой странице необходимо ввести данные для входа. После чего открывается главная страница с надписью «Приветсвуем в приложении БЧ!». Слева расположено меню, со страницами для перехода, также, в зависимости от того, кто вошел в систему (пользователь или менеджер), он отличается, для менеджера добавляется возможность перехода на страницу с панелью администратора, рисунок 6.2.

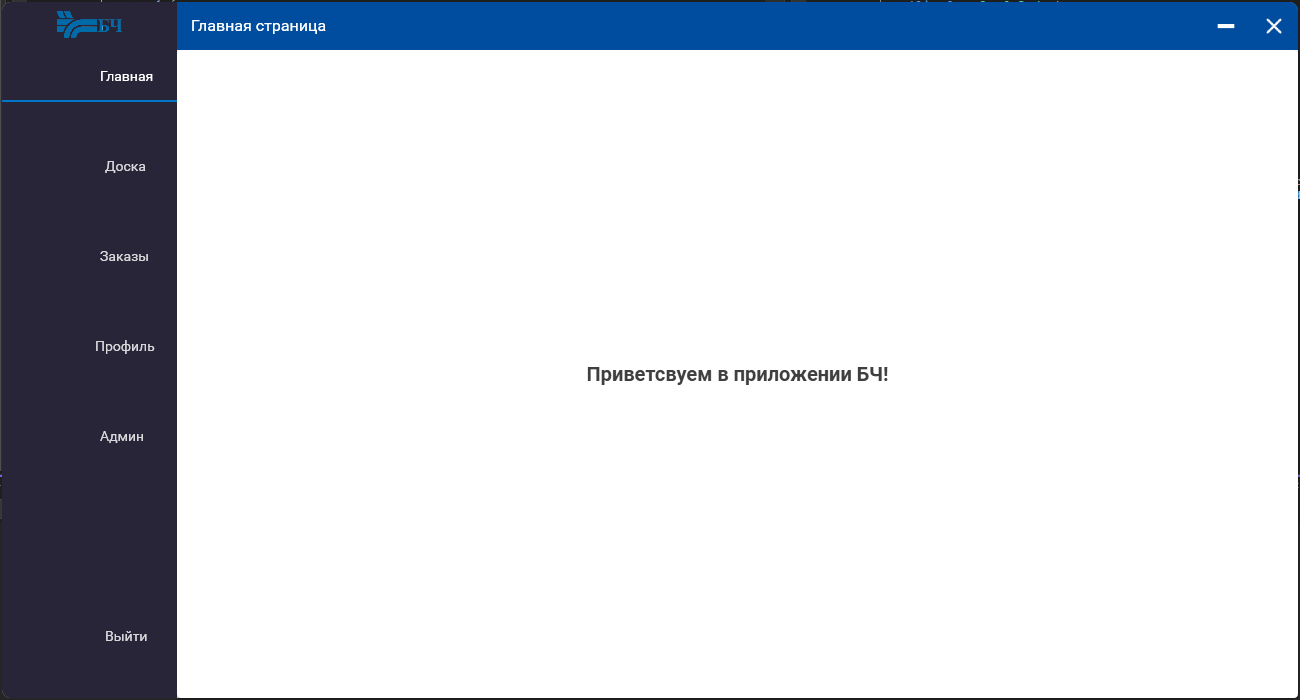


Рисунок 6.2 – Главная страница приложения

Сама панель администратора, представляет собой меню выбора таблицы для непосредственного её редактирования, рисунок 6.3.

