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

Факультет Информационных технологий
Кафедра Информационных систем и технологий
Специальность 6-05-0611-01 «Информационные системы и технологии»

# ПОЯСНИТЕЛЬНАЯ ЗАПИСКА КУРСОВОГО ПРОЕКТА

по дисциплине «Базы данных»
Тема: «База данных сервиса бронирования номеров в отеле»

**Исполнитель**
студент 2 курса 1 группы _____ Е. И. Харченко
подпись, дата

**Руководитель**
ассистент _____ В. С. Кантарович
должность, учен. степень, ученое звание подпись, дата

Допущен(а) к защите _____ _____
_____ _____
_____ _____
дата, подпись

Курсовой проект защищен с оценкой _____

Руководитель _____ _____ В. С. Кантарович
подпись дата инициалы и фамилия

Минск 2025

# Содержание

# Введение

Данный проект посвящен разработке базы данных для сервиса онлайн-бронирования номеров в отеле. В условиях растущей популярности онлайн-платформ для планирования поездок, такие системы становятся неотъемлемой частью современного рынка гостиничных услуг. Они позволяют пользователям быстро и удобно искать, сравнивать и бронировать номера, что значительно упрощает процесс выбора подходящего жилья.

Актуальность проекта обусловлена необходимостью создания надежной и эффективной базы данных, которая будет поддерживать функциональность сервиса бронирования в одном отеле.

Задачей курсового проекта является обзор и анализ аналогичных решений, определение количества ролей, разработка вариантов использования для каждой роли, проектирование таблиц и логической схемы базы данных, разработка всех необходимых объектов для работы в базе данных, а также проведение тестирования работоспособности базы данных.

Целевая аудитория проекта включает частных пользователей, нуждающихся в жилье для краткосрочного или длительного проживания, а также корпоративные клиенты, бронирующие номера для деловых поездок

Проект будет реализован с использованием СУБД Oracle, что обеспечит высокую производительность и надежность системы.

# 1 Постановка задачи и обзор аналогичных решений

## 1.1 Аналитический обзор аналогов

В рамках курсового проекта по разработке базы данных для сервиса онлайн-бронирования номеров в отеле планируется провести аналитический обзор существующих платформ для бронирования отелей. Этот обзор поможет лучше понять, какие функции и возможности могут быть реализованы в системе, а также определить инструменты и подходы, которые можно применить для создания базы данных, соответствующей требованиям и особенностям сервиса онлайн-бронирования.

### 1.1.1 Аналог «Booking.com»

Booking.com [1] представляет собой один из самых популярных и масштабных онлайн-сервисов для бронирования жилья в мире. Платформа предлагает пользователям широкий спектр вариантов размещения, включая отели, хостелы, апартаменты, виллы и даже уникальные места проживания, такие как домики на деревьях или плавучие дома.

Сервис отличается мощной и гибкой системой поиска, позволяющей пользователям искать жилье по множеству критериев. Помимо стандартных параметров, таких как географическое местоположение, даты заезда и выезда, количество гостей, Booking.com предлагает расширенные фильтры. Пользователи могут сортировать результаты по популярности, цене, рейтингу. Особенно полезна возможность фильтрации по конкретным удобствам, что позволяет найти жилье, максимально соответствующее индивидуальным потребностям путешественника.

Booking.com уделяет большое внимание предоставлению подробной информации о каждом объекте размещения. Страницы апартаментов содержат обширные галереи фотографий, детальные описания номеров и удобств, информацию о правилах проживания и политике отмены бронирования.

Для повышения лояльности клиентов Booking.com разработал программу Genius, которая предоставляет дополнительные скидки и преимущества частым пользователям сервиса.

### 1.1.2 Аналог «Expedia»

Expedia [2] выделяется среди конкурентов как комплексная платформа для планирования путешествий, предлагающая пользователям возможность бронирования не только жилья, но и других важных элементов поездки. Сервис позволяет в одном месте забронировать авиабилеты, отели, арендовать автомобиль, приобрести билеты на круизы и различные мероприятия.

Ключевой особенностью Expedia является возможность создания полноценных туристических пакетов. Пользователи могут комбинировать различные элементы своего путешествия, получая при этом значительные скидки по сравнению с отдельным бронированием каждой услуги.

Система поиска Expedia отличается гибкостью и удобством. Помимо стандартного поиска по направлениям и датам, сервис предлагает функцию «куда

угодно», которая помогает путешественникам найти вдохновение для своей следующей поездки. Платформа также предоставляет полезные инструменты планирования, такие как календарь цен, позволяющий выбрать оптимальные даты для путешествия с точки зрения стоимости.

Expedia уделяет большое внимание предоставлению подробной информации о направлениях. На страницах городов и стран пользователи могут найти информацию о погоде, лучшем времени для посещения, основных достопримечательностях и популярных активностях. Это помогает путешественникам лучше подготовиться к поездке и спланировать свой маршрут.

Как и Booking.com, Expedia имеет развитую систему отзывов и рейтингов. Пользователи могут ознакомиться с отзывами реальных путешественников не только об отелях, но и об авиакомпаниях, прокатных компаниях и других услугах. Это помогает сделать более информированный выбор при планировании путешествия.

Для повышения лояльности клиентов Expedia разработала программу Expedia Rewards. Участники программы получают баллы за бронирования, которые затем можно использовать для оплаты будущих путешествий. Это стимулирует пользователей возвращаться на платформу для новых бронирований.

### 1.1.3 Аналог «101hotels.com»

101hotels.com [3] представляет собой специализированный сервис бронирования отелей, ориентированный преимущественно на российский рынок и страны СНГ. В отличие от глобальных платформ, таких как Booking.com и Expedia, этот сервис фокусируется на предоставлении информации о доступных отелях и других вариантах жилья в пределах конкретного географического региона.

Ключевой особенностью 101hotels.com является простой и интуитивно понятный интерфейс, разработанный с учетом потребностей и предпочтений локальных пользователей. Сервис предлагает удобную систему поиска, позволяющую быстро находить подходящие варианты размещения. Пользователи могут легко фильтровать результаты по различным параметрам, таким как цена, рейтинг, тип жилья и наличие определенных удобств.

Особое внимание 101hotels.com уделяет предоставлению детальной информации о каждом объекте размещения. Страницы отелей содержат обширные галереи фотографий, позволяющие пользователям получить визуальное представление о выбранном варианте жилья. Кроме того, сервис предоставляет подробные описания номеров, включая информацию о площади, меблировке и доступных удобствах. Важной особенностью является наличие информации о правилах проживания, условиях бронирования и отмены заказа, что помогает путешественникам избежать недоразумений и неприятных сюрпризов.

101hotels.com также уделяет большое внимание предоставлению информации о специальных предложениях и акциях. Сервис регулярно обновляет раздел с выгодными предложениями, что позволяет пользователям находить наиболее экономичные варианты размещения. Это особенно привлекательно для путешественников, планирующих поездку с ограниченным бюджетом.

101hotels.com также уделяет большое внимание предоставлению информации о специальных предложениях и акциях. Сервис регулярно обновляет раздел с выгодными предложениями, что позволяет пользователям находить наиболее экономичные варианты размещения. Это особенно привлекательно для путешественников, планирующих поездку с ограниченным бюджетом.

Важным элементом 101hotels.com является система отзывов и рейтингов. Пользователи могут ознакомиться с мнениями предыдущих гостей, что помогает сформировать более объективное представление о качестве предоставляемых услуг. Отзывы часто содержат полезную информацию о расположении отеля, качестве обслуживания и соотношении цены и качества.

Отличительной чертой 101hotels.com является наличие информационных разделов о городах и регионах России и стран СНГ. Эти разделы содержат полезную информацию для путешественников, включая описание основных достопримечательностей, рекомендации по планированию поездки и советы по выбору оптимального времени для посещения.

### 1.1.4 Аналог «Airbnb»

Airbnb [4] представляет собой инновационную платформу, которая произвела революцию в индустрии путешествий, предложив альтернативу традиционным вариантам размещения. В отличие от классических сервисов бронирования отелей, Airbnb позволяет пользователям арендовать жилье непосредственно у владельцев, что открывает доступ к уникальным вариантам проживания – от городских квартир до экзотических домиков на деревьях или плавучих домов.

Ключевой особенностью Airbnb является разнообразие предлагаемых вариантов жилья. Платформа предоставляет возможность выбора не только между различными типами помещений (отдельная комната, целый дом, уникальное жилье), но и между различными стилями проживания. Это позволяет путешественникам найти жилье, которое наилучшим образом соответствует их предпочтениям и бюджету.

Система поиска Airbnb отличается гибкостью и удобством использования. Пользователи могут фильтровать результаты по множеству параметров, включая тип жилья, ценовой диапазон, удобства, и даже по уникальным характеристикам, таким как «с видом на океан» или «экологичное жилье».

Система поиска Airbnb отличается гибкостью и удобством использования. Пользователи могут фильтровать результаты по множеству параметров, включая тип жилья, ценовой диапазон, удобства, и даже по уникальным характеристикам, таким как «с видом на океан» или «экологичное жилье». Кроме того, Airbnb предлагает функцию «Впечатления», которая позволяет путешественникам бронировать не только жилье, но и различные активности и экскурсии, проводимые местными жителями.

Особое внимание Airbnb уделяет созданию сообщества и установлению доверительных отношений между хозяевами и гостями. Каждый пользователь имеет свой профиль с верификацией личности, что повышает уровень безопасности и доверия. Система двусторонних отзывов позволяет как гостям, так и хозяевам

оценивать друг друга, что способствует поддержанию высокого качества услуг на платформе.

Airbnb также предлагает инновационный подход к презентации жилья. Хозяева могут создавать виртуальные туры. Это помогает потенциальным гостям получить максимально полное представление о жилье перед бронированием.

Важной особенностью Airbnb является функция «Суперхозяин». Этот статус присваивается хозяевам, которые зачастую получают высокие оценки от гостей и отличаются особым гостеприимством. Это помогает путешественникам быстро находить проверенные и качественные варианты размещения.

### 1.2 Разработка функциональных требований, определение вариантов использования

Целью курсового проекта является разработка базы данных для сервиса бронирования номеров в отеле. В рамках проекта предстоит выполнить следующие задачи:

– Разработка структуры базы данных: будут созданы таблицы для хранения информации о номерах, пользователях, бронированиях и дополнительных услугах. Также будут определены связи между таблицами для обеспечения целостности данных.

– Обеспечение безопасности данных и разделение прав доступа: права доступа будут разделены на три категории пользователей. Пользователи смогут регистрироваться, просматривать и оставлять отзывы, а также управлять своими бронированиями, включая возможность бронировать и отменять номера, просматривать статус бронирований и добавлять услуги. Менеджеры отеля получат возможность управлять бронированиями, изменять цены на номера, а также добавлять и отменять услуги. Специальные администраторы смогут управлять менеджерами, номерами и услугами, а также блокировать и разблокировать пользователей.

– Создание процедур для импорта и экспорта данных: будут реализованы процедуры для импорта данных из JSON файлов и экспорта данных в формат JSON.

– Разработка процедур и функций для взаимодействия с базой данных: будут созданы процедуры и функции, обеспечивающие выполнение всех необходимых операций, таких как добавление номеров, бронирование и отмена бронирований, управление номерным фондом, изменение цен на номера, а также блокировка и разблокировка пользователей. Эти процедуры позволят реализовать функциональность, соответствующую ролям пользователей.

Выполнение этих задач позволит создать функциональную и эффективную базу данных, которая будет соответствовать требованиям современного сервиса онлайн-бронирования.

### 1.3 Вывод по разделу

В данном разделе была установлена цель курсового проекта – разработка базы данных сервиса для бронирования номеров в отеле.

