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

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА**

по дисциплине «Базы данных»

Тема «Реализация базы данных интернет-сервиса для резервирования номеров в отелях»

**Исполнитель**

студент 2 курса 3 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Юхимук А.А.

подпись, дата

**Руководитель**

Ассистент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Николайчук А.Н.

должность, учен. степень, ученое звание подпись, дата

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

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

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

дата, подпись

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Николайчук А.Н.

подпись дата инициалы и фамилия

Минск 2024

**Содержание**

Введение 4

1 Анализ требований к программному средству 5

1.1 Аналитический обзор аналогов 5

1.2 Разработка функциональных требований, определение вариантов использования 9

[2 Разработка архитектуры проекта 10](#_Toc166892277)

[2.1 Создание таблиц 10](#_Toc166892278)

[2.2 Диаграммы UML, взаимосвязь всех компонентов 11](#_Toc166892279)

2.3 Вывод 13

3. Разработка модели базы данных 14

3.1 Создание необходимых объектов 14

3.1.1 Функции 14

3.1.2 Процедуры 15

3.1.3 Представления 15

3.1.4 Триггеры 16

3.1.5 Пользовательские типы данных 17

3.1.6 Индексы 17

[3.2 Заполнение таблицы 100000 строк 18](#_Toc166892289)

3.3Описание технологии 18

3.4Вывод по разделу 19

[4. Установка, настройка и использование Oracle 12c 20](#_Toc166892292)

4.1 Создание ролей для разграничения доступа 20

4.2 Создание пользователей 21

4.3 Описание процедур экспорта и импорта 22

4.4 Вывод 23

5 Тестирование 24

5.1 Тестирование производительности базы данных 24

5.2 Проверка выполнения триггеров 25

5.3Проверка выполнения индексов 26

[5.4 Вывод по разделу 27](#_Toc166892300)

6 Сценарий Использования 28

Заключение 30

Список используемых источников 31

Приложение А 32

Приложение Б 39

Приложение В 53

# Введение

База данных является ключевым элементом любой современной организации, обеспечивая надежное хранение и управление данными. В данной курсовой работе ставилась задача разработки базы данных для системы бронирования номеров в отелях с использованием различных технологий СУБД ORACLE.

Основной целью работы было создание структурированной и надежной системы для хранения информации о пользователях, отелях, номерах, бронированиях и платежах. Для достижения этой цели использовались различные объекты базы данных, включая таблицы, триггеры, процедуры и функции.

Одной из важнейших составляющих этих технологий являются базы данных, которые обеспечивают эффективное хранение, управление и доступ к данным любого масштаба и сложности. В контексте предметной области гостиничного сервиса, базы данных становятся основой для хранения информации о бронировании номеров, гостях, услугах, платежах и других аспектах операционной деятельности.

Oracle Database, как одна из ведущих систем управления базами данных, предоставляет широкие возможности для создания и поддержки баз данных для гостиничного бизнеса. Её высокая производительность, масштабируемость и надежность делают её привлекательным выбором для крупных предприятий в этой сфере.

Анализ существующих подходов к проектированию и использованию баз данных в гостиничном бизнесе, изучение требований к функциональности и производительности, а также практическая реализация базы данных и приложения - все это является частью задач данного курсового проекта. В результате успешной реализации проекта ожидается получение эффективного инструмента управления информацией для предприятий гостиничного сервиса, способного повысить их конкурентоспособность и удовлетворить потребности клиентов в высококачественном обслуживании.

1. Анализ требований к программному средству

## **Аналитический обзор аналогов**

О strovok.ru - это сайт для поиска и бронирования размещения. Он предлагает широкий выбор отелей, апартаментов и домов. Плюсы включают удобный поиск и бронирование(пользователи могут легко указать свои предпочтения, такие как местоположение, даты пребывания и количество гостей, чтобы получить соответствующие результаты и забронировать желаемое размещение), а также возможность оставлять отзывы и оценки. Однако, отсутствуют конкретные данные и информация о ценах.

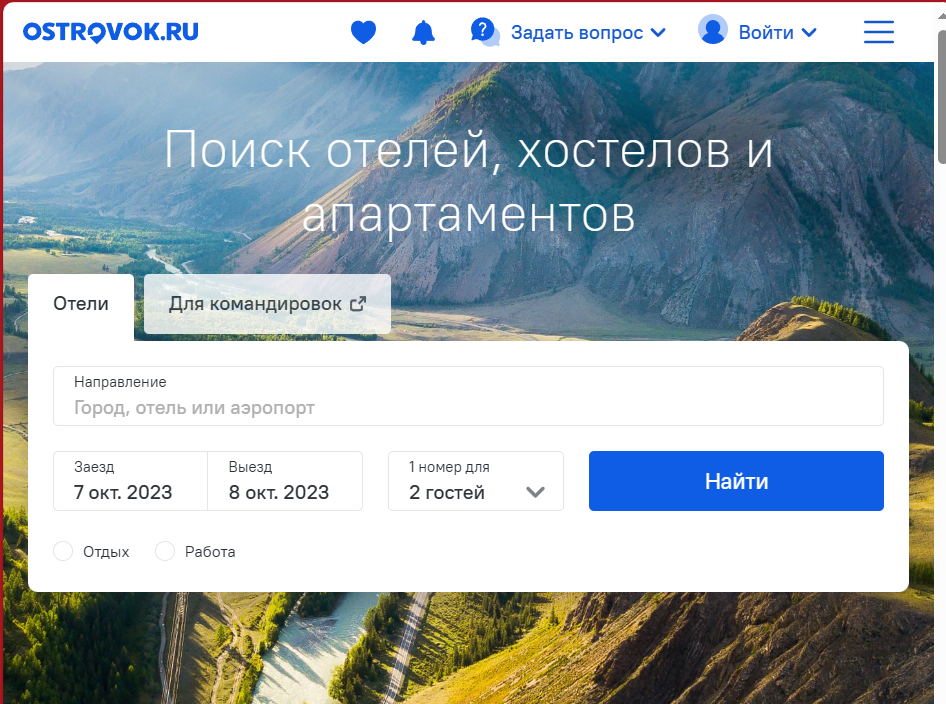


Рисунок 1.1 – Главная страница веб-сайта О strovok.ru ”

OneTwoTrip.ru - это сайт для бронирования отелей и гостиниц. Он предлагает возможность пользователям искать различные варианты размещения и бронировать их онлайн. Плюсы: Приятный для глаза дизайн. Сравнение цен (пользователи могут сравнивать цены разных отелей и гостиниц на сайте, чтобы выбрать наиболее выгодное предложение), отзывы (OneTwoTrip.ru предоставляет отзывы от других путешественников, что помогает пользователям принять информированное решение при выборе размещения)

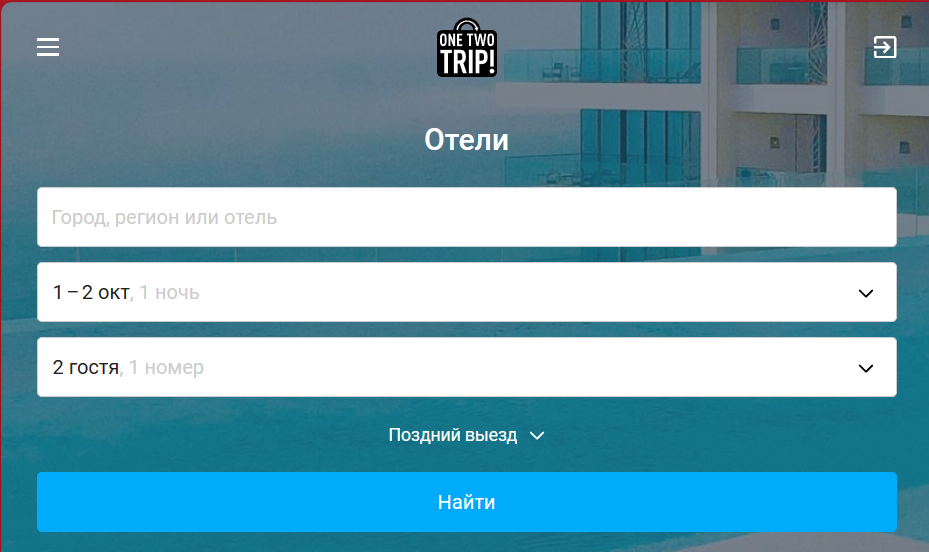


Рисунок 1.2 – Главная страница веб-сайта OneTwoTrip.ru

Booking.com - это один из крупнейших веб-сайтов для бронирования отелей, апартаментов и других типов размещения. Плюсы: Отзывы(Сайт предоставляет отзывы от реальных гостей, которые помогают пользователям принять информированное решение при выборе размещения), удобный интерфейс (Booking.com имеет интуитивно понятный интерфейс, что делает процесс поиска и бронирования быстрым и удобным). Минусы: Некоторые размещения на Booking.com могут включать скрытые сборы, которые могут быть не указаны явно на первоначальной странице поиска. Пользователям рекомендуется внимательно читать условия бронирования и уточнять все детали перед подтверждением брони.

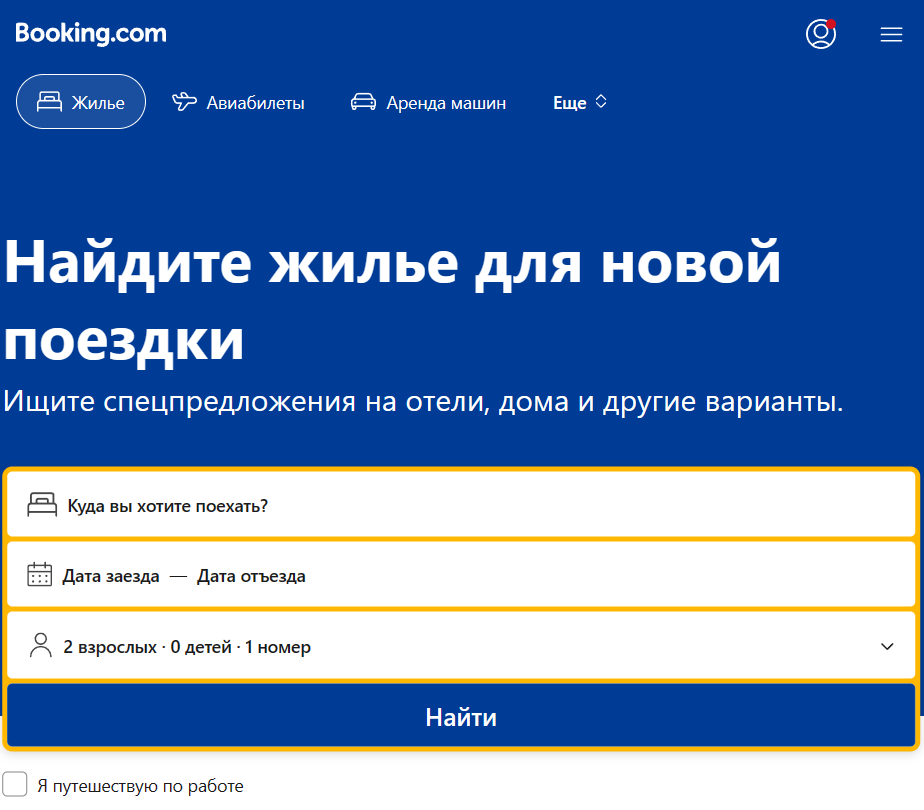


Рисунок 1.3 – Главная страница веб-сайта Booking.com

При анализе веб-сайта Radisson Hotels были обнаружены следующие основные характеристики и информация: Сайт предоставляет простой и удобный способ поиска и бронирования номеров, что позволяет быстро найти и забронировать подходящий отель. На сайте представлена информация о предстоящих мероприятиях и возможности аренды помещений для проведения свадеб, конференций и других мероприятий. Минусы: На сайте может быть ограниченное количество фильтров для поиска отелей, что может затруднить точное нахождение подходящего варианта. А также может отсутствовать точная информация о ценах на номера, что может требовать дополнительного обращения для получения актуальных данных о стоимости проживания.

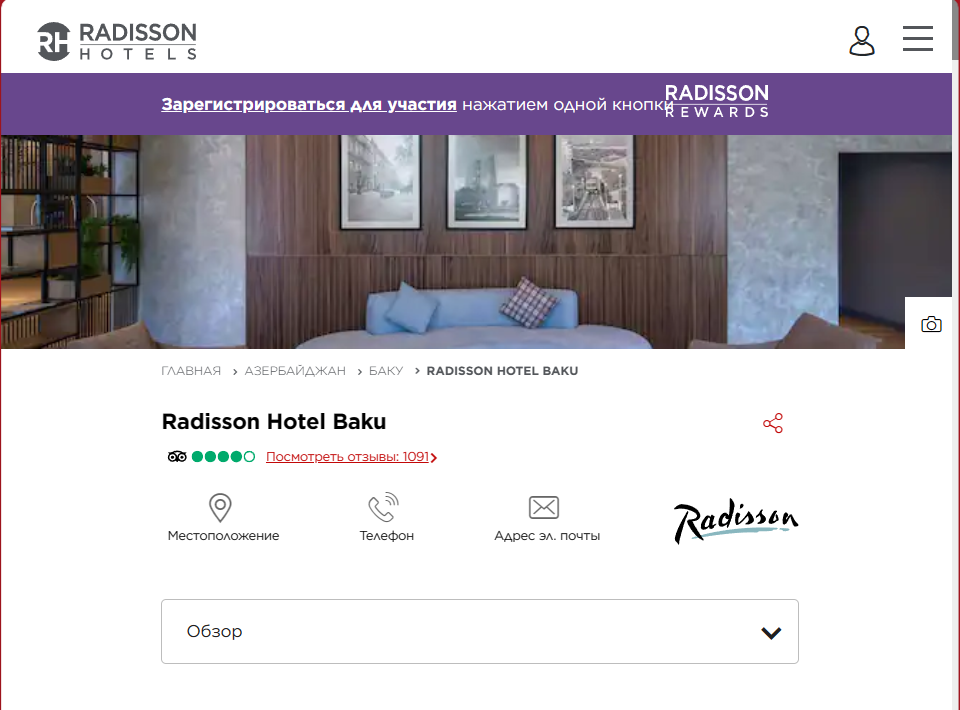


Рисунок 1.4 – Главная страница веб-сайта Radisson Hotels

В ходе аналитического разбора различных веб-сайтов для поиска и бронирования размещения были рассмотрены несколько популярных платформ: Strovok.ru, OneTwoTrip.ru, Booking.com и Radisson Hotels. Каждый из рассмотренных сайтов имеет свои преимущества и недостатки, что делает их подходящими для различных типов пользователей.