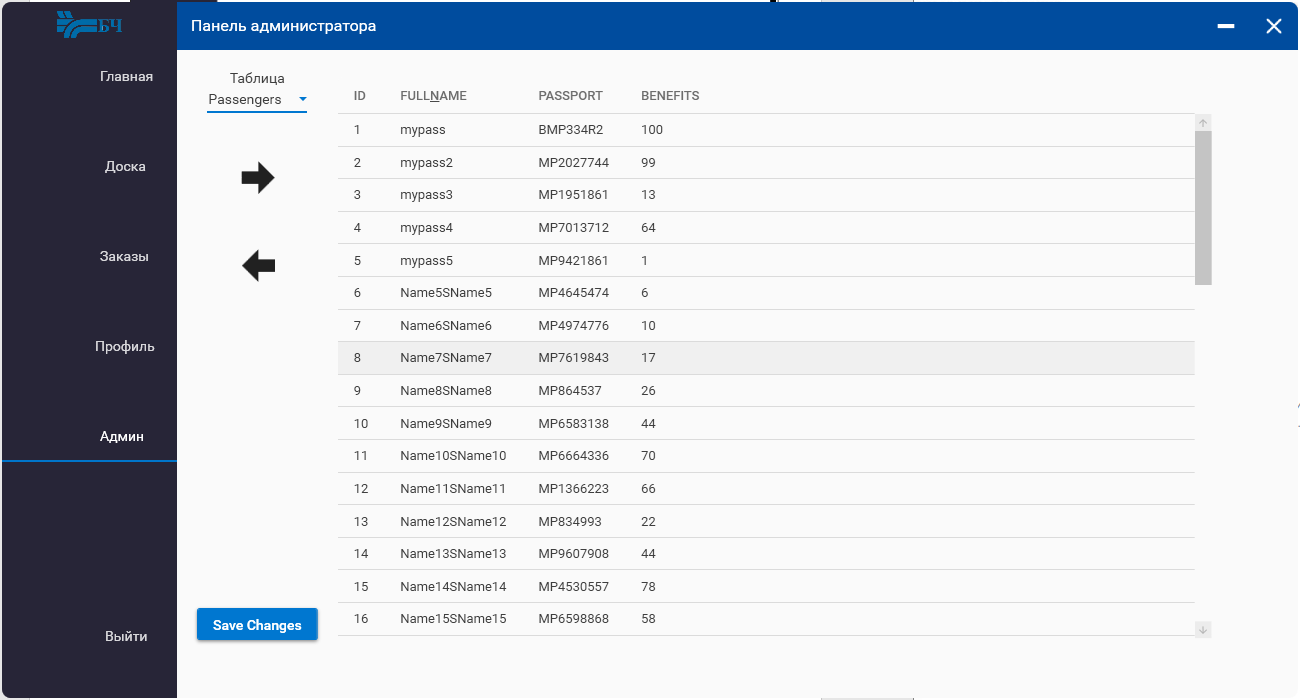


Рисунок 6.3 – Панель администратора

Также у пользователя есть возможность посмотреть доску расписаний, рисунок 6.4.