Были рассмотрены аналогичные решения интернет-сервисов для бронирования номеров в отеле, были определены задачи для реализации, включая разработку структуры базы данных, обеспечение безопасности данных и разделение прав доступа, создание процедур для импорта и экспорта данных, а также разработку процедур и функций для взаимодействия с базой данных. Помимо этого, были определены три категории ролей: пользователь, менеджер отеля и специальный администратор.

# 2 Разработка архитектуры проекта

## 2.1 Диаграммы UML, взаимосвязь всех компонентов

Use Case диаграммы (диаграммы вариантов использования) – это тип диаграмм в Unified Modeling Language (UML), которые используются для моделирования взаимодействий между пользователями (акторами) и системой. Они помогают визуализировать функциональные требования системы, показывая, какие действия могут выполнять пользователи и как система реагирует на эти действия. Данный курсовой проект подразумевает собой 3 роли: пользователь, менеджер отеля и специальный администратор.

### 2.1.1 Определение вариантов использования для роли «Пользователь»

На рисунке 2.1 предоставлена диаграмма вариантов использования для роли «Пользователь».

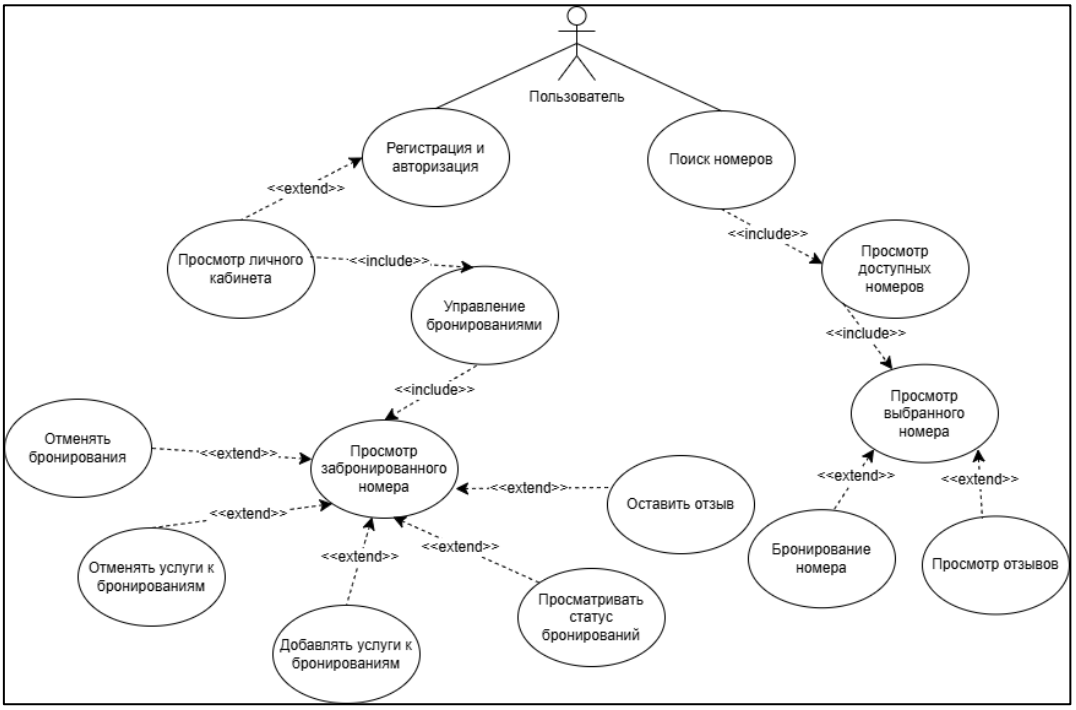



Рисунок 2.1 – Диаграмма вариантов использования для роли «Пользователь»

В системе предусмотрена регистрация и авторизация пользователей. После входа доступен личный кабинет с функционалом управления бронированиями – их просмотром, отменой, добавлением и удалением дополнительных услуг.

Пользователи могут искать номера, просматривать доступные варианты с подробным описанием, выбирать подходящий и оформлять бронь.

Для информирования гостей реализован просмотр отзывов о номерах, а также возможность оставлять собственные после проживания.

### 2.1.2 Определение вариантов использования для роли «Менеджер отеля»

На рисунке 2.2 предоставлена диаграмма вариантов использования для роли «Менеджер отеля».

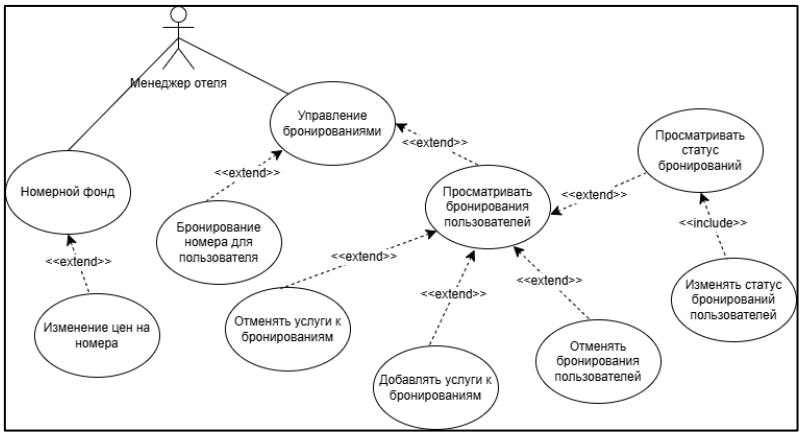



Рисунок 2.2 – Диаграмма вариантов использования для роли «Менеджер отеля»

Менеджер имеет доступ к управлению бронированиями, включая просмотр всех текущих резерваций и их статусов. В его обязанности входит изменение статусов бронирований, например, подтверждение или аннулирование.

Ценообразование также в зоне ответственности менеджера – он может корректировать стоимость номеров в зависимости от сезона или спроса. Для удобства пользователей менеджер вправе добавлять или убирать дополнительные услуги к существующим бронированиям.

### 2.1.3 Определение вариантов использования для роли «Специальный администратор»

На рисунке 2.3 предоставлена диаграмма вариантов использования для роли «Специальный администратор».

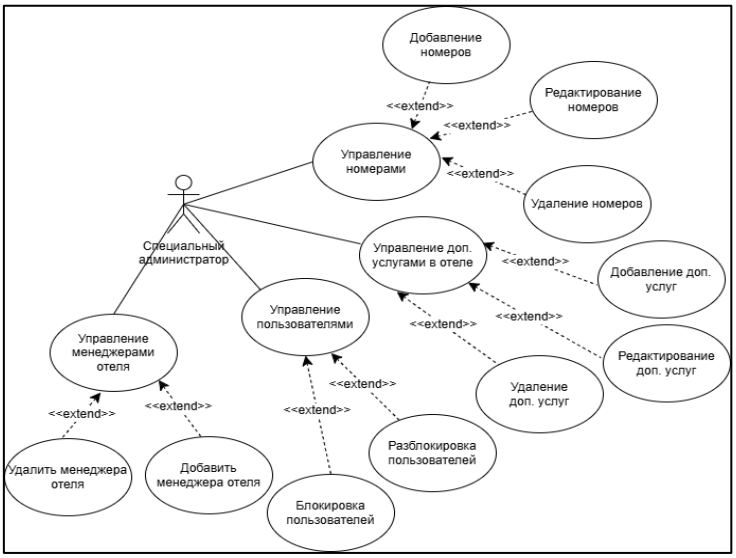



Рисунок 2.3 – Диаграмма вариантов использования для роли «Спец. админ.»

Специальный администратор обладает полным контролем над системой: управляет номерным фондом (добавление, редактирование и удаление номеров), а также дополнительными услугами отеля (создание и корректировка опций для гостей).

В его зоне ответственности работа с пользователями – блокировка и разблокировка аккаунтов при необходимости. Отдельное внимание уделено управлению менеджерами отеля: администратор может добавлять новых сотрудников или удалять их из системы, обеспечивая бесперебойную работу персонала.

## 2.2 Логическая схема базы данных

Логическая схема базы данных, её ограничения целостности, связи и поля представлена на рисунке 2.4.

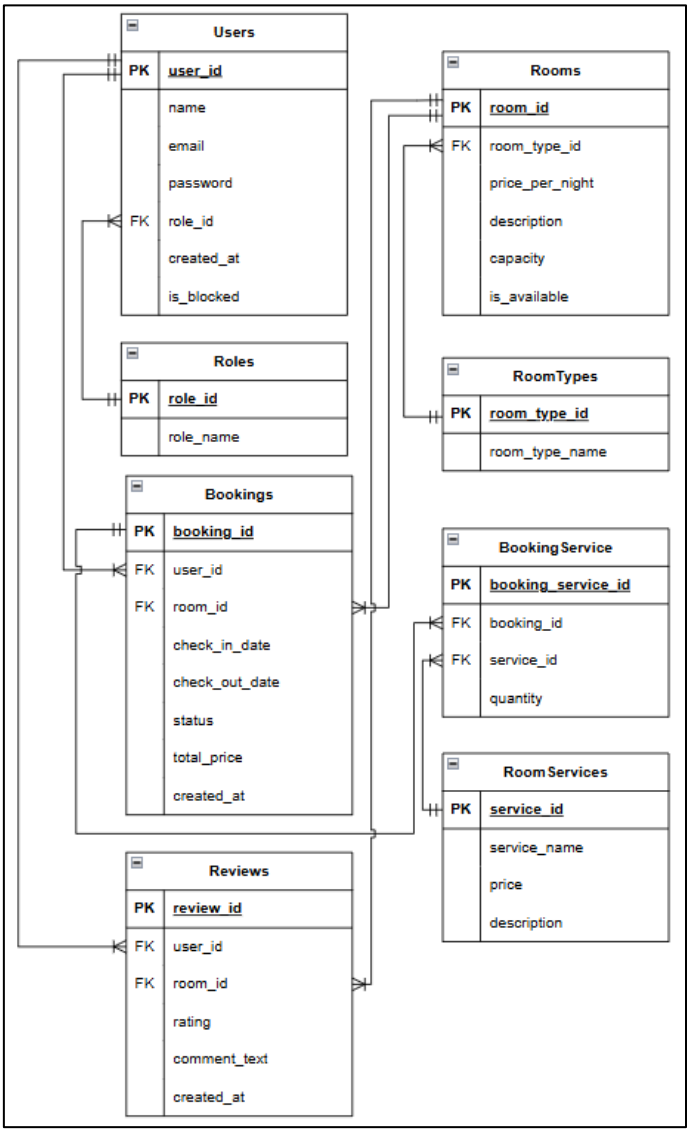



Рисунок 2.4 – Логическая схема базы данных

На данной логической схеме отображены 8 таблиц, которые будут созданы в рамках курсового проекта.

## 2.3 Описание информационных объектов и ограничений целостности

В таблице 2.1 описаны таблицы, которые будут созданы в базе данных, а в таблицах 2.2 – 2.9 приводится описание каждой таблицы, включая её поля и описание этих полей. В таблице 2.10 описаны связи между таблицами базы данных.

Таблица 2.1 – Сведения о таблицах базы данных

| Имя таблицы | Назначение таблицы |
|---|---|
| Users | Хранение информации о пользователях системы, включая их роли и основные данные. |
| Roles | Хранение ролей пользователей (user, manager, admin). |
| Rooms | Информация о номерах отеля: тип, цена, описание и доступность. |
| RoomTypes | Хранение типов номеров (например, стандарт, люкс, президентский). |
| Bookings | Информация о бронированиях номеров, включая даты заезда/выезда и статус. |
| Reviews | Отзывы пользователей об номерах с рейтингом и комментариями |
| RoomServices | Услуги, предоставляемые отелем (например, завтрак, Wi-Fi). |
| BookingServices | Связь бронирований с дополнительными услугами, включая количество. |

Здесь описано назначение всех таблиц, которые будут реализованы в рамках данного курсового проекта.