## **1.2 Разработка функциональных требований, определение вариантов использования**

Важным этапом разработки проекта является изучение требований и определение возможных вариантов использования. Требования к проекту могут быть как функциональными, то есть связанными с основными функциями проекта, так и нефункциональными, то есть связанными с качественными характеристиками проекта.

Помимо функциональных требований, важно также определить роли пользователей и их варианты использования системы. Варианты использования описывают, как пользователи будут взаимодействовать с системой в зависимости от своих ролей. Это помогает определить, какие функции должны быть доступны для каждой роли, какие данные должны быть доступны для каждой роли, а также как должна быть организована навигация в системе. Варианты использования обычно представляются в виде UML диаграмм, которые позволяют наглядно отобразить взаимодействие между пользователями и системой.

Роли пользователя — это набор прав, которые пользователь может получить в системе. Роли обычно назначаются пользователям в зависимости от их позиции, роли или ответственности в организации.

Guest (Гость) – пользователю доступна возможность регистрации и аутентификации, а также представляющий низший уровень доступа: просмотр информации об отелях; просмотр информации о номерах в отелях; проверить занятость номеров в отелях; просмотр информации о времени работы

User (авторизированный пользователь) – пользователь имеет все базовые возможности Guest, а также новый функционал, присущий роли User: искать доступные номера в отелях, просматривать информацию о номерах, включая цены, бронировать номера в выбранных отелях изменять свои данные, изменять пароль и т.д.

Adminisrator (модератор веб-сайта) – сотрудник компании который обладает такими возможностями, как: добавлять новые отели в систему, редактировать информацию о существующих отелях и устанавливать цены на номера, может создавать и удалять учетные записи сотрудников, назначать им роли и права доступа, может управлять настройками безопасности системы и контролировать доступ к конфиденциальным данным и функциям.

Чтобы описать то, как пользователи могут использовать веб сервис была разработана диаграмма использования 1.5

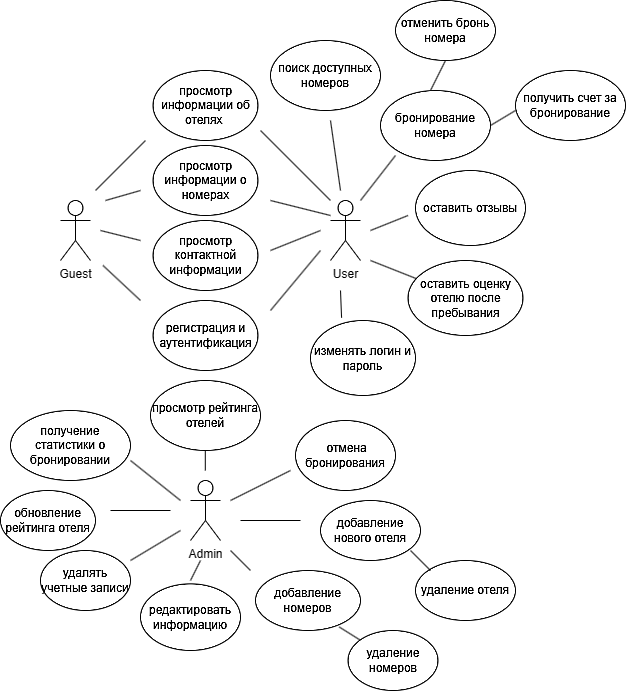


Рисунок 1.5 – UML диаграмма вариантов использования

Эти роли и варианты использования помогают ясно понять, какие функции и данные должны быть доступны для каждого типа пользователя, и как должна быть организована навигация в системе.

**1.3 Вывод**

Итого, был проведен аналитический обзор аналогов приложений для резервирования номеров в отелях и сервисов, которые уже существуют на рынке. Этот обзор позволил определить основные характеристики и функциональные возможности, которые необходимо предусмотреть в разрабатываемой системе. Также были определены функциональные требования базы данных, а также роли пользователей и варианты использования системы в зависимости от этих ролей. Была разработана UML-диаграмма, на которой отображены основные функции, которые доступны для каждой из ролей пользователей.

1. Разработка архитектуры проекта

## **2.1 Создание таблиц**

Первый этап курсового проекта – создание логически взаимосвязанных таблиц. Чтобы составить визуальную взаимосвязанную структуру базы данных, необходимо продумать, какая информация будет храниться в этих таблицах, после этого создать связи с помощью первичных и внешних ключей.

Для реализации работы базы данных было создано 6 таблиц.

Логически можно вывести 6 основных таблиц: CUSTOMERS, ROLE, PAYMENTS, RESERVATIONS, HOTELS, ROOMS.

Таблица CUSTOMERS:

ID\_CUSTOM: идентификатор клиента, тип NUMBER(10), первичный ключ.

NAME: имя клиента, тип VARCHAR2(50), не может быть пустым.

SURNAME: фамилия клиента, тип VARCHAR2(50), не может быть пустым.

EMAIL: адрес электронной почты клиента, тип VARCHAR2(50), должен быть уникальным.

PHONE\_NUM: телефон клиента, тип VARCHAR2(20), должен быть уникальным.

LOGIN: логин клиента, тип VARCHAR2(20), должен быть уникальным.

Таблица ROLE:

LOGIN: идентификатор роли, тип NUMBER(10), первичный ключ.

STATUS: статус роли, тип VARCHAR2(50), не может быть пустым.

PASSWORD: пароль роли, тип VARCHAR2(50), должен быть уникальным.

Таблица PAYMENTS:

ID\_PAY: идентификатор платежа, тип NUMBER(10), первичный ключ.

ID\_RESERV: идентификатор бронирования, тип NUMBER(10), внешний ключ, ссылается на RESERVATIONS(ID\_RESERV).

SUM\_PAY: сумма платежа, тип NUMBER(10,2), не может быть пустым.

STATUS: статус платежа, тип VARCHAR2(50).

Таблица RESERVATIONS:

ID\_RESERV: идентификатор бронирования, тип NUMBER(10), первичный ключ.

ID\_CUSTOM: идентификатор клиента, тип NUMBER(10), внешний ключ, ссылается на CUSTOMERS(ID\_CUSTOM).

STATUS: статус бронирования, тип VARCHAR2(50).

DATE\_A: дата начала бронирования, тип DATE, не может быть пустым.

DATE\_D: дата окончания бронирования, тип DATE, не может быть пустым.

Таблица HOTELS:

ID\_HOT: идентификатор отеля, тип NUMBER(10), первичный ключ.

NAME\_HOT: название отеля, тип VARCHAR2(50).

CITY: город, где расположен отель, тип VARCHAR2(50).

ADRESS: адрес отеля, тип VARCHAR2(50).

RATING: рейтинг отеля, тип VARCHAR2(50).

Таблица ROOMS:

ID\_NUM: идентификатор номера, тип NUMBER(10), первичный ключ.

ID\_HOT: идентификатор отеля, к которому привязан номер, тип NUMBER(10), внешний ключ, ссылается на HOTELS(ID\_HOT).

TYPE: тип номера, тип VARCHAR2(50).

PRICE: цена за проживание в номере, тип NUMBER(10,2).

AVAILABILITY: доступность номера, тип VARCHAR2(50).

NUM\_OF\_ROOM: количество номеров.

## **2.2 Диаграммы UML, взаимосвязь всех компонентов**

Диаграмма базы данных таблиц (или схема базы данных) - это визуальное представление структуры базы данных, включающее таблицы, их атрибуты и связи между ними. Она предоставляет обзор структуры данных и их отношений, что облегчает понимание организации базы данных.

Диаграмма базы данных будет представлена на рисунке 2.1.

Все таблицы приведены к 3NF, а также соблюдают все базовые требования для данного типа таблиц

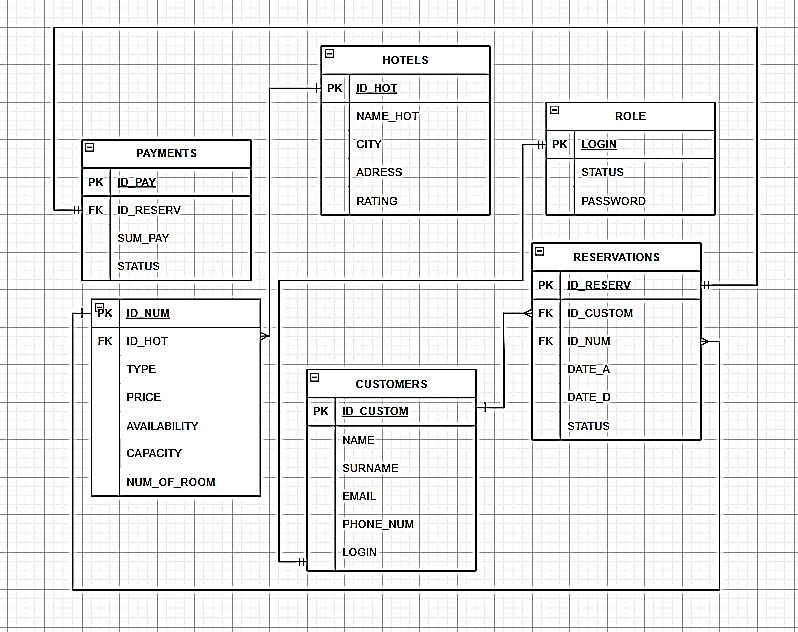


Рисунок 2.1 – Диаграмма базы данных

Роли пользователя — это набор прав, которые пользователь может получить в системе. Роли обычно назначаются пользователям в зависимости от их позиции, роли или ответственности в организации. В зависимости от роли пользователя, он может иметь доступ к различным функциям системы. В данном проекте роли пользователей будут следующими:

Admin;

User;

Guest;

На основе предоставленного списка ролей необходимо построить варианты использование.

Варианты использование изображена на рисунке 2.2.

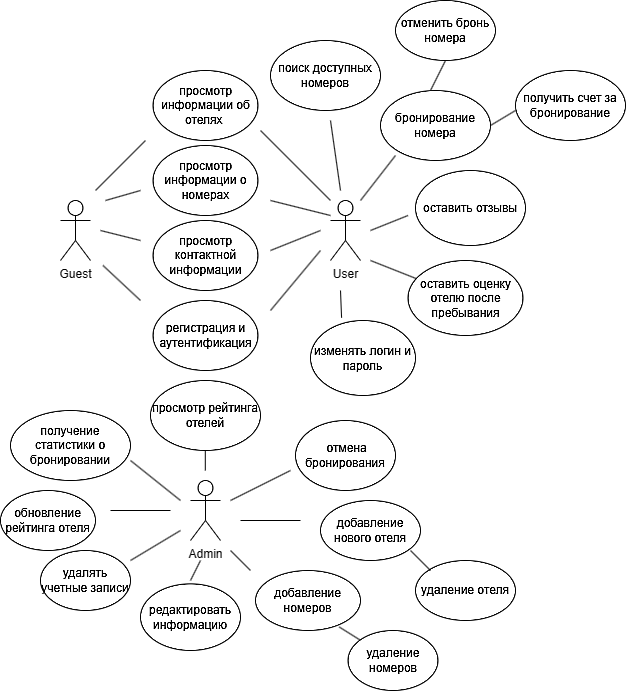


Рисунок 1.5 – UML диаграмма вариантов использования

Эти роли и варианты использования помогают ясно понять, какие функции и данные должны быть доступны для каждого типа пользователя, и как должна быть организована навигация в системе.

## **2.3 Вывод**

Разработка архитектуры проекта для сервиса бронирования номеров в отелях является ключевым шагом для определения структуры и функциональности приложения. Это позволяет определить компоненты, которые будут использоваться в приложении, и описать их взаимодействие между собой.

В сервисе для бронирования номеров в отелях основными информационными объектами могут быть сами номера, пользователи, бронирования, отели, услуги и отзывы.

В сервисе для бронирования номеров в отелях основными информационными объектами могут быть сами номера, пользователи, бронирования, отели, услуги и отзывы. Номера содержат информацию о типе номера, доступных удобствах, цене и других характеристиках размещения. Пользователи представляют собой зарегистрированных клиентов, которые могут просматривать доступные номера, оформлять бронирования, оставлять отзывы и оценивать сервис. Бронирования содержат информацию о дате заезда и выезда, количестве гостей и связанной информации об оплате. Отели предоставляют информацию о расположении, уровне обслуживания и доступных услугах. Услуги могут включать завтраки, интернет, тренажерные залы и т. д. Отзывы позволяют клиентам делиться своими впечатлениями о проживании и оценивать качество обслуживания.

В целом, архитектура проекта для сервиса бронирования номеров в отелях помогает организовать структуру приложения, определить его функциональность и взаимодействие компонентов, обеспечивая удобство использования и эффективную обработку данных для пользователей.

Архитектура проекта для сервиса бронирования номеров в отелях помогает организовать структуру приложения, определить его функциональность и взаимодействие компонентов. Это обеспечивает удобство использования и эффективную обработку данных для пользователей. Правильная организация архитектуры позволяет также учитывать масштабируемость и гибкость системы, чтобы она могла легко адаптироваться к изменяющимся требованиям и увеличивающемуся количеству пользователей. В целом, четко спланированная архитектура является основой для успешной реализации и последующей эксплуатации сервиса, способствуя повышению удовлетворенности пользователей и эффективности работы системы.