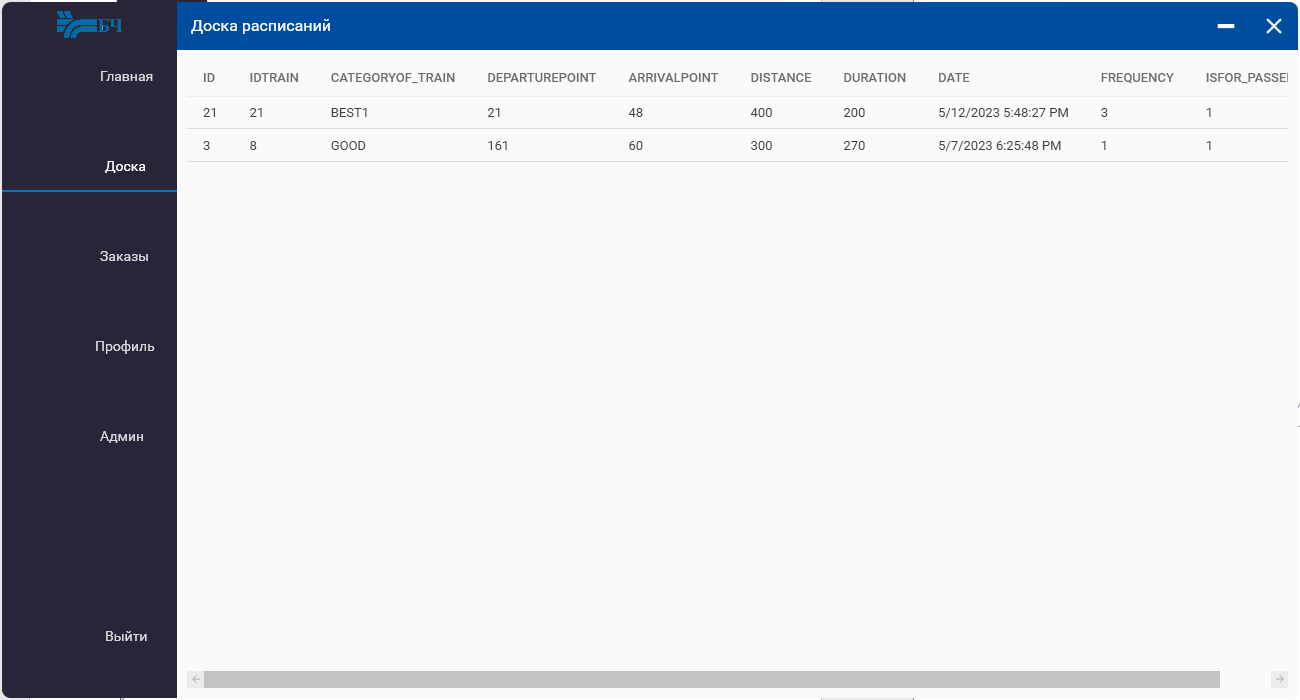


Рисунок 6.4 – Доска расписаний

Если пользователь не является администратором, то у него есть возможность сделать заказ, на странице заказа, рисунок 6.5.

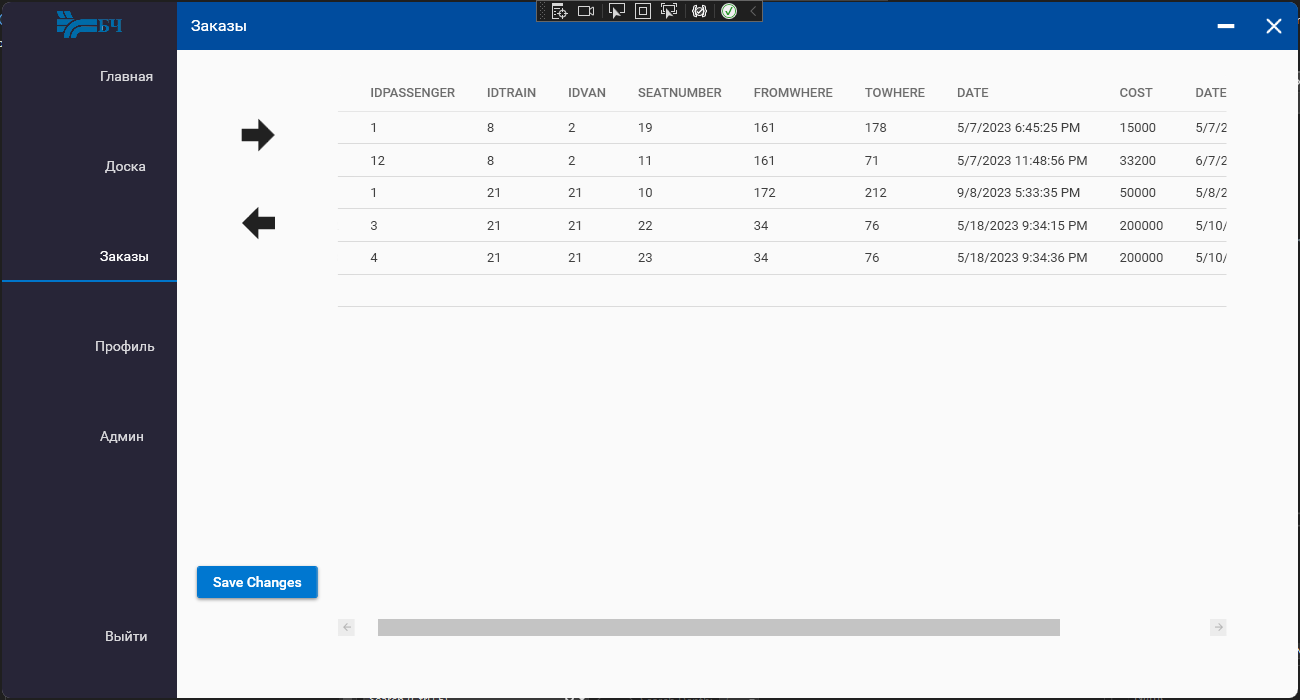


Рисунок 6.5 – Заказы

**6.3. Вывод.**

Было составлено четкое и понятное руководство пользователя для работы с приложением. Были рассмотрены базовые функции приложения, а также предоставлены скриншоты.