Таблица 2.2 – Таблица «Users»

| Поле | Назначение поля |
|---|---|
| user_id | Уникальный идентификатор пользователя. |
| name | Полное имя пользователя. |
| email | Электронная почта пользователя. |
| password | Хэшированный пароль пользователя |
| role_id | Идентификатор роли пользователя |
| created_at | Дата и время создания учетной записи |
| is_blocked | Статус блокировки учетной записи (0 — активна, 1 — заблокирована) |

Здесь описаны поля и назначение каждого из поля в таблице «Users».

Таблица 2.3 – Таблица «Roles»

| Поле | Назначение поля |
|---|---|
| role_id | Уникальный идентификатор роли |
| role_name | Название роли: user, manager, admin |

Здесь описаны поля и назначение каждого из поля в таблице «Roles».

Таблица 2.4 – Таблица «Rooms»

| Поле | Назначение поля |
|---|---|
| room_id | Уникальный идентификатор номера |
| room_type_id | Идентификатор типа номера |
| price_per_night | Цена за ночь |
| description | Описание номера |
| capacity | Вместимость номера |
| is_available | Статус доступности номера |

Здесь описаны поля и назначение каждого из поля в таблице «Rooms».

Таблица 2.5 – Таблица «RoomTypes»

| Поле | Назначение поля |
|---|---|
| room_type_id | Уникальный идентификатор типа номера |
| room_type_name | Название типа номера |

Здесь описаны поля и назначение каждого из поля в таблице «RoomTypes».

Таблица 2.6 – Таблица «Bookings»

| Поле | Назначение поля |
|---|---|
| booking_id | Уникальный идентификатор бронирования |
| user_id | Идентификатор пользователя |
| room_id | Идентификатор номера |
| check_in_date | Дата заезда |
| check_out_date | Дата выезда |
| status | Идентификатор статуса бронирования |
| total_price | Общая стоимость бронирования ((цена номера * кол-во ночей) + сумма услуг) |
| created_at | Дата и время создания бронирования |

Здесь описаны поля и назначение каждого из поля в таблице «Bookings».

Таблица 2.7 – Таблица «Reviews»

| Поле | Назначение поля |
|---|---|
| review_id | Уникальный идентификатор отзыва |
| user_id | Идентификатор автора отзыва |
| room_id | Идентификатор номера |
| rating | Оценка номера |
| comment | Текстовый комментарий отзыва |
| created_at | Дата и время создания отзыва |

Здесь описаны поля и назначение каждого из поля в таблице «Reviews».

Таблица 2.8 – Таблица «RoomServices»

| Поле | Назначение поля |
|---|---|
| service_id | Уникальный идентификатор услуги |
| service_name | Название услуги |
| price | Стоимость услуги |
| description | Описание услуги |

Здесь описаны поля и назначение каждого из поля в таблице «RoomServices».

Таблица 2.9 – Таблица «BookingServices»

| Поле | Назначение поля |
|---|---|
| booking_service_id | Уникальный идентификатор услуги |
| booking_id | Идентификатор бронирования |
| service_id | Название дополнительной услуги |
| quantity | Количество заказанных услуг |

Здесь описаны поля и назначение каждого из поля в таблице «BookingServices».

Таблица 2.10 – Сведения о связях между таблицами базы данных

| Таблица PK (поле PK) | Таблица FK (поле FK) | Тип связи | Описание связи |
|---|---|---|---|
| Roles (role_id) | Users (role_id) | Один ко многим | Роль может быть назначена многим пользователям |
| Users (user_id) | Bookings (user_id) | Один ко многим | Пользователь может иметь несколько бронирований |
| Users (user_id) | Reviews (user_id) | Один ко многим | Пользователь может оставить несколько отзывов |

| Таблица PK (поле PK) | Таблица FK (поле FK) | Тип связи | Описание связи |
|---|---|---|---|
| RoomTypes (room_type_id) | Rooms (room_type_id) | Один ко многим | Тип номера может быть присвоен нескольким номерам |
| Rooms (room_id) | Bookings (room_id) | Один ко многим | Номер может быть забронирован много раз (в разные даты) |
| Rooms (room_id) | Reviews (room_id) | Один ко многим | На номер может быть несколько отзывов |
| Bookings (booking_id) | BookingServices (booking_id) | Один ко многим | Бронирование может включать несколько услуг |
| RoomServices (service_id) | BookingServices (service_id) | Один ко многим | Услуга может быть добавлена к нескольким бронированиям |

Здесь описаны связи между полями в таблицах базы данных с приведением описания этой связи и его типа.

### 2.4 Вывод по разделу

В данном разделе была разработаны UML диаграммы вариантов использования сервиса для каждой из трёх ролей, была разработана логическая схема базы данных с подробным описанием таблиц, её полей и связей между другими таблицами.

# 3 Разработка модели базы данных

## 3.1 Создание необходимых объектов

### 3.1.1 Таблицы

Было реализовано 8 таблиц, которые были описаны во 2 разделе:

– Users: хранит информацию о пользователях системы, включая их роли и основные данные;

– Roles: хранит роли пользователей (user, manager, admin);

– Rooms: хранит информацию о номерах отеля;

– RoomTypes: хранит типы номеров;

– Bookings: хранит информацию о бронированиях номеров;

– Reviews: хранит отзывы пользователей о номерах с рейтингом и комментариями;

– RoomServices: хранит услуги, предоставляемые отелем;

– BookingServices: хранит связь бронирований с дополнительными услугами, включая их количество.

### 3.1.2 Процедуры

Было реализовано 17 процедур для пользователя, менеджера отеля и специального администратора:

– register_user: для регистрации пользователя;

– book_room: для бронирования номера;

– cancel_booking: для отмены бронирования;

– add_review: для добавления отзыва;

– add_service: для добавления услуг к бронированию;

– change_booking_status: для изменения статуса бронирования;

– update_room_price: для обновления цен номеров;

– remove_service_from_booking: для удаления услуг к бронированию;

– toggle_user_block: для блокировки/разблокировки пользователей;

– assign_manager_role: для назначения менеджера отеля;

– delete_user: для удаления пользователя;

– add_room: для добавления нового номера;

– update_room: для редактирования номера;

– delete_room: для удаления номера;

– add_new_service: для добавления новой услуги к номерам;

– update_service: для редактирования услуг к номерам;

– delete_service: для удаления услуг к номерам.

SQL-запрос одной из процедур представлен в листинге 3.1.

```
CREATE OR REPLACE PROCEDURE register_user(
    p_name IN VARCHAR2,
    p_email IN VARCHAR2,
```

```
    p_password IN VARCHAR2,
    p_role_name IN VARCHAR2 DEFAULT 'Пользователь',
    p_user_id OUT NUMBER
) AS
    v_role_id NUMBER;
BEGIN
    SELECT role_id INTO v_role_id
    FROM Roles
    WHERE LOWER(role_name) = LOWER(p_role_name);

    INSERT INTO Users (name, email, password, role_id)
    VALUES (p_name, p_email, p_password, v_role_id)
    RETURNING user_id INTO p_user_id;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN ROLLBACK; RAISE;
END;
/
```

Листинг 3.1 – SQL-запрос для создания процедуры «register_user»

Данная хранимая процедура регистрирует нового пользователя с указанными данными (имя, email, пароль) и присваивает ему роль «Пользователь», возвращая сгенерированный ID пользователя через выходной параметр. Запросы для создания остальных процедур представлены в приложении Б.

### 3.1.3 Функции

Было реализовано 4 функции для пользователя, менеджера отеля и специального администратора:

– get_room_reviews: для получения отзывов к номеру;

– get_available_rooms: для получения списка свободных номеров;

– get_user_bookings: для получения списка бронирований конкретного пользователя;

– get_all_services: для получения списка дополнительных услуг к бронированиям.

SQL-запрос создания одной из этих функций представлен в листинге 3.2.

```
CREATE OR REPLACE FUNCTION get_room_reviews(
    p_room_id IN NUMBER,
    p_user_id IN NUMBER DEFAULT NULL
) RETURN SYS_REFCURSOR AS
    v_cursor SYS_REFCURSOR;
    v_is_blocked NUMBER := 0;
BEGIN
    IF p_user_id IS NOT NULL THEN
        BEGIN
            SELECT is_blocked INTO v_is_blocked
            FROM Users
```

```
            WHERE user_id = p_user_id;

            IF v_is_blocked = 1 THEN
                DBMS_OUTPUT.PUT_LINE('Предупреждение: Пользователь
заблокирован, но может просматривать отзывы');
            END IF;
        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                NULL;
        END;
    END IF;

    OPEN v_cursor FOR
    SELECT u.name, r.rating, r.comment_text, r.created_at
    FROM Reviews r
    JOIN Users u ON r.user_id = u.user_id
    WHERE r.room_id = p_room_id
    ORDER BY r.created_at DESC;
    RETURN v_cursor;
END;
/
```

Листинг 3.2 – SQL-запрос для создания функций базы данных

Данная функция возвращает курсор с отзывами о комнате (имя автора, оценку, текст комментария и дату создания), отсортированными по дате в обратном порядке, для указанного идентификатора комнаты. Запросы для создания остальных функций представлен в приложении В.

### 3.1.4 Триггеры

Было реализовано 6 триггеров, которые охватывают все взаимодействие с базой данных:

– trg_check_booking_dates: для проверки даты заезда и выезда при бронировании;

– trg_check_room_availability: для проверки доступности номера;

– trg_check_review: для проверки отзывов;

– trg_update_booking_total: для пересчета стоимости бронирования при изменении услуг;

– trg_prevent_room_deletion: для проверки перед удалением номера;

– trg_prevent_service_deletion: для проверки перед удалением услуги.

SQL-запрос создания одного из триггеров представлен в листинге 3.3.

```
CREATE OR REPLACE TRIGGER trg_prevent_room_deletion
BEFORE DELETE ON Rooms
FOR EACH ROW
DECLARE
    v_active_bookings NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_active_bookings
```

```
    FROM Bookings
    WHERE room_id = :OLD.room_id
    AND status NOT IN ('Отменено', 'Выполнено');

    IF v_active_bookings > 0 THEN
        RAISE_APPLICATION_ERROR(-20017, 'Нельзя удалить номер с актив-
ными бронированиями'); END IF; END;
```

Листинг 3.3 – SQL-запрос для создания триггера базы данных

Этот триггер предотвращает удаление номера комнаты из таблицы Rooms, если у него есть активные бронирования (не отменённые и не выполненные), вызывая ошибку с сообщением в случае попытки удаления. Запросы для создания остальных триггеров представлены в приложении Г.

### 3.1.5 Индексы

В рамках данного курсового проекта был разработан один индекс – idx_users_is_blocked для поиска незаблокированных пользователей. SQL-запрос создания данного индекса представлен в листинге 3.4.

```
    CREATE INDEX idx_users_is_blocked ON Users(is_blocked) TABLESPACE
hotel_db_data;
```

Листинг 3.4 – SQL-запрос для создания индекса

Данный оператор создает индекс idx_users_is_blocked для столбца is_blocked в таблице Users, размещая его в табличном пространстве hotel_db_data, чтобы ускорить поиск пользователей по их статусу блокировки.

## 3.2 Описание используемой технологии

В Oracle технология email-уведомлений о событиях в базе данных реализуется с помощью триггеров, хранимых процедур и утилиты UTL_SMTP. Триггеры запускаются при вставке или обновлении строк в целевой таблице (например, Bookings), затем вызывают хранимую процедуру отправки письма, в которой формируется тело сообщения и указывается SMTP-сервер, адрес получателя и другие параметры.