3. 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.

**3.1 Создание необходимых объектов**

Для реализации базы данных сервиса бронирования номеров в отелях необходимо создать набор объектов базы данных, которые включают таблицы, представления, индексы, ограничения и хранимые процедуры. Все эти объекты должны быть разработаны с учетом требований к производительности, безопасности и надежности.

В проектируемой базе данных были созданы 6 таблиц, описание которых приведено выше. Для взаимодействия с этими таблицами были разработаны такие объекты базы данных, как функции, процедуры, представления и триггеры, а также пользовательский тип данных и индексы:

**3.1.1 Функции**

В Oracle функция является программным объектом, который выполняет определённые вычисления и возвращает одно значение. Функции могут быть использованы в SQL-запросах, PL/SQL-блоках, процедурах и других функциях. Они часто применяются для выполнения повторяющихся задач, таких как вычисления, преобразования данных или получение значения из базы данных.

Пример функции, которая проверяет на валидность email– листинг 3.1

|  |
| --- |
| CREATE OR REPLACE FUNCTION is\_valid\_email(email IN VARCHAR2)  RRETURN BOOLEAN  IS  BEGIN  -- Проверяем, соответствует ли email заданному формату  IF REGEXP\_LIKE(email, '^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$')  T WHEN  RETURN TRUE;  ELSE  RETURN FALSE;  END IF;  END; |

Листинг 3.1 – Функция проверки на валидность

В сумме, при разработке курсового проекта, было создано 3 функции:

1. Функция, которая проверяет на валидность email
2. Получение списка всех гостиниц в указанном городе
3. Получение списка бронирований в указанной гостинице за определенный период времени

**3.1.2 Процедуры**

Процедура в Oracle является программным объектом, который выполняет одну или несколько задач, таких как операции с данными, вычисления, или вызовы других процедур и функций. В отличие от функций, процедуры не возвращают значение напрямую, но могут возвращать данные через выходные параметры или изменять состояния базы данных. Процедуры часто используются для выполнения более сложных операций, таких как транзакции, валидация данных и автоматизация задач.

Пример определения процедуры, которая реализует получение списка всех номеров в конкретном отеле билете – листинг 3.2.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE GET\_ROOMS\_BY\_HOTEL(  p\_id\_hot IN NUMBER,  p\_cursor OUT SYS\_REFCURSOR  )  AS  BEGIN  OPEN p\_cursor FOR  SELECT ID\_NUM, TYPE, PRICE, AVAILABILITY, NUM\_OF\_ROOM  FROM ROOMS  WHERE ID\_HOT = p\_id\_hot;  END; |

Листинг 3.2 – Процедура получения списка номеров в отеле

В сумме, при разработке курсового проекта, было создано 32 процедуры. Весь перечень созданных процедур представлен в Приложении Б.

**3.1.3 Представления**

Представление (view) в Oracle – это виртуальная таблица, которая представляет собой результат выполнения запроса SQL. Оно не содержит данных непосредственно, а предоставляет динамически обновляемое представление данных из одной или нескольких базовых таблиц. Представления используются для упрощения сложных запросов, повышения безопасности данных и улучшения удобства работы с базой данных.

Пример определения представления, которое объединяет информацию о бронированиях– листинг 3.3.

|  |
| --- |
| CRCREATE OR REPLACE VIEW RESERVATION\_SUMMARY AS  SESELECT  r.ID\_RESERV,  c.NAME AS CUSTOMER\_NAME,  c.SURNAME AS CUSTOMER\_SURNAME,  c.EMAIL AS CUSTOMER\_EMAIL,  h.NAME\_HOT AS HOTEL\_NAME,  h.CITY AS HOTEL\_CITY,  rm.TYPE AS ROOM\_TYPE,  rm.PRICE AS ROOM\_PRICE,  r.STATUS AS RESERVATION\_STATUS,  r.DATE\_A AS DATE\_ARRIVAL,  r.DATE\_D AS DATE\_DEPARTURE  FROM  RESERVATIONS r  JOIN CUSTOMERS c ON r.ID\_CUSTOM = c.ID\_CUSTOM  JOIN ROOMS rm ON r.ID\_NUM = rm.ID\_NUM  JOIN HOTELS h ON rm.ID\_HOT = h.ID\_HOT;    SELECT \* FROM RESERVATION\_SUMMARY WHERE HOTEL\_CITY = 'Дубай' AND RESERVATION\_STATUS = 'Подтверждено'; |

Листинг 3.3 – Представление RESERVATION\_SUMMARY

В этом коде создается представление (view) RESERVATION\_SUMMARY, которое объединяет данные из нескольких таблиц (RESERVATIONS, CUSTOMERS, ROOMS, HOTELS) и предоставляет сводную информацию о бронированиях. В представлении выбираются такие поля, как ID бронирования, имя, фамилия и email клиента, название и город отеля, тип и цена номера, статус бронирования, дата прибытия и отъезда.

**3.1.4 Триггеры**

Триггеры (triggers) в Oracle – это хранимые программы, которые автоматически выполняются (срабатывают) при возникновении определенных событий в базе данных. Эти события могут быть связаны с операциями вставки, обновления или удаления данных в таблицах или представлениях. Триггеры часто используются для обеспечения целостности данных, выполнения автоматических проверок, логирования изменений и реализации сложной бизнес-логики.

Пример определения триггера для автоматического изменения статуса брони номера при оплате– листинг 3.5.

|  |
| --- |
| CREATE OR REPLACE TRIGGER payment\_trigger  AFTER UPDATE OF status ON payments  FOR EACH ROW  BEGIN  IF :OLD.status <> :NEW.status THEN  IF :NEW.status = 'Оплачено' THEN  UPDATE reservations  SET status = 'Подтверждено'  WHERE id\_reserv = :NEW.id\_reserv;  ELSIF :NEW.status = 'Не оплачено' THEN  UPDATE reservations  SET status = 'Ожидает подтверждения'  WHERE id\_reserv = :NEW.id\_reserv;  END IF;  END IF;  END; |

Листинг 3.5 – Триггер PAYMENT\_TRIGGER

Всего было разработано 3 триггера для решения следующих задач:

1. Для автоматического изменения статуса брони номера при оплате
2. Для автоматического изменения доступности номера при создании брони
3. Для проверки доступности номера при создании новой брони

**3.1.5 Пользовательские типы данных**

Пользовательский тип данных (User-Defined Type, UDT) в Oracle – это тип данных, который создается пользователем для удовлетворения специфических потребностей приложения. Пользовательские типы данных позволяют разработчикам создавать сложные структуры данных, которые не могут быть полностью описаны с использованием встроенных типов данных Oracle. В базе данных для сервиса бронирования номеров в отелях можно создать пользовательский тип данных для представления информации о бронированиях.

|  |
| --- |
| TYPE t\_name\_arr IS TABLE OF VARCHAR2(50);  TYPE t\_city\_arr IS TABLE OF VARCHAR2(50);  TYPE t\_address\_arr IS TABLE OF VARCHAR2(50);  TYPE t\_rating\_arr IS TABLE OF VARCHAR2(50); |

Листинг 3.6 – Пользовательские типы данных

Использование этих типов данных позволяет удобно работать с коллекциями строковых значений в Oracle, например, передавать их в процедуры или функции, выполнять операции с элементами массива и т.д.

**3.1.6 Индексы**

Добавление индексов к вашей базе данных поможет улучшить производительность запросов, особенно тех, которые часто используют условия поиска, соединения и сортировки.

Cозданные индексы представлены на листинге ниже.

|  |
| --- |
| -- Таблица PAYMENTS  CREATE INDEX idx\_payments\_id\_reserv ON PAYMENTS(ID\_RESERV);  -- Таблица RESERVATIONS  CREATE INDEX idx\_reservations\_id\_custom ON RESERVATIONS(ID\_CUSTOM);  CREATE INDEX idx\_reservations\_id\_num ON RESERVATIONS(ID\_NUM);  -- Таблица HOTELS  CREATE INDEX idx\_hotels\_city ON HOTELS(CITY);  CREATE INDEX idx\_hotels\_rating ON HOTELS(RATING);  -- Таблица ROOMS  CREATE INDEX idx\_rooms\_id\_hot ON ROOMS(ID\_HOT);  CREATE INDEX idx\_rooms\_availability ON ROOMS(AVAILABILITY); |

Листинг 3.7 – Созданные в проекте индексы

Использование индексов может значительно повысить производительность запросов к таблицам, особенно при работе с большими объемами данных.

**3.2 Заполнение таблицы 100000 строк**

Для заполнения таблицы 100000 строк были разработаны функции populate\_hotels\_table, populate\_rooms\_table, populate\_customers\_table, populate\_reservations\_table, populate\_payments\_table, populate\_role\_table.

Все функций представлен на листинге в приложении

* 1. Описание технологии

MyOra - это передовой инструмент мониторинга баз данных Oracle с самым быстрым временем отклика и мгновенным отображением базы данных Oracle. Этот инструмент может помочь в выполнении таких функций, как мониторинг производительности в режиме реального времени с использованием графиков, гистограмм и круговых диаграмм, просмотр активности БД в режиме реального времени с помощью шести интерактивных графиков,несколько окон мониторинга производительности для одной и той же или разных баз данных, а также возможность отображения системных ожиданий с помощью круговых диаграмм(1).

Он очень удобен в управлении:

* Разрешает несколько подключений к одной или разной базе данных.
* Запускает несколько системных мониторов для одной или разных баз данных.
* Пользователи могут открыть несколько обозревателей схем

На следующих рисунках представлен интерфейс приложения MyOra.

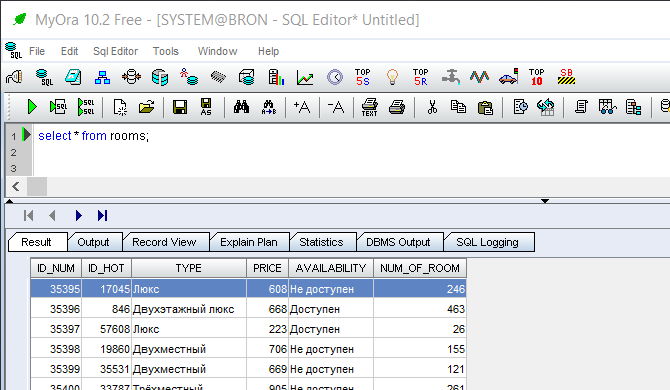


Рисунок 3.9 – Интерфейс приложения MyOra

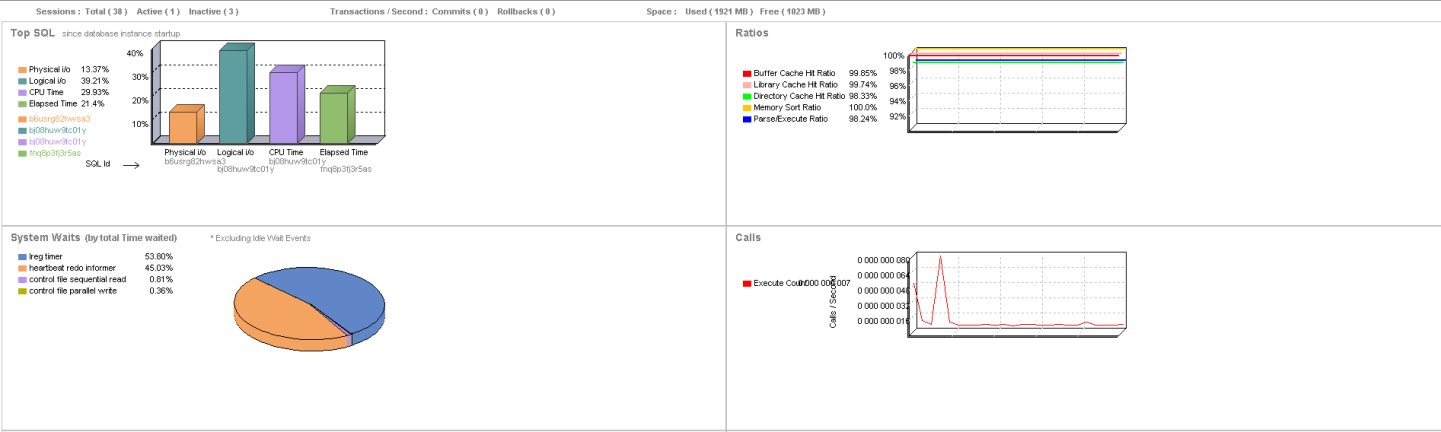


Рисунок 3.9 – Диаграммы MyOra базы данных Bron

MyOra может предоставлять информацию о времени выполнения запросов, нагрузке на сервер, использовании ресурсов и других метриках производительности баз данных Oracle. MyOra может помочь в анализе и оптимизации SQL-запросов, предоставляя информацию о плане выполнения запросов, индексах и других аспектах запросов.

* 1. Вывод по разделу

В данном разделе было описано и выполнено создание объектов для базы данных авиакомпании, таких как табличное пространство, профили безопасности, роли и пользователи, таблицы, триггеры, хранимые процедуры. Также было определено взаимодействие всех этих элементов между собой.

1. Установка, настройка и использование Oracle 12c

Oracle Database 12c - это одна из самых передовых и мощных реляционных баз данных на сегодняшний день, предоставляемая корпорацией Oracle. Это версия базы данных, которая предлагает ряд инновационных функций и улучшений, направленных на повышение производительности, масштабируемости, безопасности и управляемости баз данных.