# **Заключение**

В данной работе была реализована база данных железнодорожного вокзала с использованием СУБД Oracle и средств мониторинга состояния. Целью работы было создание эффективной и надежной базы данных, которая позволит эффективно управлять данными о поездах, рейсах, билетах и пассажирах.

В рамках работы были поставлены следующие задачи: разработка модели базы данных, создание необходимых объектов (таблиц, представлений, индексов, ограничений), реализация процедур импорта и экспорта данных, тестирование производительности и разработка руководства пользователя.

Была разработана модель базы данных, состоящая из таблиц, связанных между собой отношениями. В модели были учтены основные аспекты работы железнодорожного вокзала, такие как рейсы, поезда, билеты, пассажиры, станции и т.д.

На основе разработанной модели были созданы необходимые объекты базы данных, такие как таблицы, представления, индексы и ограничения. Были использованы средства мониторинга состояния СУБД Oracle для оптимизации работы базы данных.

Важным этапом работы было тестирование производительности базы данных. Были проведены тесты выполнения запросов. Результаты тестирования показали полную работоспособность базы данных.

Было разработано руководство пользователя, которое содержит инструкции по использованию приложения базы данных железнодорожного вокзала. В нем описаны основные функции и возможности базы данных.

Как итог, работа позволила создать удобную в использовании и гибкую в управлении базу данных, которая может быть использована для управления данными о железнодорожных рейсах и пассажирах, а также может быть адаптирована под конкретные потребности пользователей.

**Список использованных литературных источников**

1. Жуков А. А. Мониторинг баз данных [Электронный ресурс] / Справочник Режим доступа:

СПб.: БХВ-Петербург, 2010. – 320 с. – Дата доступа 10.03.2023.

1. Статья "Development of Railway Station Management Information System Based on Internet of Things" (J. Lin, X. Zhang, H. Shao, C. Jiang, L. Yu, 2019) опубликованная в журнале "IEEE Access" – Дата доступа 11.03.2023.
2. Жуков А. А. Мониторинг баз данных [Электронный ресурс] / Справочник Режим доступа:

СПб.: БХВ-Петербург, 2010. – 320 с. – Дата доступа 10.03.2023.

1. Форум StackOverflow [Электронный ресурс] / Справочник Режим доступа:

https://ru.stackoverflow.com/ – Дата доступа 02.04.2023.

1. Оффициальный сайт Oracle [Электронный ресурс] / Справочник Режим доступа:

https://www.oracle.com/ – Дата доступа 6.04.2023.

1. Оффициальная документация Oracle [Электронный ресурс] / Справочник Режим доступа:

https://docs.oracle.com/en/ – Дата доступа 12.04.2023.

1. Информационный портал Oracle-patches [Электронный ресурс] / Справочник Режим доступа:

https://oracle-patches.com/ – Дата доступа 18.04.2023.

1. Информационный портал Oracle-dba [Электронный ресурс] / Справочник Режим доступа:

https://oracle-dba.ru/docs – Дата доступа 24.04.2023.

1. Документация по Oracle PL/SQL [Электронный ресурс] / Справочник Режим доступа:

https://oracleplsql.ru/ – Дата доступа 26.04.2023.

1. Оффициальный блог Oracle [Электронный ресурс] / Справочник Режим доступа:

https://blogs.oracle.com/ – Дата доступа 01.05.2023.

1. Статья по коллекциям в Oracle [Электронный ресурс] / Справочник Режим доступа:

https://habr.com/ru/articles/254355/ – Дата доступа 04.05.2023.