Для обеспечения защищённой передачи данных и работы с внешним SMTP-сервером Gmail была проведена комплексная предварительная настройка. Сначала был создан Oracle Wallet, содержащий SSL-сертификаты, необходимые для установления TLS-соединения с сервером smtp.gmail.com. Wallet был размещён в указанной директории на сервере и подключён к Oracle с помощью параметров wallet_path и wallet_password в функции UTL_SMTP.OPEN_CONNECTION.

Затем была выполнена настройка прав доступа через Access Control List (ACL), чтобы разрешить подключение к SMTP-серверу из базы данных. Были заданы разрешения для пользователя базы данных на выполнение сетевых операций

по нужному хосту и порту (smtp.gmail.com:587), включая разрешение на использование протоколов TLS и SMTP.

Была создана процедура send_gmail_fixed, реализующая отправку письма через UTL_SMTP с поддержкой STARTTLS, а также выполнена аутентификация через Gmail с использованием специального пароля приложений. Эта процедура затем вызывалась внутри триггера, настроенного на таблицу Bookings, что позволило автоматически отправлять email-уведомления при создании или обновлении записей.

Работоспособность данной технологии представлена на рисунке 3.1.

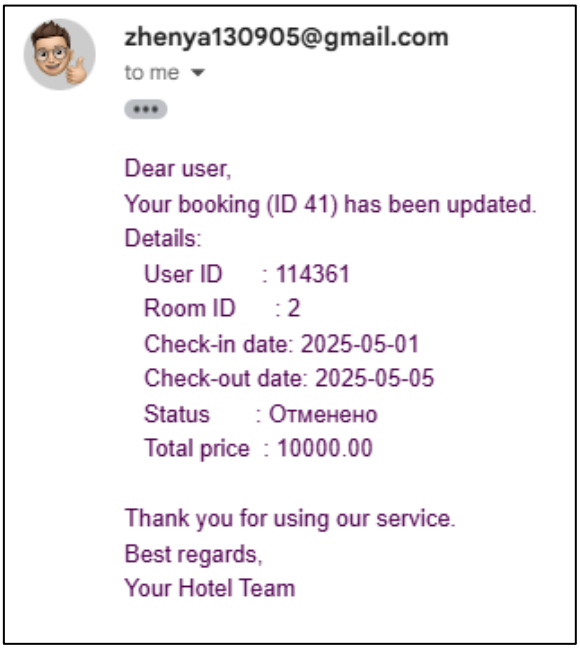



Рисунок 3.1 – Отправленное письмо об изменении статуса бронирования

SQL-запрос создания процедуры отправки email-уведомлений представлен в листинге 3.5.

```
CREATE OR REPLACE PROCEDURE send_gmail(
  p_to      VARCHAR2,
  p_subject VARCHAR2,
  p_body    VARCHAR2
) AS
  v_conn      UTL_SMTP.connection;
  v_crlf      VARCHAR2(2) := CHR(13) || CHR(10);
  v_username  VARCHAR2(100) := 'zhenya130905@gmail.com';
  v_password  VARCHAR2(100) := '';
  v_from      VARCHAR2(100) := 'zhenya130905@gmail.com';
BEGIN
    v_conn := UTL_SMTP.OPEN_CONNECTION(
      host                          => 'smtp.gmail.com',
      port                          => 587,
      tx_timeout                    => 30,
        wallet_path             => 'file:D:/oracle/wallets/server_wal-
let.',
      wallet_password               => 'SecretPwd123',
      secure_connection_before_smtp => FALSE
```

```
    );
    UTL_SMTP.EHLO(v_conn, 'smtp.gmail.com');
    UTL_SMTP.STARTTLS(v_conn);
    UTL_SMTP.EHLO(v_conn, 'smtp.gmail.com');
    UTL_SMTP.AUTH(v_conn,           v_username,           v_password,
UTL_SMTP.ALL_SCHEMES);

  UTL_SMTP.MAIL(v_conn, v_from);
  UTL_SMTP.RCPT(v_conn, p_to);

    UTL_SMTP.OPEN_DATA(v_conn);

    UTL_SMTP.WRITE_DATA(v_conn,
      'From: ' || v_from || v_crlf ||
      'To: ' || p_to || v_crlf ||
      'Subject:   =?UTF-8?B?'   ||   UTL_ENCODE.TEXT_ENCODE(p_subject,
'UTF8', UTL_ENCODE.BASE64) || '?=' || v_crlf ||
      'Content-Type: text/plain; charset=UTF-8' || v_crlf ||
      'Content-Transfer-Encoding: base64' || v_crlf || v_crlf
    );

    UTL_SMTP.WRITE_RAW_DATA(v_conn,
      UTL_ENCODE.BASE64_ENCODE(UTL_RAW.CAST_TO_RAW(CONVERT(p_body,
'UTF8')))
    );

    UTL_SMTP.CLOSE_DATA(v_conn);


  UTL_SMTP.QUIT(v_conn);

  DBMS_OUTPUT.PUT_LINE('Email успешно отправлен на ' || p_to);
EXCEPTION
  WHEN OTHERS THEN
    BEGIN
      UTL_SMTP.QUIT(v_conn);
    EXCEPTION
      WHEN OTHERS THEN NULL;
    END;
    DBMS_OUTPUT.PUT_LINE('Ошибка отправки: ' || SQLERRM);
END;
```

Листинг 3.5 – Запрос создания процедуры для отправки email-уведомлений

Листинг триггера для активации процедуры отправки представлен в приложении Ж.

### 3.3 Вывод по разделу

В данном разделе были описаны созданные объекты базы данных: таблицы, процедуры, функции, триггеры, индексы, а также была описана используемая технология – email уведомления о событиях в базе данных.

## 4 Установка, настройка и использование Oracle

Oracle Database – это мощная, высокопроизводительная и надежная реляционная система управления базами данных (СУБД), разработанная корпорацией Oracle. Она является одной из самых популярных и широко используемых коммерческих СУБД в мире, особенно в крупных корпоративных и государственных организациях.

### 4.1 Создание таблиц

Было реализовано 8 таблиц, которые были описаны в 3 разделе: Users, Roles, Rooms, RoomTypes, Bookings, Reviews, RoomServices, BookingServices. SQL-запрос для создания таблицы «Users» представлен в листинге 4.1.

```
CREATE TABLE Users (
    user_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR2(100) NOT NULL,
    email VARCHAR2(100) UNIQUE NOT NULL,
    password VARCHAR2(255) NOT NULL,
    role_id NUMBER NOT NULL,
    created_at TIMESTAMP DEFAULT SYSTIMESTAMP,
    is_blocked NUMBER(1) DEFAULT 0 CHECK (is_blocked IN (0, 1)),
    CONSTRAINT fk_user_role FOREIGN KEY (role_id) REFERENCES
Roles(role_id)
) TABLESPACE hotel_db_data;
```

Листинг 4.1 – SQL-запрос для создания таблицы «Users»

SQL-запросы для создания остальных таблиц представлены в приложении А.

### 4.2 Создание ролей для разграничения доступа

Было реализовано 3 роли: пользователь, менеджер отеля и администратор. Каждая из ролей получила доступ на запуск определенных процедур и функций, соответствующий этим ролям, на возможность просмотра и/или изменения таблиц, характерных для этих ролей. Были созданы пользователи, к которым были применены заранее созданные роли. Листинг SQL-запросов на создание ролей, предоставление прав на использование процедур и функций, предоставление прав на создание подключения, предоставление прав на работу с таблицами, а также создание пользователей представлен в приложении Е.

### 4.3 Описание процедур экспорта и импорта

Импорт и экспорт данных в формате JSON представляет собой возможность передачи информации между базой данных и различными внешними системами, а также удобный для чтения формат данных.

Экспорт данных выполняется при помощи пакета UTL_FILE, позволяющего работать с файлами. SQL-запрос для экспорта данных из таблицы «Bookings» представлен в листинге 4.2.

```
CREATE OR REPLACE PROCEDURE export_to_file IS
  l_json CLOB;
  l_file UTL_FILE.FILE_TYPE;
BEGIN
  SELECT JSON_ARRAYAGG(
           JSON_OBJECT(
             'bookingId' VALUE b.booking_id,
             'userId' VALUE b.user_id,
             'roomId' VALUE b.room_id,
             'checkInDate' VALUE TO_CHAR(b.check_in_date, 'YYYY-MM-
DD'),
             'checkOutDate' VALUE TO_CHAR(b.check_out_date, 'YYYY-MM-
DD'),
             'status' VALUE b.status,
             'totalPrice' VALUE b.total_price,
             'createdAt' VALUE TO_CHAR(b.created_at, 'YYYY-MM-
DD"T"HH24:MI:SS')
           )
         )
  INTO l_json
  FROM Bookings b;

  l_file := UTL_FILE.FOPEN('JSON_DIR', 'bookings.json', 'w', 32767);

  FOR i IN 0 .. CEIL(DBMS_LOB.GETLENGTH(l_json) / 32767) - 1 LOOP
    UTL_FILE.PUT(l_file, DBMS_LOB.SUBSTR(l_json, 32767, i * 32767 +
1));
  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;
```

Листинг 4.2 – SQL-запрос для создания процедуры для экспорта данных

Эта хранимая процедура экспортирует все данные о бронированиях из таблицы «Bookings» в JSON-файл «bookings.json» (в директории JSON_DIR), преобразуя записи в форматированный JSON-массив с ключами bookingId, userId, roomId и другими атрибутами, обрабатывая возможные ошибки и закрывая файл при завершении.

Импорт данных выполняется при помощи пакета UTL_FILE, позволяющего работать с файлами. SQL-запрос для импорта данных в таблицу «Bookings» представлен в приложении Д.

### 4.4 Тестирование производительности базы данных

Тестирование производительности базы данных является важным этапом проверки ее эффективности.

Тестирование было проведено для таблицы «Users». Данная таблица была заполнена 100.000 записей с различным тестовым значением, приближенным к реальности, что позволили сымитировать реальную нагрузку на систему. Из этих 100.000 записей примерно 5% записей были с пользователями, которые были заблокированы. Код анонимного блока на добавление 100.000 записей представлен в листинге 4.3.

```
    BEGIN
        FOR i IN 1..100000 LOOP
            INSERT INTO Users (name, email, password, role_id, cre-
ated_at, is_blocked)
            VALUES (
                'User' || DBMS_RANDOM.STRING('L', 5) || i,
                LOWER(DBMS_RANDOM.STRING('L', 5) || i || '@exam-
ple.com'),
                DBMS_RANDOM.STRING('X', 20),
                1,
                SYSTIMESTAMP - DBMS_RANDOM.VALUE(0, 365),
                CASE WHEN DBMS_RANDOM.VALUE(0, 1) > 0.95 THEN 0 ELSE
0 END
            );

        END LOOP;
        COMMIT;
    END;
    /
```

Листинг 4.3 – Анонимный блок для добавления 100.000 записей в таблицу

На рисунке 4.1 представлено затраченное время на поиск незаблокированных пользователей в таблице «Users».