Oracle Database поддерживает различные операционные системы и аппаратные платформы, что делает ее универсальным решением для любых задач, связанных с управлением данными. Она также обладает мощными инструментами для администрирования и мониторинга баз данных, что делает ее идеальным выбором для профессионалов в области баз данных.

После установки Oracle DB была создана база данных с названием BRON, которая будет использоваться в дальнейшем для хранения данных.

4.1 Создание ролей для разграничения доступа

Роли в Oracle позволяют управлять безопасностью базы данных, ограничивая доступ к конфиденциальным данным и защищая данные от несанкционированного доступа.

Роль (Role) - это набор привилегий, которые могут быть назначены пользователям или другим ролям. Роли представляют собой удобный механизм управления доступом, который позволяет администраторам базы данных определять группы пользователей с общими привилегиями и назначать им эти привилегии.

Будут созданы три роли для разграничения доступа к базе данных:

rl\_user

rl\_admin;

Роль rl\_admin имеет права на выполнение различных функций и процедур, связанных с управлением пользователями, номера и отелями в базе данных.

|  |
| --- |
| GRANT ALL PRIVILEGES ON TO rl\_admin;  GRANT CONNECT TO rl\_admin;  GRANT CREATE SESSION TO rl\_admin;  GRANT CREATE ANY TABLE TO rl\_admin;  GRANT CREATE VIEW TO rl\_admin;  GRANT CREATE TRIGGER TO rl\_admin;  GRANT SELECT ANY TABLE TO rl\_admin;  GRANT CREATE PROCEDURE TO rl\_admin;  GRANT CREATE TABLESPACE TO rl\_admin;  GRANT DROP ANY TABLE TO rl\_admin;  GRANT DROP ANY VIEW TO rl\_admin;  GRANT EXECUTE ON ADD\_HOTEL rl\_admin;  GRANT EXECUTE ON DELETE\_HOTEL rl\_admin;  GRANT EXECUTE ON ADD\_ROOM rl\_admin;  GRANT EXECUTE ON DELETE\_ROOM rl\_admin;  GRANT EXECUTE ON UPDATE\_HOTEL\_RATING rl\_admin;  GRANT EXECUTE ON GET\_RESERVATION\_STATISTICS rl\_admin;  GRANT EXECUTE ON VIEW\_PAYMENTS rl\_admin;  GRANT EXECUTE ON CONFIRM\_PAYMENT rl\_admin;  GRANT EXECUTE ON VIEW\_RESERVATIONS rl\_admin;  GRANT EXECUTE ON CONFIRM\_RESERVATION rl\_admin;  GRANT EXECUTE ON CANCEL\_RESERVATION rl\_admin;  GRANT EXECUTE ON GET\_ALL\_USERS rl\_admin; |

Листинг 4.1 – Привилегии, выданные роли rl\_admin

Роль rl\_user имеет права на выполнение различных функций и процедур, связанных с получением информации об отеле, номере и т.д.

|  |
| --- |
| GRANT SELECT ON HOTELS TO rl\_user;  GRANT EXECUTE ON BOOK\_ROOM TO rl\_user;  GRANT EXECUTE ON CANCEL\_RESERVATION TO rl\_user;  GRANT EXECUTE ON GET\_MY\_RESERVATIONS TO rl\_user;  GRANT EXECUTE ON GET\_AVAILABLE\_ROOMS TO rl\_user;  GRANT EXECUTE ON GET\_RESERVATION\_BILL TO rl\_user;  GRANT EXECUTE ON GET\_ROOMS\_BY\_HOTEL TO rl\_user;  GRANT EXECUTE ON GET\_AVAILABLE\_ROOMS\_IN\_HOTEL TO rl\_user; |

Листинг 4.2 – Привилегии, выданные роли rl\_user

Таким образом, были созданы три роли для ограничения доступа к базе данных: rl\_user, rl\_admin. Каждая роль имеет определенный набор привилегий, который позволяет пользователю выполнять определенные функции в базе данных.

4.2 Создание пользователей

Пользователь базы данных – это физическое лицо, которое имеет доступ к БД и пользуется услугами системы для получения информации.

При проектировании базы данных было использовано 2 пользователя:

|  |
| --- |
| CREATE USER USER\_DB  IDENTIFIED BY userdb  DEFAULT TABLESPACE TS\_COURSE  QUOTA UNLIMITED ON TS\_COURSE  ACCOUNT UNLOCK; |

Листинг 4.3 – Скрипт создания пользователя USER\_DB

|  |
| --- |
| CREATE USER ADMIN\_DB  IDENTIFIED BY admindb  DEFAULT TABLESPACE TS\_COURSE  QUOTA UNLIMITED ON TS\_COURSE  ACCOUNT UNLOCK; |

Листинг 4.4 – Скрипт создания пользователя ADMIN\_DB

Скрипт выдачи прав пользователям на представления, процедуры и функции представлен в приложении Г.

4.3 Описание процедур экспорта и импорта

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

Формат JSON (JavaScript Object Notation) – это легковесный формат обмена данными, который используется для представления структурированных данных в формате пар "имя/значение". Он широко применяется в веб-приложениях, мобильных приложениях и базах данных. Использование JSON для импорта и экспорта данных обеспечивает удобство и гибкость при обмене данными между различными базами данных или веб-приложениями.

Процедура export\_role\_to\_json предназначена для экспорта данных из таблицы role в формат JSON и сохранения их в файл. Эта процедура открывает файл в директории DATA\_PUMP\_DIR, итерируется по данным из таблицы role, формирует для каждой записи строку в формате JSON и записывает ее в файл. В конце процедуры файл закрывается. В случае возникновения ошибки файл закрывается и исключение передается дальше.

|  |
| --- |
| CREATE OR REPLACE DIRECTORY DATA\_PUMP\_DIR AS 'D:\KURSORACLE\ARHIEVE';  GRANT READ, WRITE ON DIRECTORY DATA\_PUMP\_DIR TO PUBLIC;  CREATE OR REPLACE PROCEDURE export\_hotels\_to\_json  (  p\_file\_name IN VARCHAR2  )  AS  l\_file UTL\_FILE.FILE\_TYPE;  l\_buffer VARCHAR2(32767);  l\_first\_row BOOLEAN := TRUE;  CURSOR c\_hotels IS  SELECT id\_hot, name\_hot, city, adress, rating  FROM hotels;  BEGIN  l\_file := UTL\_FILE.FOPEN('DATA\_PUMP\_DIR', p\_file\_name, 'w');  UTL\_FILE.PUT\_LINE(l\_file, '[');  FOR rec IN c\_hotels  LOOP  IF l\_first\_row THEN  l\_first\_row := FALSE;  ELSE  UTL\_FILE.PUT\_LINE(l\_file, ',');  END IF;    l\_buffer := ' {' ||  '"id\_hot": ' || rec.id\_hot || ',' ||  '"name\_hot": "' || rec.name\_hot || '",' ||  '"city": "' || rec.city || '",' ||  '"adress": "' || rec.adress || '",' ||  '"rating": "' || rec.rating || '"' ||  '}';  UTL\_FILE.PUT\_LINE(l\_file, l\_buffer);  END LOOP;  UTL\_FILE.PUT\_LINE(l\_file, ']');  UTL\_FILE.FCLOSE(l\_file);  EXCEPTION  WHEN OTHERS THEN  IF UTL\_FILE.IS\_OPEN(l\_file) THEN  UTL\_FILE.FCLOSE(l\_file);  END IF;  RAISE;  END;  BEGIN  export\_hotels\_to\_json('hotels.json');  END; |

Листинг 4.5 – Скрипт процедуры экспорта из базы данных

Процедура export\_role\_to\_json предназначена для экспорта данных из таблицы role в формат JSON и сохранения их в файл. Она открывает файл в директории DATA\_PUMP\_DIR, итерируется по данным из таблицы role, формирует для каждой записи строку в формате JSON и записывает ее в файл. В случае возникновения ошибки, файл закрывается и исключение передается дальше.

4.4 Вывод

В данном разделе были рассмотрены основные этапы установки, настройки и использования Oracle DB. Были описаны процедуры создания таблиц, ролей для разграничения доступа к базе данных и пакетов процедур для выполнения различных операций с данными, а также приведены их листинги.

Таким образом, Oracle DB представляет собой надежную и мощную систему управления базами данных, способную эффективно обрабатывать и хранить большие объемы данных. Правильная конфигурация и оптимизация Oracle DB, а также умелое использование инструментов для оптимизации запросов, существенно повышают производительность и эффективность работы с базой данных.

**5 Тестирование**

**5.1 Тестирование производительности базы данных**

Тестирование производительности является важным этапом разработки, поскольку позволяет определить, насколько хорошо база данных может обрабатывать запросы и как быстро она может возвращать результаты.

У нас уже созданы необходимые объекты базы данных, а также таблицы заполнены необходимым количеством данных – рисунок 5.1. Теперь необходимо протестировать, что всё работает, как задумано.

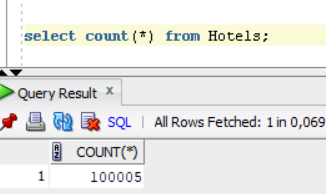


Рисунок 5.1 – Количество данных в таблице

Для клиентской части были написаны процедуры работы с данными из базы данных. А именно добавления, изменения и удаления данных. Например процедуры получения информации по номеру комнаты:

1. get\_room\_type - Вывод типа комнаты по номеру комнаты и ID отеля.
2. get\_room\_price - Вывод цены комнаты по номеру комнаты и ID отеля.
3. get\_room\_availability - Вывод доступности комнаты по номеру комнаты и ID отеля.

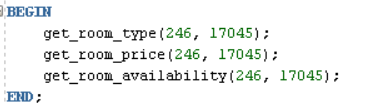


Рисунок 5.2 – Вызов процедур получения информации комнаты

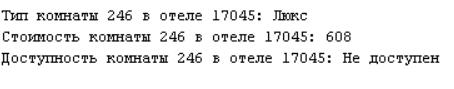


Рисунок 5.3 – Результат процедуры получения информации комнаты

А также были запущены и проверены все процедуры

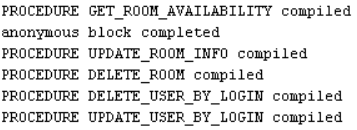


Рисунок 5.4 – процедуры получения информации комнаты

В ходе поверки выяснилось, что все процедуры выполнены успешно

**5.2 Проверка выполнения триггеров**

Триггер payment\_trigger предназначен для автоматического изменения статуса бронирования номера при изменении статуса платежа. Когда статус платежа обновляется, триггер проверяет новое значение статуса и соответственно обновляет статус связанного бронирования в таблице reservations.

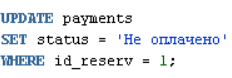


Рисунок 5.5 – вызов триггера payment\_trigger



Рисунок 5.6 – вызов произведен успешно



Рисунок 5.7 – вывод таблицы reservations (status=”Ожидает подтверждения”)

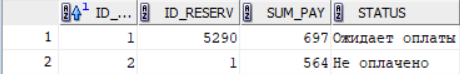


Рисунок 5.8 – вывод таблицы payments (status =” Ожидает оплаты”)

В ходе поверки выяснилось, что все триггеры выполнены успешно

* 1. **Проверка выполнения индексов**

Для проверки индексов был написан скрипт для измерения статистики выполнения до применения индексов idx\_hotels\_city и idx\_hotels\_rating и после их применения.

|  |
| --- |
| EXPLAIN PLAN FOR  SELECT \* FROM HOTELS  WHERE CITY = 'New York'  ORDER BY RATING;  SELECT \* FROM TABLE(DBMS\_XPLAN.DISPLAY);  -- Таблица HOTELS  CREATE INDEX idx\_hotels\_city ON HOTELS(CITY);  CREATE INDEX idx\_hotels\_rating ON HOTELS(RATING); |

Листинг 5.9 – Скрипт плана выполнения запроса

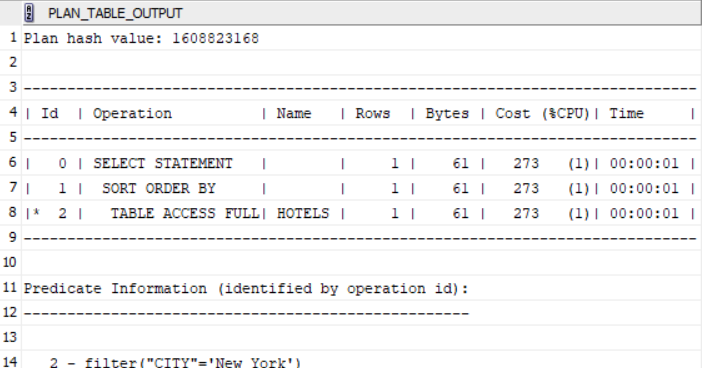
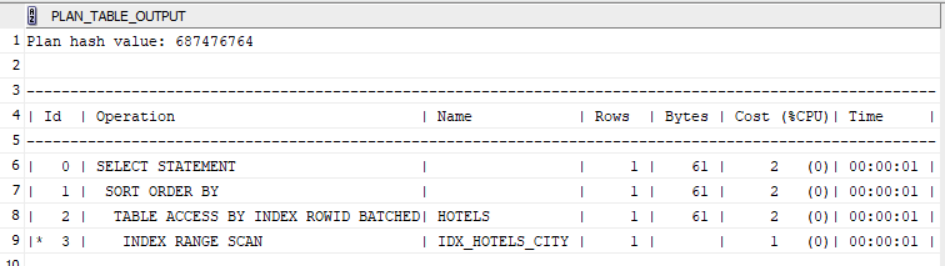


Рисунок 5.10 – Результат выполнения запросов до индексов

Рисунок 5.11 – Стоимость запроса после индексов

После создания индексов запрос к этой таблице показал значительное улучшение производительности. Результаты тестирования можно увидеть на рисунке 5.11

Создание индексов на полях, по которым выполняются частые запросы, может существенно повысить производительность базы данных.

5.4 Вывод по разделу

В данном разделе были протестированы процедуры, индексы и триггеры. Для проведения тестирования были выбраны таблицы HOTELS и PAYMENTS.

Процедуры предоставляют удобный способ управления данными и выполнения операций в базе данных. Создание индексов на полях, по которым выполняются частые запросы, может значительно повысить производительность базы данных, что особенно важно при работе с большими объемами данных. Триггеры могут быть полезны для обеспечения целостности данных и принятия автоматических действий в ответ на изменения данных.

Результаты тестирования помогают выявить проблемы и улучшить производительность, надежность и функциональность базы данных. По итогам тестирования можно сказать, что все работает корректно и база данных надёжна.

**6** **Сценарий использования**

При открытии сервиса по бронированию номеров в отелях пользователю представлен гостевой режим, в котором можно просматривать общую информацию об отелях, номерах и их доступности. Используя процедуры GET\_ALL\_HOTELS, GET\_ROOMS\_BY\_HOTEL и GET\_AVAILABLE\_ROOMS\_IN\_HOTEL, гость может получить списки всех отелей в системе, номеров в конкретном отеле и информацию о свободных номерах на определенные даты.

Для получения более детальной информации, такой как рейтинг отеля в определенном городе, тип номера, его цена и доступность, гость может воспользоваться процедурами get\_hotel\_rating\_in\_city, get\_room\_type, get\_room\_price и get\_room\_availability.

Однако, для бронирования номеров и получения доступа к дополнительным функциям, пользователю необходимо пройти регистрацию и авторизацию в системе. Также предусмотрена проверка на корректность электронной почты (is\_valid\_email).

После успешной регистрации пользователь может перейти на страницу входа в систему, ввести свой логин и пароль, что запустит процедуру CREATE\_USER\_WITH\_ROLE. При успешной авторизации пользователь получает роль USER, и ему становится доступен расширенный функционал системы.

В режиме пользователя доступны такие функции, как бронирование номеров (BOOK\_ROOM), отмена бронирований (CANCEL\_RESERVATION), получение счетов за свои бронирования (GET\_RESERVATION\_BILL) и просмотр собственных бронирований (GET\_MY\_RESERVATIONS). Также пользователь может просматривать свои платежи (VIEW\_PAYMENTS) и подтверждать их (CONFIRM\_PAYMENT). Когда пользователь оплатит свою бронь вызовется триггер (PAYMENT\_TRIGGER), который изменит статус брони при оплате. Как только бронь подтвердится будет вызван триггер (RESERVATION\_TRIGGER) и доступность номера изменится на “Недоступен”.

Пользователь может зайти в личный кабинет и изменить свои личные данные, например, дату рождения, что запустит процедуру UPDATE\_USER\_BY\_LOGIN. Если пользователь решил поменять пароль от учетной записи, сработает процедура UPDATE\_USER\_BY\_LOGIN.

Для администраторов системы доступен расширенный набор функций по управлению отелями, номерами, пользователями и бронированиями. Они могут изменять информацию о номерах (update\_room\_info), получать данные о доступных отелях и номерах (GET\_AVAILABLE\_ROOMS), удалять номера (delete\_room) и пользователей (delete\_user\_by\_login), добавлять новые отели (ADD\_HOTEL) и номера (ADD\_ROOM), удалять отели (DELETE\_HOTEL) и номера из отелей (DELETE\_ROOM), обновлять рейтинги отелей (UPDATE\_HOTEL\_RATING) и получать статистику по бронированиям (GET\_RESERVATION\_STATISTICS). Кроме того, администраторы могут просматривать и управлять бронированиями (VIEW\_RESERVATIONS, CONFIRM\_RESERVATION, CANCEL\_RESERVATION). Кроме того, администратору доступен просмотр списка всех пользователей системы с помощью процедуры GET\_ALL\_USERS. Также администратор может получать список бронирований в указанной гостинице за определенные период времени (get\_reservations\_in\_hotel).

Сценарий использования приведен на листинге В.2 ниже.

**6.1 Вывод**

Система предоставляет широкий спектр функций для различных типов пользователей. Гостям предоставляется возможность просматривать информацию и проверять доступность номеров, тогда как зарегистрированные пользователи могут бронировать номера и управлять своими бронированиями. Администраторы обладают полномочиями по управлению всей системой, включая добавление и удаление отелей и номеров, а также управление пользователями и их бронированиями. Эта структура позволяет эффективно удовлетворять потребности всех категорий пользователей.

# Заключение

Цель данной курсовой работы заключалась в создании реляционной базы данных для системы бронирования номеров в отелях, которая обеспечивает пользователя доступом к информации о доступных номерах и отелях.

База данных, будучи организованным собранием данных, играла ключевую роль в данном проекте. В процессе работы была разработана реляционная база данных, где данные представлены в виде таблиц, состоящих из строк и столбцов. В качестве системы управления базами данных была выбрана СУБД Oracle, известная своей надежностью и высокой производительностью, что позволило эффективно управлять большим объемом данных.

В процессе выполнения работы были использованы различные объекты, включая таблицы, триггеры и функции, чтобы обеспечить структурированное хранение данных и своевременный доступ к ним. В результате, цель работы была успешно достигнута, и база данных готова к использованию.

Тестирование базы данных было проведено при использовании большого объема данных, и результаты были положительными. Были реализованы процедуры для импорта и экспорта данных в формате Json, что обеспечило удобство использования и управления данными.

В целом, результаты работы говорят о том, что база данных полностью соответствует поставленным требованиям и может успешно использоваться для сервиса резервирования номеров в отелях

# Список используемых источников

1. [MyOra - Бесплатный инструмент Oracle для мониторинга баз данных и SQL (myorasql.com)](https://myorasql.com/) [Электронный ресурс] – Режим доступа: <https://myorasql.com/index.html> – Дата доступа: 27.04.2024
2. Блинова Е.А. Курс лекций по базам данных, [Электронный ресурс]. – Режим доступа: https://diskstation.belstu.by:5001/. – Дата доступа: 17.05.2024.
3. Habr «10 приёмов работы с Oracle» [Электронный ресурс] –  Режим доступа: https://habr.com/ru/companies/sberbank/articles/519358 – Дата доступа: 01.04.2024.
4. Официальная документация Oracle [Электронный ресурс] – Режим доступа: https://docs.oracle.com/en/ – Дата доступа 12.04.2024.
5. Oracle Documentation: JSON Developer's Guide [Электронный ресурс] - Режим доступа: <https://docs.oracle.com/en/database/oracle/oracle-database/19/adxjs/index.html> – Дата доступа: 09.03.2024
6. Oracle для профессионалов: Пер. с англ . - М: 000 "ИД. Вильяме" 2016 - 960 с

**Приложение А**

|  |
| --- |
| --табличные пространства  CREATE TABLESPACE TS\_COURSE  DATAFILE 'D:\KURSORACLE\TS\_COURSE.DBF'  SIZE 20M  AUTOEXTEND ON NEXT 1M  MAXSIZE 2048M;  CREATE TABLESPACE TS\_USER  DATAFILE 'D:\KURSORACLE\TS\_USER.dbf'  SIZE 100M  AUTOEXTEND ON;  CREATE TABLESPACE TS\_HOTELS  DATAFILE 'D:\KURSORACLE\TS\_HOTELS.dbf'  SIZE 100M  AUTOEXTEND ON;  CREATE TABLESPACE TS\_RESERVATION  DATAFILE 'D:\KURSORACLE\TS\_RESERVATION.dbf'  SIZE 100M  AUTOEXTEND ON;  --таблицы  CREATE TABLE CUSTOMERS (  ID\_CUSTOM NUMBER(10) PRIMARY KEY,  NAME VARCHAR2(50) NOT NULL,  SURNAME VARCHAR2(50) NOT NULL,  EMAIL VARCHAR2(50) NOT NULL UNIQUE,  PHONE\_NUM VARCHAR2(20) NOT NULL UNIQUE,  LOGIN VARCHAR2(20) NOT NULL UNIQUE  )TABLESPACE TS\_USER;  CREATE TABLE ROLE(  LOGIN NUMBER(10) PRIMARY KEY,  STATUS VARCHAR2(50) NOT NULL,  PASSWORD VARCHAR2(50) NOT NULL  )TABLESPACE TS\_USER;  CREATE TABLE PAYMENTS(  ID\_PAY NUMBER(10) PRIMARY KEY,  ID\_RESERV NUMBER(10),  SUM\_PAY NUMBER(10, 2) NOT NULL,  STATUS VARCHAR2(50),  CONSTRAINT FK\_ID\_RESERV FOREIGN KEY (ID\_RESERV) REFERENCES RESERVATIONS(ID\_RESERV)  )TABLESPACE TS\_RESERVATION;  CREATE TABLE RESERVATIONS(  ID\_RESERV NUMBER(10) PRIMARY KEY,  ID\_CUSTOM NUMBER(10),  ID\_NUM NUMBER(10),  STATUS VARCHAR2(50),  DATE\_A DATE NOT NULL,  DATE\_D DATE NOT NULL,  CONSTRAINT FK\_CUSTOM FOREIGN KEY (ID\_CUSTOM) REFERENCES CUSTOMERS(ID\_CUSTOM),  CONSTRAINT FK\_NUM FOREIGN KEY (ID\_NUM) REFERENCES ROOMS(ID\_NUM)  )TABLESPACE TS\_RESERVATION;  CREATE TABLE HOTELS(  ID\_HOT NUMBER(10) PRIMARY KEY,  NAME\_HOT VARCHAR2(50),  CITY VARCHAR2(50),  ADRESS VARCHAR2(50),  RATING VARCHAR2(50)  )TABLESPACE TS\_HOTELS;  CREATE TABLE ROOMS(  ID\_NUM NUMBER(10) PRIMARY KEY,  ID\_HOT NUMBER(10) ,  TYPE VARCHAR2(50),  PRICE NUMBER(10,2),  AVAILABILITY VARCHAR2(50),  NUM\_OF\_ROOM NUMBER(10),  CONSTRAINT FK\_ID\_HOT FOREIGN KEY (ID\_HOT) REFERENCES HOTELS(ID\_HOT)  )TABLESPACE TS\_HOTELS; |