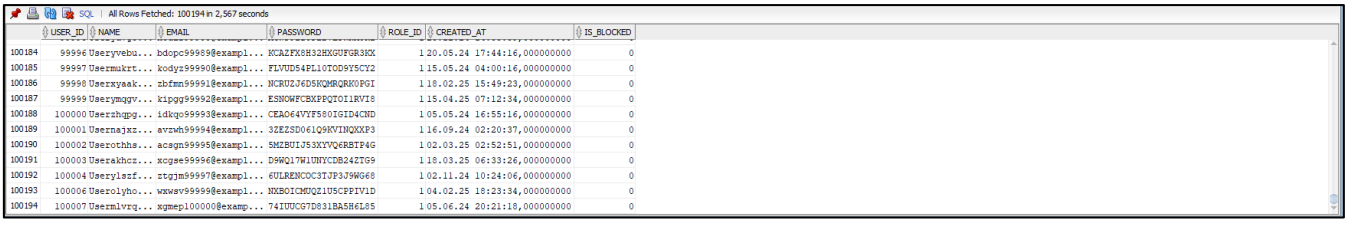
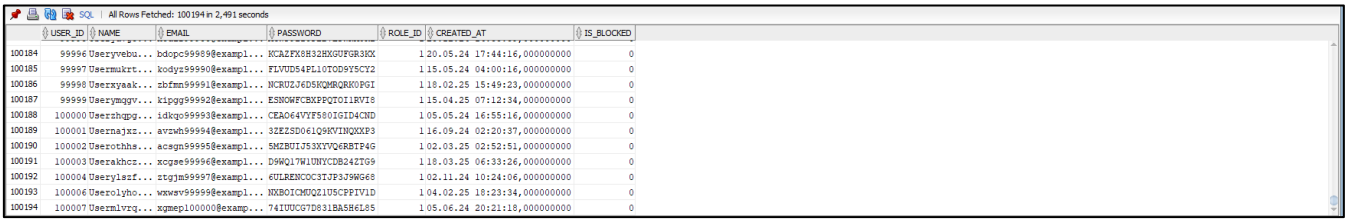



Рисунок 4.1 – Затраченное время на просмотр записей

На данном рисунке видно, что для просмотра 100194 записей было затрачено 2,567 секунд, что является очень большим числом, учитывая, что по мере расширения сервиса и увеличения количества пользователей, данное число будет только расти в геометрической прогрессии.

Для оптимизации выполнения запроса был создан индекс idx_users_is_blocked по столбцу is_blocked, позволяющий базе данных использовать более быстрый способ выборки данных.

На рисунке 4.2 представлено затраченное время на поиск незаблокированных пользователей в таблицу «Users» после создания индекса.


All Rows Fetched: 100194 in 2,491 seconds

| | USER_ID | NAME | EMAIL | PASSWORD | ROLE_ID | CREATED_AT | IS_BLOCKED |
|---|---|---|---|---|---|---|---|
| 100184 | 99996 | Useryvebu... | bdopc99989@exampl... | KCAZFX8H32HXGUFGR3KX | 1 | 20.05.24 17:44:16,000000000 | 0 |
| 100185 | 99997 | Usermukrt... | kodyz99990@exampl... | FLVUD54PL10TOD9Y5CY2 | 1 | 15.05.24 04:00:16,000000000 | 0 |
| 100186 | 99998 | Userxyaak... | zbfmn99991@exampl... | NCRUZJ6D5KQMRQRK0PGI | 1 | 18.02.25 15:49:23,000000000 | 0 |
| 100187 | 99999 | Userymqqv... | kipgg99992@exampl... | ESNOWFCBXPFQTO1IRVI8 | 1 | 15.04.25 07:12:34,000000000 | 0 |
| 100188 | 100000 | Userzhqpg... | idkgo99993@exampl... | CEAO64VYF580IGID4CND | 1 | 05.05.24 16:55:16,000000000 | 0 |
| 100189 | 100001 | Usernajxz... | avzwh99994@exampl... | 3ZEZSD061Q9KVINQXXP3 | 1 | 16.09.24 02:20:37,000000000 | 0 |
| 100190 | 100002 | Userothhs... | acsgn99995@exampl... | 5MZBUIJ53XYVQ6RBTP4G | 1 | 02.03.25 02:52:51,000000000 | 0 |
| 100191 | 100003 | Userakhcz... | xcgse99996@exampl... | D9WQ17W1UNYCDB24ZTG9 | 1 | 18.03.25 06:33:26,000000000 | 0 |
| 100192 | 100004 | Userylszf... | ztgjm99997@exampl... | 6ULRENCOC3TJP3J9WG68 | 1 | 02.11.24 10:24:06,000000000 | 0 |
| 100193 | 100006 | Userolyho... | wxwsv99999@exampl... | NXBOICMQQ2IU5CPPIV1D | 1 | 04.02.25 18:23:34,000000000 | 0 |
| 100194 | 100007 | Usermlvrq... | xgmep100000@examp... | 74IUUCG7D831BA5H6L85 | 1 | 05.06.24 20:21:18,000000000 | 0 |

Рисунок 4.2 – Затраченное время на просмотр записей после индексирования

На данном рисунке видно, что для просмотра 100.194 записей было затрачено 2,491 секунды, что меньше первого результата на 0,076 секунд. Учитывая, что поле «is_blocked» имеет только два возможных значения (0 и 1), что является примером низкой селективности, уменьшение времени на 0,076 секунд считается хорошим результатом.

### 4.5 Вывод по разделу

В данном разделе была описаны: выбранная база данных, процесс создания таблиц и ролей для разграничения доступа, процедуры для экспорта и импорта данных, а также было проведено тестирование производительности базы данных путем добавления в одну из таблиц 100.000 записей с последующей индексацией по какому-либо полю.

# 5 Тестирование, проверка работоспособности и анализ полученных результатов

## 5.1 Тестирование клиентской части

Для начала тестирования нам необходимо создать пользователя, за что отвечает процедура «register_user». Регистрация пользователя представлена на рисунке 5.1.

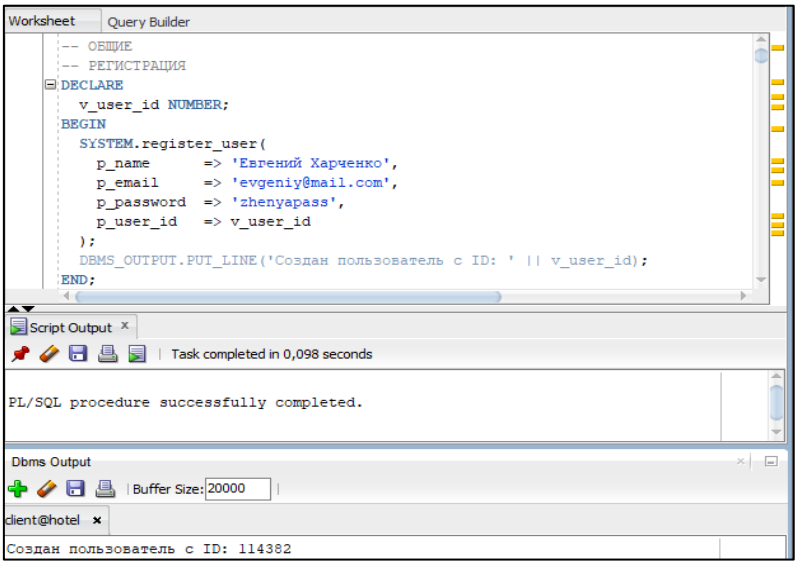



Рисунок 5.1 – Регистрация нового пользователя

После регистрации пользователь уже может ознакомиться с доступными номерами, за что отвечает функция «get_available_rooms». Результат выполнения данной функции представлен ни рисунке 5.2.

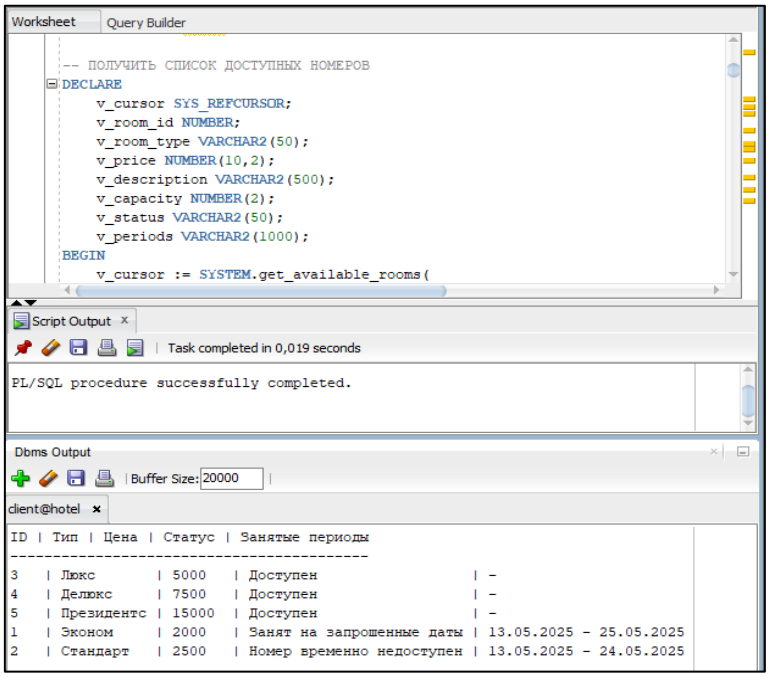



Рисунок 5.2 – Результат выполнения процедуры просмотра свободных номеров

После просмотра свободных номеров пользователь может забронировать любой понравившейся ему из них. За это отвечает процедура «book_room». Результат выполнения данной процедуры представлен на рисунке 5.3.

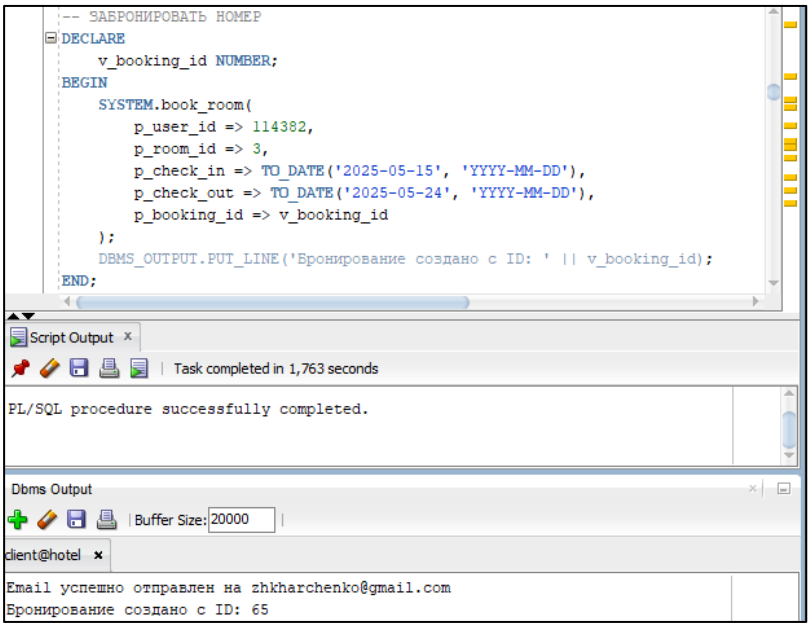



Рисунок 5.3 – Результат выполнения процедуры бронирования номера

Помимо прочего, пользователь может добавить услуги к его актуальным бронированиям, за это отвечают функция «get_all_services» (для просмотра дополнительных услуг к бронированиям) и процедура «add_service» (для добавления услуг к бронированиям). Результат выполнения функции и процедуры представлен на рисунке 5.4.