Листинг А.1 – Создание таблиц

|  |
| --- |
| --заполнение HOTELS  CREATE OR REPLACE FUNCTION populate\_hotels\_table  RETURN BOOLEAN  AS  TYPE t\_name\_arr IS TABLE OF VARCHAR2(50);  TYPE t\_city\_arr IS TABLE OF VARCHAR2(50);  TYPE t\_address\_arr IS TABLE OF VARCHAR2(50);  TYPE t\_rating\_arr IS TABLE OF VARCHAR2(50);  names t\_name\_arr := t\_name\_arr('Курорт Закатный Пляж', 'Лесной Замок', 'Королевский Дворец', 'Курорт Золотые Пески', 'Тропический Рай Инн',  'Шале Снежные Пики', 'Отель Городские Огни', 'Речная Усадьба', 'Пустынный Оазис', 'Отель Вид на Гавань');  cities t\_city\_arr := t\_city\_arr('Гонолулу', 'Денвер', 'Лондон', 'Дубай', 'Бали', 'Уистлер', 'Токио', 'Париж', 'Сидней');  addresses t\_address\_arr := t\_address\_arr('ул. Вайкики, 123', 'ул. Сосновая, 456', 'ул. Букингемская, 789', 'ул. Джумейра Бич, 123', 'ул. Пляж Кута, 456',  'ул. Горный Городок, 789', 'ул. Шиндзюку, 123', 'ул. Набережная Сены, 456', 'ул. Аравийская Пустыня, 789',  'ул. Круговая Причал, 123');  ratings t\_rating\_arr := t\_rating\_arr('4 звезды', '3 звезды', '5 звезд');  name\_idx NUMBER;  city\_idx NUMBER;  address\_idx NUMBER;  rating\_idx NUMBER;  i PLS\_INTEGER := 5;  BEGIN  WHILE i <= 100005 LOOP  name\_idx := TRUNC(DBMS\_RANDOM.VALUE(1, names.COUNT + 1));  city\_idx := TRUNC(DBMS\_RANDOM.VALUE(1, cities.COUNT + 1));  address\_idx := TRUNC(DBMS\_RANDOM.VALUE(1, addresses.COUNT + 1));  rating\_idx := TRUNC(DBMS\_RANDOM.VALUE(1, ratings.COUNT + 1));    INSERT INTO HOTELS (ID\_HOT, NAME\_HOT, CITY, ADRESS, RATING)  VALUES (i, names(name\_idx), cities(city\_idx), addresses(address\_idx), ratings(rating\_idx));  i:= i+1;  END LOOP;  RETURN TRUE;  EXCEPTION  WHEN OTHERS THEN  RETURN FALSE;  END;  --ЗАПУСК ФУНКЦИИ  BEGIN  IF populate\_hotels\_table() THEN  DBMS\_OUTPUT.PUT\_LINE('Таблица HOTELS успешно заполнена.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Ошибка заполнения таблицы HOTELS.');  END IF;  END;  --заполнение ROOMS  CREATE OR REPLACE FUNCTION populate\_rooms\_table  RETURN BOOLEAN IS  BEGIN  FOR i IN 1..100000 LOOP  INSERT INTO ROOMS (ID\_NUM, ID\_HOT, TYPE, PRICE, AVAILABILITY, NUM\_OF\_ROOM)  VALUES (i,  ROUND(DBMS\_RANDOM.VALUE(1, 100000)),  CASE ROUND(DBMS\_RANDOM.VALUE(1, 6))  WHEN 1 THEN 'Стандартный'  WHEN 2 THEN 'Люкс'  WHEN 3 THEN 'Двухэтажный люкс'  WHEN 4 THEN 'Двухместный'  WHEN 5 THEN 'Трёхместный'  WHEN 6 THEN 'Президентский люкс'  END,  ROUND(DBMS\_RANDOM.VALUE(100, 1000)),  CASE ROUND(DBMS\_RANDOM.VALUE(0, 1))  WHEN 0 THEN 'Доступен'  WHEN 1 THEN 'Не доступен'  END,  ROUND(DBMS\_RANDOM.VALUE(1, 500)));  END LOOP;    RETURN TRUE;  EXCEPTION  WHEN OTHERS THEN  RETURN FALSE;  END;  --запуск функции  BEGIN  IF populate\_rooms\_table() THEN  DBMS\_OUTPUT.PUT\_LINE('Таблица ROOMS успешно заполнена.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Ошибка заполнения таблицы ROOMS.');  END IF;  END;  --заполнение CUSTOMERS  CREATE OR REPLACE FUNCTION populate\_customers\_table  RETURN BOOLEAN IS  BEGIN  FOR i IN 1..100000 LOOP  INSERT INTO CUSTOMERS (ID\_CUSTOM, NAME, SURNAME, EMAIL, PHONE\_NUM, LOGIN)  VALUES (i,  CASE ROUND(DBMS\_RANDOM.VALUE(1, 15))  WHEN 1 THEN 'Иван'  WHEN 2 THEN 'Мария'  WHEN 3 THEN 'Александр'  WHEN 4 THEN 'Елена'  WHEN 5 THEN 'Дмитрий'  WHEN 6 THEN 'Ольга'  WHEN 7 THEN 'Стефания'  WHEN 8 THEN 'Виктория'  WHEN 9 THEN 'Яна'  WHEN 10 THEN 'Алеся'  WHEN 11 THEN 'Полина'  WHEN 12 THEN 'Андрей'  WHEN 13 THEN 'Николай'  WHEN 14 THEN 'Артур'  WHEN 15 THEN 'Ксения'  END,  CASE ROUND(DBMS\_RANDOM.VALUE(1, 12))  WHEN 1 THEN 'Гришко'  WHEN 2 THEN 'Савко'  WHEN 3 THEN 'Грицкевич'  WHEN 4 THEN 'Дашкевич'  WHEN 5 THEN 'Микайлян'  WHEN 6 THEN 'Мущук'  WHEN 7 THEN 'Панченко'  WHEN 8 THEN 'Гончар'  WHEN 9 THEN 'Осоко'  WHEN 10 THEN 'Якубенко'  WHEN 11 THEN 'Юхимук'  WHEN 12 THEN 'Дятко'  END,  DBMS\_RANDOM.STRING('a', 10) || i || '@gmail.com',  '+375' || i,  'login\_' || i );  END LOOP;    RETURN TRUE;  EXCEPTION  WHEN OTHERS THEN  RETURN FALSE;  END;  --запуск функции  BEGIN  IF populate\_customers\_table() THEN  DBMS\_OUTPUT.PUT\_LINE('Таблица CUSTOMERS успешно заполнена.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Ошибка заполнения таблицы CUSTOMERS.');  END IF;  END;  --заполнение RESERVATIONS  CREATE OR REPLACE FUNCTION populate\_reservations\_table RETURN BOOLEAN IS  BEGIN  FOR i IN 1..100000 LOOP  INSERT INTO RESERVATIONS (ID\_RESERV, ID\_CUSTOM, ID\_NUM, STATUS, DATE\_A, DATE\_D)  VALUES (  i,  ROUND(DBMS\_RANDOM.VALUE(1, 100000)),  ROUND(DBMS\_RANDOM.VALUE(1, 100000)),  CASE ROUND(DBMS\_RANDOM.VALUE(1, 2))  WHEN 1 THEN 'Подтверждено'  WHEN 2 THEN 'Ожидает подтверждения'  END,  TO\_DATE('2024-05-01', 'YYYY-MM-DD') + TRUNC(DBMS\_RANDOM.VALUE(0, 365)), -- Случайная дата в пределах года  TO\_DATE('2024-12-31', 'YYYY-MM-DD') - TRUNC(DBMS\_RANDOM.VALUE(0, 365))  );  END LOOP;    RETURN TRUE;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' || SQLERRM);  RETURN FALSE;  END;  --запуск функции  BEGIN  IF populate\_reservations\_table() THEN  DBMS\_OUTPUT.PUT\_LINE('Таблица RESERVATIONS успешно заполнена.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Ошибка заполнения таблицы RESERVATIONS.');  END IF;  END;  --заполнение payments  CREATE SEQUENCE payment\_seq START WITH 1 INCREMENT BY 1;  CREATE OR REPLACE FUNCTION populate\_payments\_table RETURN BOOLEAN IS  BEGIN  FOR i IN 1..100000 LOOP  DECLARE  v\_price NUMBER(10, 2);  v\_pay\_id NUMBER(10);  BEGIN  SELECT r.ID\_NUM INTO v\_price  FROM RESERVATIONS r  WHERE r.ID\_RESERV = i;  SELECT payment\_seq.NEXTVAL INTO v\_pay\_id FROM dual;  INSERT INTO PAYMENTS (ID\_PAY, ID\_RESERV, SUM\_PAY, STATUS)  VALUES (  v\_pay\_id,  i,  (SELECT PRICE FROM ROOMS WHERE ID\_NUM = v\_price),  CASE ROUND(DBMS\_RANDOM.VALUE(1, 2))  WHEN 1 THEN 'Оплачено'  WHEN 2 THEN 'Ожидает оплаты'  END  );  END;  END LOOP;    RETURN TRUE;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' || SQLERRM);  RETURN FALSE;  END;  --запуск функции  BEGIN  IF populate\_payments\_table() THEN  DBMS\_OUTPUT.PUT\_LINE('Таблица payments успешно заполнена.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Ошибка заполнения таблицы payments.');  END IF;  END;  --заполнение roles  CREATE OR REPLACE FUNCTION populate\_role\_table RETURN BOOLEAN IS  BEGIN  INSERT INTO ROLE (LOGIN, STATUS, PASSWORD)  VALUES ('admin','Admin','Pa$$w0rd');  FOR i IN 2..100000 LOOP  INSERT INTO ROLE (LOGIN, STATUS, PASSWORD)  VALUES (  'login\_' || i,  'User',  'pass' || i  );  END LOOP;    RETURN TRUE;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' || SQLERRM);  RETURN FALSE;  END;  ----запуск функции  BEGIN  IF populate\_role\_table() THEN  DBMS\_OUTPUT.PUT\_LINE('Таблица role успешно заполнена.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Ошибка заполнения таблицы role.');  END IF;  END; |

Листинг А.2 – Заполнение таблиц на 100000 строк

|  |
| --- |
| alter session set "\_oracle\_script"=true;  CREATE ROLE rl\_user;  CREATE ROLE rl\_admin;  CREATE ROLE rl\_guest;  -------admin-------  GRANT CREATE SESSION,  CREATE TABLE,  CREATE VIEW,  CREATE PROCEDURE,  CREATE TRIGGER TO rl\_admin;  CREATE USER ADMIN\_DB  IDENTIFIED BY admindb  DEFAULT TABLESPACE TS\_COURSE  QUOTA UNLIMITED ON TS\_COURSE  ACCOUNT UNLOCK;  GRANT rl\_admin TO ADMIN\_DB;  CREATE PROFILE PADMIN LIMIT  FAILED\_LOGIN\_ATTEMPTS 5  PASSWORD\_LIFE\_TIME UNLIMITED  PASSWORD\_GRACE\_TIME UNLIMITED  PASSWORD\_LOCK\_TIME UNLIMITED;  ALTER USER ADMIN\_DB PROFILE PADMIN;  GRANT ALL PRIVILEGES ON TO rl\_admin;  GRANT CONNECT TO rl\_admin;  GRANT CREATE SESSION TO rl\_admin;  GRANT CREATE ANY TABLE TO rl\_admin;  GRANT CREATE VIEW TO rl\_admin;  GRANT CREATE TRIGGER TO rl\_admin;  GRANT SELECT ANY TABLE TO rl\_admin;  GRANT CREATE PROCEDURE TO rl\_admin;  GRANT CREATE TABLESPACE TO rl\_admin;  GRANT DROP ANY TABLE TO rl\_admin;  GRANT DROP ANY VIEW TO rl\_admin;  GRANT EXECUTE ON ADD\_HOTEL rl\_admin;  GRANT EXECUTE ON DELETE\_HOTEL rl\_admin;  GRANT EXECUTE ON ADD\_ROOM rl\_admin;  GRANT EXECUTE ON DELETE\_ROOM rl\_admin;  GRANT EXECUTE ON UPDATE\_HOTEL\_RATING rl\_admin;  GRANT EXECUTE ON GET\_RESERVATION\_STATISTICS rl\_admin;  GRANT EXECUTE ON VIEW\_PAYMENTS rl\_admin;  GRANT EXECUTE ON CONFIRM\_PAYMENT rl\_admin;  GRANT EXECUTE ON VIEW\_RESERVATIONS rl\_admin;  GRANT EXECUTE ON CONFIRM\_RESERVATION rl\_admin;  GRANT EXECUTE ON CANCEL\_RESERVATION rl\_admin;  GRANT EXECUTE ON GET\_ALL\_USERS rl\_admin;  --------user-------------  GRANT SELECT ON HOTELS TO rl\_user;  GRANT EXECUTE ON BOOK\_ROOM TO rl\_user;  GRANT EXECUTE ON CANCEL\_RESERVATION TO rl\_user;  GRANT EXECUTE ON GET\_MY\_RESERVATIONS TO rl\_user;  GRANT EXECUTE ON GET\_AVAILABLE\_ROOMS TO rl\_user;  GRANT EXECUTE ON GET\_RESERVATION\_BILL TO rl\_user;  GRANT EXECUTE ON GET\_ROOMS\_BY\_HOTEL TO rl\_user;  GRANT EXECUTE ON GET\_AVAILABLE\_ROOMS\_IN\_HOTEL TO rl\_user;  GRANT CREATE SESSION  TO rl\_admin;  CREATE USER USER\_DB  IDENTIFIED BY userdb  DEFAULT TABLESPACE TS\_COURSE  QUOTA UNLIMITED ON TS\_COURSE  ACCOUNT UNLOCK;  GRANT rl\_user TO USER\_DB;  CREATE PROFILE PUSER LIMIT  FAILED\_LOGIN\_ATTEMPTS 5  PASSWORD\_LIFE\_TIME UNLIMITED  PASSWORD\_GRACE\_TIME UNLIMITED  PASSWORD\_LOCK\_TIME UNLIMITED;  ALTER USER USER\_DB PROFILE PUSER;  ---------guest------  CREATE USER GUEST\_DB  IDENTIFIED BY guestdb  DEFAULT TABLESPACE TS\_COURSE  QUOTA UNLIMITED ON TS\_COURSE  ACCOUNT UNLOCK;  GRANT rl\_guest TO GUEST\_DB;  GRANT EXECUTE ON GET\_ALL\_USERS TO rl\_guest;  GRANT EXECUTE ON GET\_ALL\_HOTELS TO rl\_guest;  GRANT EXECUTE ON GET\_ROOMS\_BY\_HOTEL TO rl\_guest;  GRANT EXECUTE ON GET\_AVAILABLE\_ROOMS\_IN\_HOTEL TO rl\_guest;  GRANT EXECUTE ON get\_hotel\_rating\_in\_city TO rl\_guest;  GRANT EXECUTE ON get\_room\_type TO rl\_guest;  GRANT EXECUTE ON get\_room\_price TO rl\_guest;  GRANT EXECUTE ON get\_room\_availability TO rl\_guest; |

Листинг А.3 – Создание ролей и пользователей

**Приложение Б**

|  |
| --- |
| ------procedures----  -----admin--------------------------------  --процедуры для просмотра платежей и их подтверждения  CREATE OR REPLACE PROCEDURE VIEW\_PAYMENTS  AS  BEGIN  FOR p IN (SELECT ID\_PAY, ID\_RESERV, SUM\_PAY, STATUS FROM PAYMENTS)  LOOP  DBMS\_OUTPUT.PUT\_LINE('ID Платежа: ' || p.ID\_PAY);  DBMS\_OUTPUT.PUT\_LINE('ID Бронирования: ' || p.ID\_RESERV);  DBMS\_OUTPUT.PUT\_LINE('Сумма Платежа: ' || p.SUM\_PAY);  DBMS\_OUTPUT.PUT\_LINE('Статус: ' || p.STATUS);  DBMS\_OUTPUT.PUT\_LINE('-----------------------------');  END LOOP;  END;  --Процедура для изменения информации о номере  CREATE OR REPLACE PROCEDURE update\_room\_info (  p\_id\_num IN NUMBER,  p\_id\_hot IN NUMBER,  p\_type IN VARCHAR2,  p\_price IN NUMBER,  p\_availability IN VARCHAR2,  p\_num\_of\_room IN NUMBER  ) IS  BEGIN  UPDATE rooms  SET id\_hot = p\_id\_hot,  type = p\_type,  price = p\_price,  availability = p\_availability,  num\_of\_room = p\_num\_of\_room  WHERE id\_num = p\_id\_num;  IF SQL%ROWCOUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('Room with ID ' || p\_id\_num || ' not found.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Room with ID ' || p\_id\_num || ' successfully updated.');  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END;  --Получение списка всех пользователей  CREATE OR REPLACE PROCEDURE GET\_ALL\_CUSTOMERS\_INFO  AS  BEGIN  FOR r IN (SELECT ID\_CUSTOM, NAME, SURNAME, EMAIL, PHONE\_NUM, LOGIN  FROM CUSTOMERS)  LOOP  DBMS\_OUTPUT.PUT\_LINE('Customer ID: ' || r.ID\_CUSTOM);  DBMS\_OUTPUT.PUT\_LINE('Name: ' || r.NAME);  DBMS\_OUTPUT.PUT\_LINE('Surname: ' || r.SURNAME);  DBMS\_OUTPUT.PUT\_LINE('Email: ' || r.EMAIL);  DBMS\_OUTPUT.PUT\_LINE('Phone Number: ' || r.PHONE\_NUM);  DBMS\_OUTPUT.PUT\_LINE('Login: ' || r.LOGIN);  DBMS\_OUTPUT.PUT\_LINE('-----------------------------');  END LOOP;  END;  --удаление комнаты  CREATE OR REPLACE PROCEDURE delete\_room (  p\_id\_num IN NUMBER  ) IS  BEGIN  -- Удаляем номер из таблицы ROOMS  DELETE FROM rooms  WHERE id\_num = p\_id\_num;  -- Проверяем, была ли удалена какая-либо строка  IF SQL%ROWCOUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('Room with ID ' || p\_id\_num || ' not found.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('Room with ID ' || p\_id\_num || ' successfully deleted.');  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END;  --Процедура для удаления пользователя по логину  CREATE OR REPLACE PROCEDURE delete\_user\_by\_login (  p\_login IN VARCHAR2  ) IS  BEGIN  -- Удаление пользователя из таблицы CUSTOMERS  DELETE FROM customers  WHERE login = p\_login;  -- Удаление пользователя из таблицы ROLE  DELETE FROM role  WHERE login = p\_login;  -- Проверка, была ли удалена хотя бы одна строка из таблицы CUSTOMERS  IF SQL%ROWCOUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('User with login ' || p\_login || ' not found.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('User with login ' || p\_login || ' successfully deleted.');  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END;  --Процедура для изменения данных пользователя  CREATE OR REPLACE PROCEDURE update\_user\_by\_login (  p\_login IN VARCHAR2,  p\_name IN VARCHAR2,  p\_surname IN VARCHAR2,  p\_email IN VARCHAR2,  p\_phone\_num IN VARCHAR2  ) IS  BEGIN  -- Проверяем валидность email  is\_valid := is\_valid\_email(p\_email);  IF NOT is\_valid THEN  DBMS\_OUTPUT.PUT\_LINE('Invalid email format: ' || p\_email);  RETURN;  END IF;  -- Обновление данных пользователя в таблице CUSTOMERS  UPDATE customers  SET  name = p\_name,  surname = p\_surname,  email = p\_email,  phone\_num = p\_phone\_num  WHERE login = p\_login;  -- Проверка, была ли обновлена хотя бы одна строка  IF SQL%ROWCOUNT = 0 THEN  DBMS\_OUTPUT.PUT\_LINE('User with login ' || p\_login || ' not found.');  ELSE  DBMS\_OUTPUT.PUT\_LINE('User with login ' || p\_login || ' successfully updated.');  END IF;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END;  --Удаление отеля из системы  CREATE OR REPLACE PROCEDURE DELETE\_HOTEL(  p\_id\_hot IN NUMBER  )  AS  BEGIN  DELETE FROM HOTELS WHERE ID\_HOT = p\_id\_hot;  COMMIT;  END;  --Добавление нового номера в отель  CREATE OR REPLACE PROCEDURE ADD\_ROOM(  p\_id\_num IN NUMBER,  p\_id\_hot IN NUMBER,  p\_type IN VARCHAR2,  p\_price IN NUMBER,  p\_availability IN VARCHAR2,  p\_num\_of\_room IN NUMBER  )  AS  BEGIN  INSERT INTO ROOMS (ID\_NUM, ID\_HOT, TYPE, PRICE, AVAILABILITY, NUM\_OF\_ROOM)  VALUES (p\_id\_num, p\_id\_hot, p\_type, p\_price, p\_availability, p\_num\_of\_room);  COMMIT;  END;  --Удаление номера из отеля  CREATE OR REPLACE PROCEDURE DELETE\_ROOM(  p\_id\_num IN NUMBER  )  AS  BEGIN  DELETE FROM ROOMS WHERE ID\_NUM = p\_id\_num;  COMMIT;  END;  --процедура для обновления рейтинга отеля  CREATE OR REPLACE PROCEDURE UPDATE\_HOTEL\_RATING(  p\_id\_hot IN NUMBER,  p\_new\_rating IN VARCHAR2  )  AS  BEGIN  UPDATE HOTELS SET RATING = p\_new\_rating WHERE ID\_HOT = p\_id\_hot;  COMMIT;  END;  --Получение статистики по бронированиям в системе  CREATE OR REPLACE PROCEDURE GET\_RESERVATION\_STATISTICS  AS  total\_reservations NUMBER;  confirmed\_reservations NUMBER;  pending\_reservations NUMBER;  cancelled\_reservations NUMBER;  BEGIN  SELECT COUNT(\*),  SUM(CASE WHEN STATUS = 'Подтверждено' THEN 1 ELSE 0 END),  SUM(CASE WHEN STATUS = 'Ожидает подтверждения' THEN 1 ELSE 0 END),  SUM(CASE WHEN STATUS = 'Отменено' THEN 1 ELSE 0 END)  INTO total\_reservations, confirmed\_reservations, pending\_reservations, cancelled\_reservations  FROM RESERVATIONS;  DBMS\_OUTPUT.PUT\_LINE('Статистика бронирований:');  DBMS\_OUTPUT.PUT\_LINE('Всего бронирований: ' || total\_reservations);  DBMS\_OUTPUT.PUT\_LINE('Подтвержденных бронирований: ' || confirmed\_reservations);  DBMS\_OUTPUT.PUT\_LINE('Бронирований, ожидающих подтверждения: ' || pending\_reservations);  DBMS\_OUTPUT.PUT\_LINE('Отмененных бронирований: ' || cancelled\_reservations);  END;  --для пользователя---------------------------------------------------------  -- Процедура для бронирования номера в отеле  CREATE SEQUENCE SEQ\_RESERVATION  START WITH 1  INCREMENT BY 1  NOCACHE  NOCYCLE;    CREATE OR REPLACE PROCEDURE BOOK\_ROOM(  CUSTOMER\_ID IN NUMBER,  HOTEL\_ID IN NUMBER,  ROOM\_ID IN NUMBER,  START\_DATE IN DATE,  END\_DATE IN DATE)  AS  BEGIN  INSERT INTO RESERVATIONS (ID\_RESERV, ID\_CUSTOM, ID\_NUM, STATUS, DATE\_A, DATE\_D)  VALUES (SEQ\_RESERVATION.NEXTVAL, CUSTOMER\_ID, ROOM\_ID, 'Pending', START\_DATE, END\_DATE);  COMMIT;  END;  -- Процедура для отмены собственного бронирования  CREATE OR REPLACE PROCEDURE CANCEL\_RESERVATION(  RESERVATION\_ID IN NUMBER)  AS  BEGIN  DELETE FROM RESERVATIONS WHERE ID\_RESERV = RESERVATION\_ID ;  COMMIT;  END;  -- Процедура для получения счета за собственное бронирование  CREATE OR REPLACE PROCEDURE GET\_SUM\_PAY\_BY\_RESERVATION(  p\_id\_reserv IN NUMBER  )  AS  v\_sum\_pay NUMBER(10, 2);  BEGIN  -- Получение суммы платежей для данного ID\_RESERV  SELECT SUM(SUM\_PAY)  INTO v\_sum\_pay  FROM PAYMENTS  WHERE ID\_RESERV = p\_id\_reserv;  -- Проверка, чтобы сумма не была NULL  IF v\_sum\_pay IS NULL THEN  v\_sum\_pay := 0;  END IF;  -- Вывод суммы  DBMS\_OUTPUT.PUT\_LINE('Сумма платежей для бронирования: ' || v\_sum\_pay);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Бронирование с ID ' || p\_id\_reserv || ' не найдено.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Ошибка: ' || SQLERRM);  END;  --добавление нового отеля в систему  CREATE OR REPLACE PROCEDURE ADD\_HOTEL(  p\_id\_hot IN NUMBER,  p\_name\_hot IN VARCHAR2,  p\_city IN VARCHAR2,  p\_address IN VARCHAR2,  p\_rating IN VARCHAR2  )  AS  BEGIN  INSERT INTO HOTELS (ID\_HOT, NAME\_HOT, CITY, ADRESS, RATING)  VALUES (p\_id\_hot, p\_name\_hot, p\_city, p\_address, p\_rating);  COMMIT;  END;  -- Процедура для получения информации о собственных бронированиях  CREATE OR REPLACE PROCEDURE GET\_MY\_RESERVATIONS(  p\_id\_custom IN NUMBER  )  AS  BEGIN  FOR r IN (SELECT ID\_RESERV, ID\_CUSTOM, ID\_NUM, STATUS, DATE\_A, DATE\_D  FROM RESERVATIONS  WHERE ID\_CUSTOM = p\_id\_custom)  LOOP  DBMS\_OUTPUT.PUT\_LINE('Комната: ' || r.ID\_NUM);  DBMS\_OUTPUT.PUT\_LINE('Статус: ' || r.STATUS);  DBMS\_OUTPUT.PUT\_LINE('Дата заезда : ' || TO\_CHAR(r.DATE\_A, 'DD-MM-YYYY'));  DBMS\_OUTPUT.PUT\_LINE('Дата отъезда: ' || TO\_CHAR(r.DATE\_D, 'DD-MM-YYYY'));  DBMS\_OUTPUT.PUT\_LINE('-----------------------------');  END LOOP;  END;  CREATE OR REPLACE PROCEDURE CONFIRM\_PAYMENT(  p\_id\_pay IN NUMBER  )  AS  BEGIN  UPDATE PAYMENTS SET STATUS = 'Оплачено' WHERE ID\_PAY = p\_id\_pay;  COMMIT;  END;  --Управление бронированиями  --просмотр  CREATE OR REPLACE PROCEDURE VIEW\_RESERVATIONS(  p\_out\_cursor OUT SYS\_REFCURSOR  )  AS  BEGIN  OPEN p\_out\_cursor FOR  SELECT \* FROM RESERVATIONS;  END;  --подтверждение  CREATE OR REPLACE PROCEDURE CONFIRM\_RESERVATION(  p\_id\_reserv IN NUMBER  )  AS  BEGIN  UPDATE RESERVATIONS SET STATUS = 'Подтверждено' WHERE ID\_RESERV = p\_id\_reserv;  COMMIT;  END;  --отмена  CREATE OR REPLACE PROCEDURE CANCEL\_RESERVATION(  p\_id\_reserv IN NUMBER  )  AS  BEGIN  UPDATE RESERVATIONS SET STATUS = 'Отменено' WHERE ID\_RESERV = p\_id\_reserv;  COMMIT;  END;  --для гостя  --Получение списка всех отелей  CREATE OR REPLACE PROCEDURE GET\_ALL\_HOTELS  AS  BEGIN  FOR r IN (SELECT ID\_HOT, NAME\_HOT, CITY, ADRESS, RATING  FROM HOTELS)  LOOP  DBMS\_OUTPUT.PUT\_LINE('Hotel ID: ' || r.ID\_HOT);  DBMS\_OUTPUT.PUT\_LINE('Hotel Name: ' || r.NAME\_HOT);  DBMS\_OUTPUT.PUT\_LINE('City: ' || r.CITY);  DBMS\_OUTPUT.PUT\_LINE('Address: ' || r.ADRESS);  DBMS\_OUTPUT.PUT\_LINE('Rating: ' || r.RATING);  DBMS\_OUTPUT.PUT\_LINE('-----------------------------');  END LOOP;  END;  --Получение списка всех номеров в конкретном отеле  CREATE OR REPLACE PROCEDURE GET\_ROOMS\_IN\_HOTEL(  p\_id\_hot IN NUMBER  ) IS  CURSOR c\_available\_rooms IS  SELECT num\_of\_room, price  FROM rooms  WHERE id\_hot = p\_id\_hot;  v\_num\_of\_room rooms.num\_of\_room%TYPE;  v\_price rooms.price%TYPE;  BEGIN  OPEN c\_available\_rooms;  LOOP  FETCH c\_available\_rooms INTO v\_num\_of\_room, v\_price;  EXIT WHEN c\_available\_rooms%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE('Комната: ' || v\_num\_of\_room || ' свободна ' );  END LOOP;  CLOSE c\_available\_rooms;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END;  --вывод информации о свободных номерах в конкретном отеле  CREATE OR REPLACE PROCEDURE GET\_AVAILABLE\_ROOMS\_IN\_HOTEL(  p\_id\_hot IN NUMBER  ) IS  CURSOR c\_available\_rooms IS  SELECT num\_of\_room, price  FROM rooms  WHERE availability = 'Свободен'  AND id\_hot = p\_id\_hot;  v\_num\_of\_room rooms.num\_of\_room%TYPE;  v\_price rooms.price%TYPE;  BEGIN  OPEN c\_available\_rooms;  LOOP  FETCH c\_available\_rooms INTO v\_num\_of\_room, v\_price;  EXIT WHEN c\_available\_rooms%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE('Комната: ' || v\_num\_of\_room || ' свободна, Цена: ' || v\_price ||' byn');  END LOOP;  CLOSE c\_available\_rooms;  EXCEPTION  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END;  --Процедура для вывода рейтинга определенного отеля  CREATE OR REPLACE PROCEDURE get\_hotel\_rating\_in\_city(  p\_name\_hot IN VARCHAR2,  p\_city IN VARCHAR2  ) AS  CURSOR c\_hotel\_ratings IS  SELECT rating  FROM hotels  WHERE name\_hot = p\_name\_hot  AND city = p\_city;  v\_rating hotels.rating%TYPE;  BEGIN  OPEN c\_hotel\_ratings;  LOOP  FETCH c\_hotel\_ratings INTO v\_rating;  EXIT WHEN c\_hotel\_ratings%NOTFOUND;  DBMS\_OUTPUT.PUT\_LINE('Рейтинг отеля: ' || p\_name\_hot || ' in ' || p\_city || ': ' || v\_rating);  END LOOP;  CLOSE c\_hotel\_ratings;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Hotel ' || p\_name\_hot || ' in ' || p\_city || ' not found.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END;  --Процедура для вывода типа комнаты  CREATE OR REPLACE PROCEDURE get\_room\_type(  p\_num\_of\_room IN NUMBER,  p\_id\_hot IN NUMBER  ) AS  v\_type rooms.type%TYPE;  BEGIN  SELECT type  INTO v\_type  FROM rooms  WHERE num\_of\_room = p\_num\_of\_room  AND id\_hot = p\_id\_hot;    DBMS\_OUTPUT.PUT\_LINE('Тип комнаты ' || p\_num\_of\_room || ' в отеле ' || p\_id\_hot || ': ' || v\_type);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Комната ' || p\_num\_of\_room || ' в отеле ' || p\_id\_hot || 'не найдена.');  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('Комнаты ' || p\_num\_of\_room || ' в отеле ' || p\_id\_hot || '.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END;  --Процедура для вывода цены комнаты  CREATE OR REPLACE PROCEDURE get\_room\_price(  p\_num\_of\_room IN NUMBER,  p\_id\_hot IN NUMBER  ) AS  v\_price rooms.price%TYPE;  BEGIN  SELECT price  INTO v\_price  FROM rooms  WHERE num\_of\_room = p\_num\_of\_room  AND id\_hot = p\_id\_hot;    DBMS\_OUTPUT.PUT\_LINE('Стоимость комнаты ' || p\_num\_of\_room || ' в отеле ' || p\_id\_hot || ': ' || v\_price || ' byn.');  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Комната' || p\_num\_of\_room || ' в отеле ' || p\_id\_hot || ' не найдена.');  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('Комнаты ' || p\_num\_of\_room || 'в отеле ' || p\_id\_hot || '.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END;  --Процедура для вывода доступности комнаты  CREATE OR REPLACE PROCEDURE get\_room\_availability(  p\_num\_of\_room IN NUMBER,  p\_id\_hot IN NUMBER  ) AS  v\_availability rooms.availability%TYPE;  BEGIN  SELECT availability  INTO v\_availability  FROM rooms  WHERE num\_of\_room = p\_num\_of\_room  AND id\_hot = p\_id\_hot;    DBMS\_OUTPUT.PUT\_LINE('Доступность комнаты ' || p\_num\_of\_room || ' в отеле ' || p\_id\_hot || ': ' || v\_availability);  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  DBMS\_OUTPUT.PUT\_LINE('Команата ' || p\_num\_of\_room || ' в отеле ' || p\_id\_hot || ' не найдена.');  WHEN TOO\_MANY\_ROWS THEN  DBMS\_OUTPUT.PUT\_LINE('Комнаты ' || p\_num\_of\_room || 'в отеле ' || p\_id\_hot || '.');  WHEN OTHERS THEN  DBMS\_OUTPUT.PUT\_LINE('Error: ' || SQLERRM);  END; |