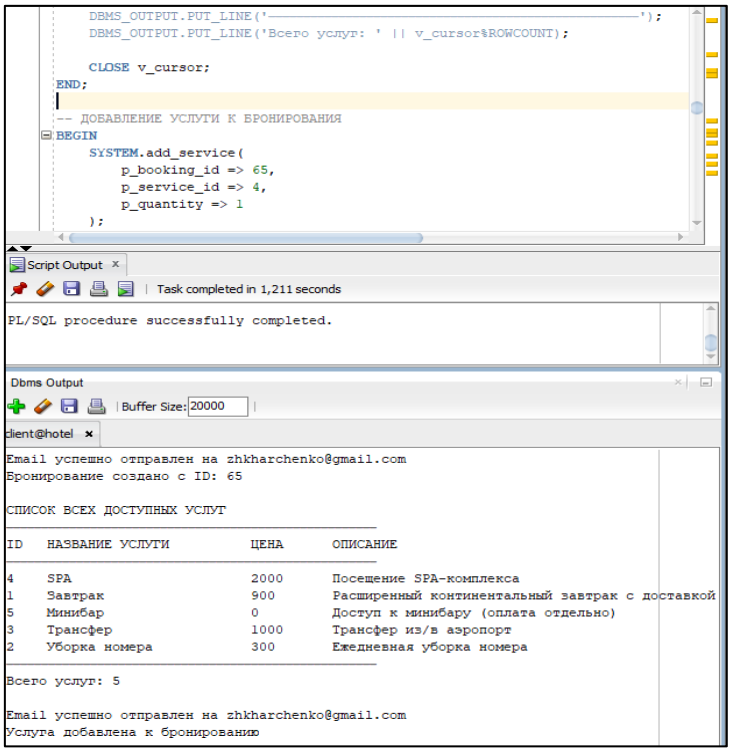



Рисунок 5.4 – Просмотр доп. услуг и добавление одной из услуг к бронированию

Также пользователь может ознакомиться со своими бронированиями. За это отвечает функция «get_user_bookings». Результат выполнения данной функции представлен на рисунке 5.5.

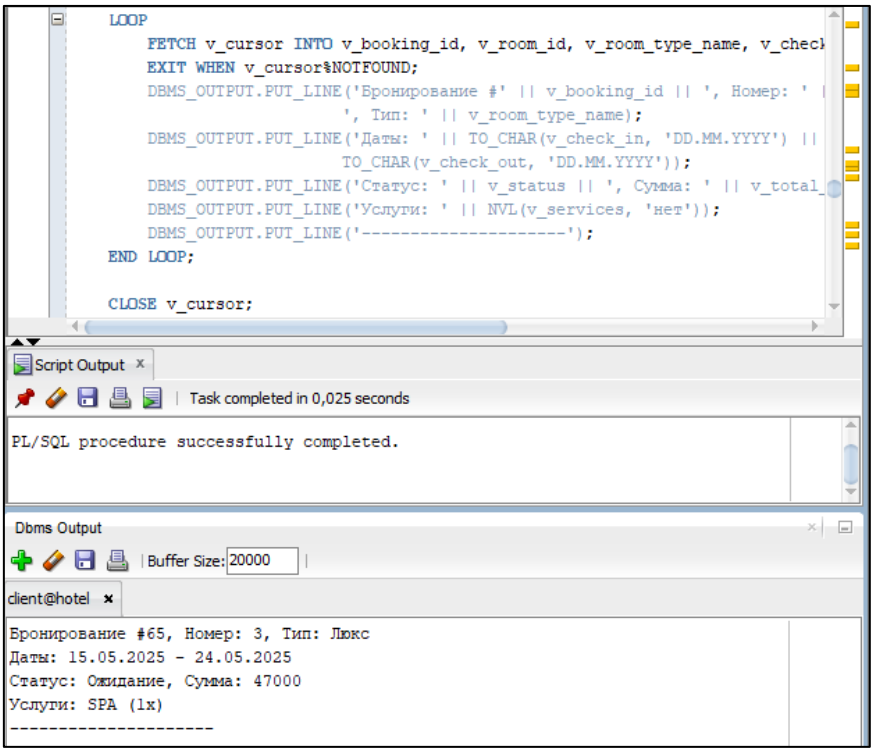



Рисунок 5.5 – Результат выполнения функции просмотра бронирований

Пользователь может отменить своё бронирование при помощи процедуры «cancel_booking». Результат выполнения данной процедуры представлен на рисунке 5.6.

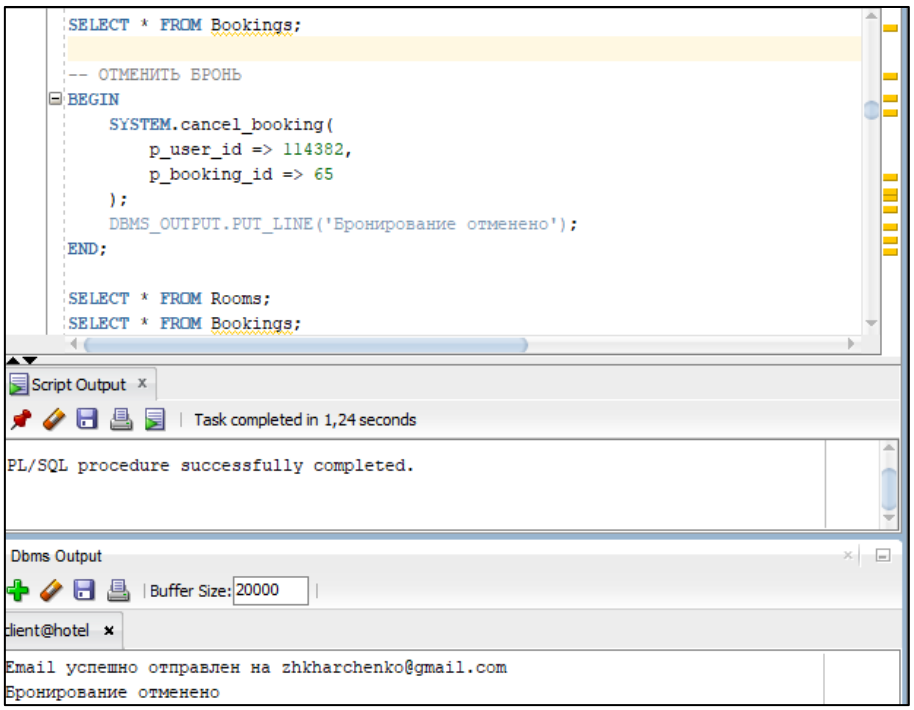



Рисунок 5.6 – Результат выполнения процедуры отмены бронирования

Также пользователь может ознакомиться с отзывами к номерам и оставить свой отзыв.

### 5.2 Тестирование области работы менеджера отеля

Основная возможность менеджера отеля – возможность изменения статуса бронирований. За это отвечает процедура «change_booking_status». Результат выполнения данной процедуры представлен на рисунке 5.7.

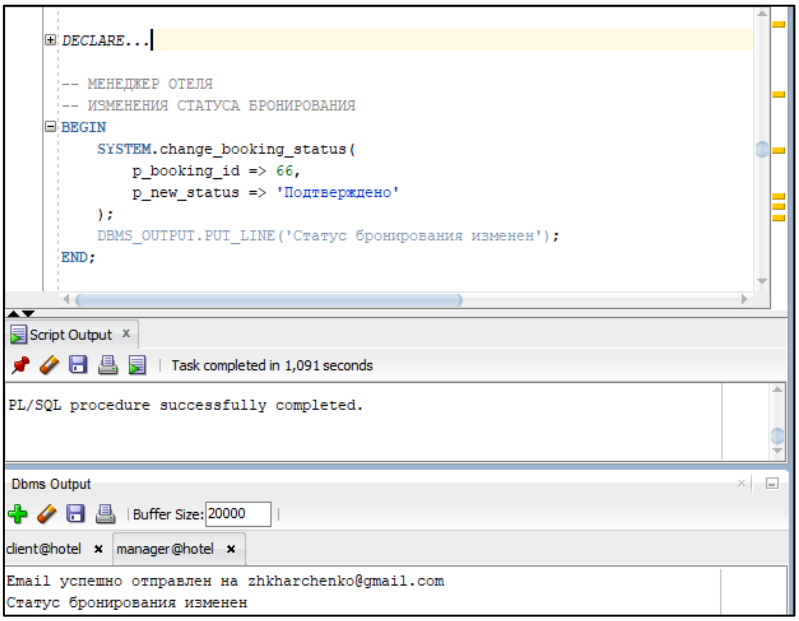



Рисунок 5.7 – Результат выполнения процедуры изменения статуса бронирования

Также менеджер может обновлять цены на номера. За это отвечает процедура «update_room_price». Результат выполнения данной процедуры представлен на рисунке 5.8.

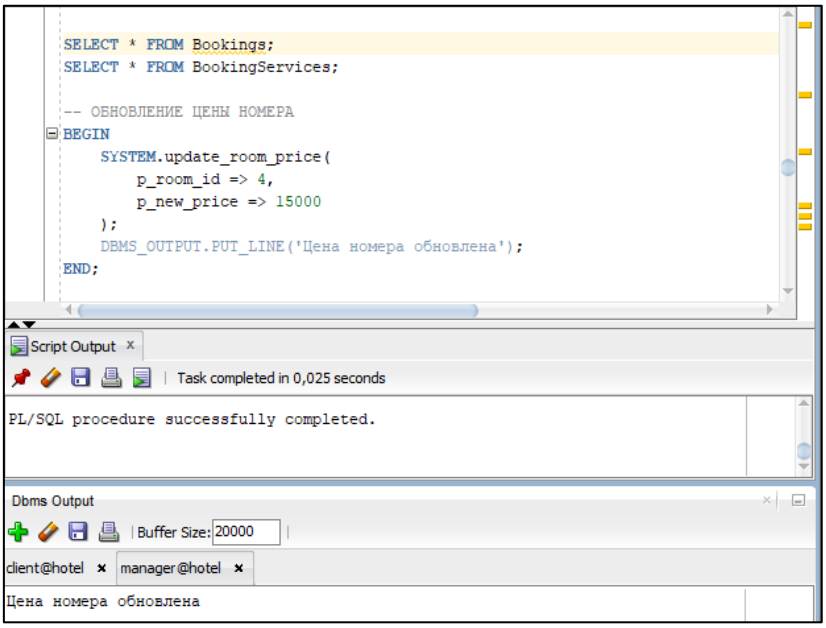



Рисунок 5.8 – Результат выполнения процедуры на изменение цены номера

Помимо прочего, менеджер ещё может удалить услуги из бронирований пользователя и в принципе проверять список бронирований пользователя.

### 5.3 Тестирование области работы специального администратора

Ключевой возможностью специального администратора является работа с пользователями: назначение менеджера отеля, блокировка/разблокировка пользователя и его удаления. К примеру, результат выполнения процедуры удаления пользователя представлен на рисунке 5.9.