Листинг Б.1 – Создание процедур

|  |
| --- |
| --Триггер для автоматического изменения  --статуса брони номера при оплате  CREATE OR REPLACE TRIGGER payment\_trigger  AFTER UPDATE OF status ON payments  FOR EACH ROW  BEGIN  IF :OLD.status <> :NEW.status THEN  IF :NEW.status = 'Оплачено' THEN  UPDATE reservations  SET status = 'Подтверждено'  WHERE id\_reserv = :NEW.id\_reserv;  ELSIF :NEW.status = 'Не оплачено' THEN  UPDATE reservations  SET status = 'Ожидает подтверждения'  WHERE id\_reserv = :NEW.id\_reserv;  END IF;  END IF;  END;  UPDATE payments  SET status = 'Не оплачено'  WHERE id\_reserv = 1;  --Триггер для автоматического изменения  --доступности номера при создании брони  CREATE OR REPLACE TRIGGER reservation\_trigger  AFTER INSERT ON reservations  FOR EACH ROW  BEGIN  UPDATE rooms  SET availability = 'Не доступен'  WHERE id\_num = :new.id\_num;  END;  INSERT INTO reservations (ID\_RESERV, ID\_CUSTOM, ID\_NUM, STATUS, DATE\_A, DATE\_D)  VALUES (5, 1, 3, 'Ожидает подтверждения', TO\_DATE('2024-08-05', 'YYYY-MM-DD'),  TO\_DATE('2024-08-10', 'YYYY-MM-DD'))  --Триггер для проверки доступности  --номера при создании новой брони  CREATE OR REPLACE TRIGGER availability\_trigger  BEFORE INSERT ON reservations  FOR EACH ROW  DECLARE  room\_count NUMBER;  BEGIN  SELECT COUNT(\*)  INTO room\_count  FROM rooms  WHERE id\_num = :NEW.id\_num  AND availability = 'Доступен';    IF room\_count = 0 THEN  RAISE\_APPLICATION\_ERROR(-20001, 'Номер занят. Пожалуйста, выберите другой');  END IF;  END;  INSERT INTO reservations (ID\_RESERV, ID\_CUSTOM, ID\_NUM, STATUS, DATE\_A, DATE\_D)  VALUES (8, 1, 1, 'Ожидает подтверждения', TO\_DATE('2024-08-05', 'YYYY-MM-DD'),  TO\_DATE('2024-08-10', 'YYYY-MM-DD')) |

Листинг Б.2 – Создание триггеров

|  |
| --- |
| Таблица PAYMENTS  CREATE INDEX idx\_payments\_id\_reserv ON PAYMENTS(ID\_RESERV);  -- Таблица RESERVATIONS  CREATE INDEX idx\_reservations\_id\_custom ON RESERVATIONS(ID\_CUSTOM);  CREATE INDEX idx\_reservations\_id\_num ON RESERVATIONS(ID\_NUM);  -- Таблица HOTELS  CREATE INDEX idx\_hotels\_city ON HOTELS(CITY);  CREATE INDEX idx\_hotels\_rating ON HOTELS(RATING);  -- Таблица ROOMS  CREATE INDEX idx\_rooms\_id\_hot ON ROOMS(ID\_HOT);  CREATE INDEX idx\_rooms\_availability ON ROOMS(AVAILABILITY); |

Листинг Б.3 – Создание индексов

|  |
| --- |
| --функция, которая проверяет  --на валидносьт email  CREATE OR REPLACE FUNCTION is\_valid\_email(email IN VARCHAR2) RETURN BOOLEAN IS  BEGIN  -- Проверяем, соответствует ли email заданному формату  IF REGEXP\_LIKE(email, '^[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$') THEN  RETURN TRUE;  ELSE  RETURN FALSE;  END IF;  END;  --Получение списка всех гостиниц в указанном городе  CREATE OR REPLACE FUNCTION get\_hotels\_in\_city(city\_name IN VARCHAR2)  RETURN SYS\_REFCURSOR  AS  hotel\_cursor SYS\_REFCURSOR;  BEGIN  OPEN hotel\_cursor FOR  SELECT \*  FROM hotels  WHERE city = city\_name;    RETURN hotel\_cursor;  END;  --Получение списка бронирований в указанной гостинице  --за определенный период времени  CREATE OR REPLACE FUNCTION get\_reservations\_in\_hotel(hotel\_id IN NUMBER, start\_date IN DATE, end\_date IN DATE)  RETURN SYS\_REFCURSOR  AS  reservation\_cursor SYS\_REFCURSOR;  BEGIN  OPEN reservation\_cursor FOR  SELECT \*  FROM reservations  WHERE id\_num IN (SELECT id\_num FROM rooms WHERE id\_hot = hotel\_id)  AND DATE\_A BETWEEN start\_date AND end\_date;    RETURN reservation\_cursor;  END; |

Листинг Б.4 – Создание функций

**Приложение В**

|  |
| --- |
| CREATE OR REPLACE DIRECTORY DATA\_PUMP\_DIR AS 'D:\KURSORACLE\ARHIEVE';  GRANT READ, WRITE ON DIRECTORY DATA\_PUMP\_DIR TO PUBLIC;  CREATE OR REPLACE PROCEDURE export\_role\_to\_json  (  p\_file\_name IN VARCHAR2  )  AS  l\_file UTL\_FILE.FILE\_TYPE;  l\_buffer VARCHAR2(32767);  CURSOR c\_role IS  SELECT login, status, password  FROM role;  BEGIN  l\_file := UTL\_FILE.FOPEN('DATA\_PUMP\_DIR', p\_file\_name, 'w');  UTL\_FILE.PUT\_LINE(l\_file, '[');  FOR rec IN c\_role  LOOP  l\_buffer := ' {' ||  '"login": "' || rec.login || '",' ||  '"status": "' || rec.status || '",' ||  '"password": "' || rec.password || '"' ||  '}';  UTL\_FILE.PUT\_LINE(l\_file, l\_buffer);  IF c\_role%ISOPEN THEN  UTL\_FILE.PUT\_LINE(l\_file, ',');  END IF;  END LOOP;  UTL\_FILE.PUT\_LINE(l\_file, ']');  UTL\_FILE.FCLOSE(l\_file);  EXCEPTION  WHEN OTHERS THEN  IF UTL\_FILE.IS\_OPEN(l\_file) THEN  UTL\_FILE.FCLOSE(l\_file);  END IF;  RAISE;  END;  BEGIN  export\_role\_to\_json('role.json');  END;  CREATE OR REPLACE PROCEDURE import\_role\_from\_json  (  p\_file\_name IN VARCHAR2  )  AS  l\_file UTL\_FILE.FILE\_TYPE;  l\_buffer VARCHAR2(32767);  l\_json\_str VARCHAR2(32767);  l\_src NUMBER DEFAULT 1;  l\_json\_value JSON\_VALUE;  l\_json\_object JSON\_OBJECT\_T;  l\_login NUMBER;  l\_status VARCHAR2(50);  l\_password VARCHAR2(50);  BEGIN  l\_file := UTL\_FILE.FOPEN('DATA\_PUMP\_DIR', p\_file\_name, 'r');  LOOP  BEGIN  UTL\_FILE.GET\_LINE(l\_file, l\_buffer);  l\_json\_str := l\_json\_str || l\_buffer;  EXCEPTION  WHEN NO\_DATA\_FOUND THEN  EXIT;  END;  END LOOP;  l\_json\_value := NEW JSON\_VALUE(l\_json\_str);  FOR i IN 1..l\_json\_value.get\_JsonArray().get\_count()  LOOP  l\_json\_object := l\_json\_value.get\_JsonArray().get\_object(i);  l\_login := l\_json\_object.get\_number('login');  l\_status := l\_json\_object.get\_string('status');  l\_password := l\_json\_object.get\_string('password');  INSERT INTO role (login, status, password)  VALUES (l\_login, l\_status, l\_password);  END LOOP;  UTL\_FILE.FCLOSE(l\_file);  EXCEPTION  WHEN OTHERS THEN  IF UTL\_FILE.IS\_OPEN(l\_file) THEN  UTL\_FILE.FCLOSE(l\_file);  END IF;  RAISE;  END;  ALTER SESSION SET "\_ORACLE\_SCRIPT"=true; |

Листинг В.1 – Экспорт/импорт в json

|  |
| --- |
| --гость в приложение  --Получение списка всех отелей  exec GET\_ALL\_HOTELS();  --вывод рейтинга определенного отеля:  EXEC get\_hotel\_rating\_in\_city('Лесной Замок', 'Денвер');  --Получение списка всех номеров в конкретном отеле  EXEC GET\_ROOMS\_IN\_HOTEL(2);  --вывод информации о свободных номерах в конкретном отеле  EXEC GET\_AVAILABLE\_ROOMS\_IN\_HOTEL(2);  --вывод доступности комнаты:  EXEC get\_room\_availability(102, 2);  --вывод типа комнаты:  EXEC get\_room\_type(201, 2);  --вывод цены комнаты:  EXEC get\_room\_price(201, 2);  --user  --бронирование номера в отеле:  EXEC BOOK\_ROOM(1, 2, 10, TO\_DATE('2023-07-01', 'YYYY-MM-DD'), TO\_DATE('2023-07-10', 'YYYY-MM-DD'));  --получения счета за собственное бронирование:  exec GET\_SUM\_PAY\_BY\_RESERVATION(2);  --отмена собственного бронирования:  --до  exec GET\_MY\_RESERVATIONS(76207);  --отмена  EXEC CANCEL\_RESERVATION(1);  --после  exec GET\_MY\_RESERVATIONS(76207);  --admin  --вывод информации о пользователе  exec GET\_ALL\_USERS();  --Получение списка всех отелей  exec GET\_ALL\_HOTELS();  -- добавление нового отеля в систему  EXEC ADD\_HOTEL(100010, 'Веста', 'Брест', 'ул. Ковалева, 12', '5 звезд');  --вывод инфы о платежах  exec VIEW\_PAYMENTS();  --подтверждение платежа:  EXEC CONFIRM\_PAYMENT(11);  --изменение информации о номере:  EXEC update\_room\_info(1, 2, 'Двухместный', 100, 'Свободен', 201);  --удаление комнаты  EXEC delete\_room(2);  --добавления нового номера в отель:  EXEC ADD\_ROOM(100010, 5, 'Люкс', 500, 'Доступен', 501);  --удаление пользователя по логину:  EXEC delete\_user\_by\_login('login\_10');  --изменение данных пользователя:  --с проверкой почты  EXEC update\_user\_by\_login('login\_10', 'Joh', 'Doe', 'john.doe@example.com', '+1234567890');  --удаления отеля из системы  EXEC DELETE\_HOTEL(10);  --обновление рейтинга отеля  EXEC UPDATE\_HOTEL\_RATING(2, '4 звезды');  --статистика бронирований  exec GET\_RESERVATION\_STATISTICS; |

Листинг В.2 – Сценарий использования