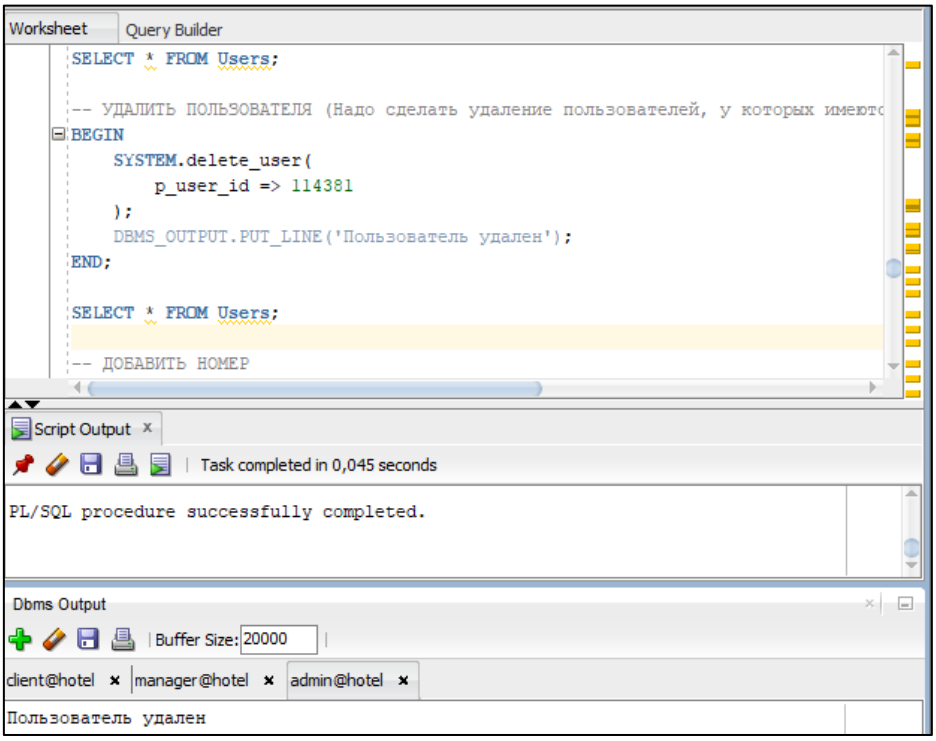



Рисунок 5.9 – Результат выполнения процедуры удаления пользователя

Помимо прочего, специальный администратор также имеет полные полномочия на изменения номерного фонда отеля, а также списка дополнительных услуг, а именно их добавление, редактирование и удаление.

### 5.4 Вывод по разделу

В данном разделе было проведено тестирование возможностей пользователя, менеджера отеля и специального администратора. По итогам тестирования можно утверждать, что данный функционал полностью удовлетворяет всем критериям реализации сервиса онлайн-бронирования номеров в отеле и в полной мере покрывает все потребности в возможностях для работы с данным сервисом.

## Заключение

Разработанная база данных для системы онлайн-бронирования отеля успешно решает все поставленные в проекте задачи. В ней реализована трехуровневая система ролей (пользователь, менеджер отеля, специальный администратор), обеспечивающая четкое разграничение прав доступа и безопасность данных.

Организован полный функционал управления бронированиями:

– Для пользователей: просмотр номеров, бронирование, управление своими заказами и отзывами;

– Для менеджеров: управление всеми бронированиями, ценообразованием и дополнительными услугами;

– Для администраторов: контроль номерного фонда, услуг и учетных записей.

Для интеграции с внешними системами разработаны процедуры импорта и экспорта данных в формате JSON, автоматизирующие обмен информацией.

Для повышения удобства пользователей внедрена система email-уведомлений о статусах бронирований с использованием защищенного SMTP-соединения через Gmail.

База данных демонстрирует стабильную работу при высоких нагрузках, отвечая требованиям масштабируемости, надежности и безопасности.

Проект полностью готов к внедрению и может служить основой для дальнейшего расширения функциональности, включая разработку мобильного приложения.

## Список использованных источников

1. Booking.com [Электронный ресурс]. / Режим доступа: URL https://www.booking.com. – Дата доступа: 01.04.2025.
2. Expedia [Электронный ресурс]. / Режим доступа: URL https://www.expedia.com. – Дата доступа: 01.04.2025.
3. 101hotels.com [Электронный ресурс]. / Режим доступа: URL https://101hotels.com. – Дата доступа: 01.04.2025.
4. Airbnb [Электронный ресурс]. / Режим доступа: URL https://www.airbnb.com. – Дата доступа: 01.04.2025.

# ПРИЛОЖЕНИЕ А.
## Запросы для создания таблиц БД

```
CREATE TABLE Roles (
    role_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    role_name VARCHAR2(50) NOT NULL
) TABLESPACE hotel_db_data;

INSERT INTO ROLES (role_name) VALUES ('Пользователь');
INSERT INTO ROLES (role_name) VALUES ('Менеджер отеля');
INSERT INTO ROLES (role_name) VALUES ('Специальный администратор');

CREATE TABLE Users (
    user_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    name VARCHAR2(100) NOT NULL,
    email VARCHAR2(100) UNIQUE NOT NULL,
    password VARCHAR2(255) NOT NULL,
    role_id NUMBER NOT NULL,
    created_at TIMESTAMP DEFAULT SYSTIMESTAMP,
    is_blocked NUMBER(1) DEFAULT 0 CHECK (is_blocked IN (0, 1)),
    CONSTRAINT fk_user_role FOREIGN KEY (role_id) REFERENCES
Roles(role_id)
) TABLESPACE hotel_db_data;

CREATE TABLE RoomTypes (
    room_type_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    room_type_name VARCHAR2(50) NOT NULL
) TABLESPACE hotel_db_data;

INSERT INTO RoomTypes(room_type_name) VALUES ('Эконом');
INSERT INTO RoomTypes(room_type_name) VALUES ('Стандарт');
INSERT INTO RoomTypes(room_type_name) VALUES ('Люкс');
INSERT INTO RoomTypes(room_type_name) VALUES ('Делюкс');
INSERT INTO RoomTypes(room_type_name) VALUES ('Президентский');

CREATE TABLE Rooms (
    room_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    room_type_id NUMBER NOT NULL,
    price_per_night NUMBER(10,2) NOT NULL,
    description VARCHAR2(500),
    capacity NUMBER(2) NOT NULL,
    is_available NUMBER(1) DEFAULT 1 CHECK (is_available IN (0, 1)),
    CONSTRAINT fk_room_type FOREIGN KEY (room_type_id) REFERENCES
RoomTypes(room_type_id)
) TABLESPACE hotel_db_data;

CREATE TABLE Bookings (
    booking_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    user_id NUMBER NOT NULL,
    room_id NUMBER NOT NULL,
    check_in_date DATE NOT NULL,
    check_out_date DATE NOT NULL,
```

```
    status VARCHAR2(25) DEFAULT 'Ожидание' CHECK (status IN
('Ожидание', 'Подтверждено', 'Отменено', 'Выполнено')),
    total_price NUMBER(10,2) NOT NULL,
    created_at TIMESTAMP DEFAULT SYSTIMESTAMP,
    CONSTRAINT fk_booking_user FOREIGN KEY (user_id) REFERENCES Us-
ers(user_id),
    CONSTRAINT fk_booking_room FOREIGN KEY (room_id) REFERENCES
Rooms(room_id),
    CONSTRAINT chk_dates CHECK (check_out_date > check_in_date)
) TABLESPACE hotel_db_data;

CREATE TABLE RoomServices (
    service_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    service_name VARCHAR2(100) NOT NULL,
    price NUMBER(10,2) NOT NULL,
    description VARCHAR2(500)
) TABLESPACE hotel_db_data;

CREATE TABLE BookingServices (
    booking_service_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    booking_id NUMBER NOT NULL,
    service_id NUMBER NOT NULL,
    quantity NUMBER DEFAULT 1,
    CONSTRAINT fk_bs_booking FOREIGN KEY (booking_id) REFERENCES Book-
ings(booking_id),
    CONSTRAINT fk_bs_service FOREIGN KEY (service_id) REFERENCES Room-
Services(service_id)
) TABLESPACE hotel_db_data;

CREATE TABLE Reviews (
    review_id NUMBER GENERATED ALWAYS AS IDENTITY PRIMARY KEY,
    user_id NUMBER NOT NULL,
    room_id NUMBER NOT NULL,
    rating NUMBER(1) NOT NULL CHECK (rating BETWEEN 1 AND 5),
    comment_text VARCHAR2(1000),
    created_at TIMESTAMP DEFAULT SYSTIMESTAMP,
    CONSTRAINT fk_review_user FOREIGN KEY (user_id) REFERENCES Us-
ers(user_id),
    CONSTRAINT fk_review_room FOREIGN KEY (room_id) REFERENCES
Rooms(room_id)
) TABLESPACE hotel_db_data;
```

# ПРИЛОЖЕНИЕ Б.
## Запросы для создания процедур

```
-- ОБЩИЕ ПРОЦЕДУРЫ
CREATE OR REPLACE PROCEDURE register_user(
    p_name IN VARCHAR2,
    p_email IN VARCHAR2,
    p_password IN VARCHAR2,
    p_role_name IN VARCHAR2 DEFAULT 'Пользователь',
    p_user_id OUT NUMBER
) AS
    v_role_id NUMBER;
BEGIN
    SELECT role_id INTO v_role_id
    FROM Roles
    WHERE LOWER(role_name) = LOWER(p_role_name);

    INSERT INTO Users (name, email, password, role_id)
    VALUES (p_name, p_email, p_password, v_role_id)
    RETURNING user_id INTO p_user_id;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN ROLLBACK; RAISE;
END;
/

-- ДЛЯ ОБЫЧНОГО ПОЛЬЗОВАТЕЛЯ
CREATE OR REPLACE PROCEDURE book_room(
    p_user_id IN NUMBER,
    p_room_id IN NUMBER,
    p_check_in IN DATE,
    p_check_out IN DATE,
    p_booking_id OUT NUMBER
) AS
    v_price_per_night NUMBER(10,2);
    v_is_available NUMBER;
    v_conflict_count NUMBER;
    v_current_date DATE := TRUNC(SYSDATE);
    v_is_blocked NUMBER;
BEGIN
    SELECT is_blocked INTO v_is_blocked
    FROM Users
    WHERE user_id = p_user_id;

    IF v_is_blocked = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Ошибка: Пользователь заблокирован и не
может совершать бронирования');
        RAISE_APPLICATION_ERROR(-20007, 'Пользователь заблокирован');
    END IF;
```

```
    BEGIN
        SELECT price_per_night, is_available
        INTO v_price_per_night, v_is_available
        FROM Rooms
        WHERE room_id = p_room_id;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20001, 'Номер с ID ' ||
p_room_id || ' не найден');
    END;

    IF p_check_in < v_current_date THEN
        RAISE_APPLICATION_ERROR(-20005, 'Дата заезда не может быть в
прошлом');
    END IF;

    IF p_check_out < v_current_date THEN
        RAISE_APPLICATION_ERROR(-20006, 'Дата выезда не может быть в
прошлом');
    END IF;

    IF p_check_in >= p_check_out THEN
        RAISE_APPLICATION_ERROR(-20004, 'Дата выезда должна быть
позже даты заезда');
    END IF;

    IF v_is_available = 0 THEN
        RAISE_APPLICATION_ERROR(-20002, 'Номер в данный момент за-
нят');
    END IF;

    SELECT COUNT(*) INTO v_conflict_count
    FROM Bookings
    WHERE room_id = p_room_id
    AND status NOT IN ('Отменено', 'Выполнено')
    AND NOT (p_check_out <= check_in_date OR p_check_in >=
check_out_date);

    IF v_conflict_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20003, 'Номер уже забронирован на
указанные даты');
    END IF;

    INSERT INTO Bookings (user_id, room_id, check_in_date,
check_out_date, total_price)
    VALUES (p_user_id, p_room_id, p_check_in, p_check_out,
           v_price_per_night * (p_check_out - p_check_in))
    RETURNING booking_id INTO p_booking_id;
```

```
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/

CREATE OR REPLACE PROCEDURE cancel_booking(
    p_user_id IN NUMBER,
    p_booking_id IN NUMBER
) AS
    v_room_id NUMBER;
BEGIN
    SELECT room_id INTO v_room_id FROM Bookings
    WHERE booking_id = p_booking_id AND user_id = p_user_id;

    UPDATE Bookings SET status = 'Отменено'
    WHERE booking_id = p_booking_id;

    UPDATE Rooms SET is_available = 1 WHERE room_id = v_room_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN ROLLBACK; RAISE;
END;
/

CREATE OR REPLACE PROCEDURE add_review(
    p_user_id IN NUMBER,
    p_room_id IN NUMBER,
    p_rating IN NUMBER,
    p_comment IN VARCHAR2 DEFAULT NULL
) AS
    v_has_booking NUMBER;
    v_is_blocked NUMBER;
BEGIN
    SELECT is_blocked INTO v_is_blocked
    FROM Users
    WHERE user_id = p_user_id;

    IF v_is_blocked = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Ошибка: Пользователь заблокирован и не
может оставлять отзывы');
        RAISE_APPLICATION_ERROR(-20008, 'Пользователь заблокирован');
    END IF;

    SELECT COUNT(*) INTO v_has_booking
    FROM Bookings
    WHERE user_id = p_user_id AND room_id = p_room_id AND status =
'Выполнено';
```

```
    IF v_has_booking = 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Нельзя оставить отзыв без
завершенного бронирования');
    END IF;

    INSERT INTO Reviews (user_id, room_id, rating, comment_text)
    VALUES (p_user_id, p_room_id, p_rating, p_comment);
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN ROLLBACK; RAISE;
END;
/

CREATE OR REPLACE PROCEDURE add_service(
    p_booking_id IN NUMBER,
    p_service_id IN NUMBER,
    p_quantity IN NUMBER DEFAULT 1
) AS
    v_room_price NUMBER(10,2);
    v_nights NUMBER;
    v_services_total NUMBER(10,2);
BEGIN
    INSERT INTO BookingServices (booking_id, service_id, quantity)
    VALUES (p_booking_id, p_service_id, p_quantity);

    SELECT r.price_per_night, (b.check_out_date - b.check_in_date)
    INTO v_room_price, v_nights
    FROM Bookings b
    JOIN Rooms r ON b.room_id = r.room_id
    WHERE b.booking_id = p_booking_id;

    SELECT NVL(SUM(rs.price * bs.quantity), 0)
    INTO v_services_total
    FROM BookingServices bs
    JOIN RoomServices rs ON bs.service_id = rs.service_id
    WHERE bs.booking_id = p_booking_id;

    UPDATE Bookings
    SET total_price = (v_room_price * v_nights) + v_services_total
    WHERE booking_id = p_booking_id;

    COMMIT;
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        ROLLBACK;
        RAISE_APPLICATION_ERROR(-20001, 'Бронирование или услуга не
найдены');
    WHEN OTHERS THEN
        ROLLBACK;
```

```
        RAISE;
END;
/

-- ДЛЯ МЕНЕДЖЕРА ОТЕЛЯ:
CREATE OR REPLACE PROCEDURE change_booking_status(
    p_booking_id IN NUMBER,
    p_new_status IN VARCHAR2
) AS
    v_room_id Rooms.room_id%TYPE;
    v_active_bookings NUMBER;
BEGIN
    SELECT room_id INTO v_room_id FROM Bookings WHERE booking_id =
p_booking_id;

    UPDATE Bookings
    SET status = p_new_status
    WHERE booking_id = p_booking_id;

    IF p_new_status IN ('Отменено', 'Выполнено') THEN
        SELECT COUNT(*) INTO v_active_bookings
        FROM Bookings
        WHERE room_id = v_room_id AND status = 'Подтверждено' AND
booking_id != p_booking_id;

        IF v_active_bookings = 0 THEN
            UPDATE Rooms SET is_available = 1 WHERE room_id =
v_room_id;
        END IF;

    ELSIF p_new_status = 'Подтверждено' THEN
        UPDATE Rooms SET is_available = 0 WHERE room_id = v_room_id;
    END IF;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;

CREATE OR REPLACE PROCEDURE update_room_price(
    p_room_id IN NUMBER,
    p_new_price IN NUMBER
) AS
BEGIN
    UPDATE Rooms SET price_per_night = p_new_price
    WHERE room_id = p_room_id;
    COMMIT;
EXCEPTION
```

```
    WHEN OTHERS THEN ROLLBACK; RAISE;
END;

CREATE OR REPLACE PROCEDURE remove_service_from_booking(
    p_booking_id IN NUMBER,
    p_service_id IN NUMBER
) AS
BEGIN
    DELETE FROM BookingServices
    WHERE booking_id = p_booking_id AND service_id = p_service_id;

    UPDATE Bookings b
    SET total_price = (
        SELECT r.price_per_night * (b.check_out_date -
b.check_in_date) +
               NVL(SUM(rs.price * bs.quantity), 0)
        FROM Rooms r
        LEFT JOIN BookingServices bs ON b.booking_id = bs.booking_id
        LEFT JOIN RoomServices rs ON bs.service_id = rs.service_id
        WHERE r.room_id = b.room_id
        GROUP BY r.price_per_night, b.check_out_date, b.check_in_date
    )
    WHERE booking_id = p_booking_id;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/

-- ДЛЯ СПЕЦИАЛЬНЫХ АДМИНИСТРАТОРОВ:
CREATE OR REPLACE PROCEDURE toggle_user_block(
    p_user_id IN NUMBER,
    p_block IN NUMBER
) AS
BEGIN
    UPDATE Users SET is_blocked = p_block
    WHERE user_id = p_user_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN ROLLBACK; RAISE;
END;
/

CREATE OR REPLACE PROCEDURE assign_manager_role(
    p_user_id IN NUMBER
) AS
BEGIN
```

```
    UPDATE Users
    SET role_id = (SELECT role_id FROM Roles WHERE role_name =
'Менеджер отеля')
    WHERE user_id = p_user_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN ROLLBACK; RAISE;
END;
/

CREATE OR REPLACE PROCEDURE delete_user(
    p_user_id IN NUMBER
) AS
BEGIN
    DELETE FROM BookingServices
    WHERE booking_id IN (SELECT booking_id FROM Bookings WHERE
user_id = p_user_id);

    DELETE FROM Reviews WHERE user_id = p_user_id;

    DELETE FROM Bookings WHERE user_id = p_user_id;

    DELETE FROM Users WHERE user_id = p_user_id;

    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/

CREATE OR REPLACE PROCEDURE add_room(
    p_room_type_id IN NUMBER,
    p_price_per_night IN NUMBER,
    p_description IN VARCHAR2,
    p_capacity IN NUMBER,
    p_room_id OUT NUMBER
) AS
BEGIN
    INSERT INTO Rooms (room_type_id, price_per_night, description,
capacity)
    VALUES (p_room_type_id, p_price_per_night, p_description, p_ca-
pacity)
    RETURNING room_id INTO p_room_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
```

```
END;
/

CREATE OR REPLACE PROCEDURE update_room(
    p_room_id IN NUMBER,
    p_room_type_id IN NUMBER DEFAULT NULL,
    p_price_per_night IN NUMBER DEFAULT NULL,
    p_description IN VARCHAR2 DEFAULT NULL,
    p_capacity IN NUMBER DEFAULT NULL,
    p_is_available IN NUMBER DEFAULT NULL
) AS
BEGIN
    UPDATE Rooms
    SET room_type_id = NVL(p_room_type_id, room_type_id),
        price_per_night = NVL(p_price_per_night, price_per_night),
        description = NVL(p_description, description),
        capacity = NVL(p_capacity, capacity),
        is_available = NVL(p_is_available, is_available)
    WHERE room_id = p_room_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/

CREATE OR REPLACE PROCEDURE delete_room(
    p_room_id IN NUMBER
) AS
    v_booking_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_booking_count
    FROM Bookings
    WHERE room_id = p_room_id AND status NOT IN ('Отменено',
'Выполнено');

    IF v_booking_count > 0 THEN
        RAISE_APPLICATION_ERROR(-20001, 'Нельзя удалить номер с ак-
тивными бронированиями');
    END IF;

    DELETE FROM Rooms WHERE room_id = p_room_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/
```

```
CREATE OR REPLACE PROCEDURE add_new_service(
    p_service_name IN VARCHAR2,
    p_price IN NUMBER,
    p_description IN VARCHAR2 DEFAULT NULL,
    p_service_id OUT NUMBER
) AS
BEGIN
    INSERT INTO RoomServices (service_name, price, description)
    VALUES (p_service_name, p_price, p_description)
    RETURNING service_id INTO p_service_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/

CREATE OR REPLACE PROCEDURE update_service(
    p_service_id IN NUMBER,
    p_service_name IN VARCHAR2 DEFAULT NULL,
    p_price IN NUMBER DEFAULT NULL,
    p_description IN VARCHAR2 DEFAULT NULL
) AS
BEGIN
    UPDATE RoomServices
    SET service_name = NVL(p_service_name, service_name),
        price = NVL(p_price, price),
        description = NVL(p_description, description)
    WHERE service_id = p_service_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/

CREATE OR REPLACE PROCEDURE delete_service(
    p_service_id IN NUMBER
) AS
    v_booking_count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_booking_count
    FROM BookingServices
    WHERE service_id = p_service_id;

    IF v_booking_count > 0 THEN
```

```
        RAISE_APPLICATION_ERROR(-20002, 'Нельзя удалить услугу, свя-
занную с бронированиями');
    END IF;

    DELETE FROM RoomServices WHERE service_id = p_service_id;
    COMMIT;
EXCEPTION
    WHEN OTHERS THEN
        ROLLBACK;
        RAISE;
END;
/
```

# ПРИЛОЖЕНИЕ В.
## Запросы для создания функций

```
CREATE OR REPLACE FUNCTION get_available_rooms(
    p_check_in DATE,
    p_check_out DATE,
    p_user_id NUMBER DEFAULT NULL
) RETURN SYS_REFCURSOR AS
    v_cursor SYS_REFCURSOR;
    v_is_blocked NUMBER := 0;
BEGIN
    IF p_user_id IS NOT NULL THEN
        BEGIN
            SELECT is_blocked INTO v_is_blocked
            FROM Users
            WHERE user_id = p_user_id;

            IF v_is_blocked = 1 THEN
                DBMS_OUTPUT.PUT_LINE('Предупреждение: Пользователь
заблокирован, но может просматривать доступные номера');
            END IF;
        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                NULL;
        END;
    END IF;

    OPEN v_cursor FOR
    SELECT
        r.room_id,
        rt.room_type_name,
        r.price_per_night,
        r.description,
        r.capacity,
        CASE
            WHEN r.is_available = 0 THEN 'Номер временно недоступен'
            WHEN EXISTS (
                SELECT 1 FROM Bookings b
                WHERE b.room_id = r.room_id
                AND b.status NOT IN ('Отменено', 'Выполнено')
                AND NOT (p_check_out <= b.check_in_date OR p_check_in
>= b.check_out_date)
            ) THEN 'Занят на запрошенные даты'
            ELSE 'Доступен'
        END AS availability_status,
        (
            SELECT LISTAGG(
                TO_CHAR(b.check_in_date, 'DD.MM.YYYY') || ' - ' ||
                TO_CHAR(b.check_out_date, 'DD.MM.YYYY'),
                ', '
```

```
            ) WITHIN GROUP (ORDER BY b.check_in_date)
            FROM Bookings b
            WHERE b.room_id = r.room_id
            AND b.status NOT IN ('Отменено', 'Выполнено')
            AND b.check_out_date > SYSDATE
        ) AS occupied_periods
    FROM Rooms r
    JOIN RoomTypes rt ON r.room_type_id = rt.room_type_id
    GROUP BY
        r.room_id,
        rt.room_type_name,
        r.price_per_night,
        r.description,
        r.capacity,
        r.is_available,
        CASE
            WHEN r.is_available = 0 THEN 'Номер временно недоступен'
            WHEN EXISTS (
                SELECT 1 FROM Bookings b
                WHERE b.room_id = r.room_id
                AND b.status NOT IN ('Отменено', 'Выполнено')
                AND NOT (p_check_out <= b.check_in_date OR p_check_in
>= b.check_out_date)
            ) THEN 'Занят на запрошенные даты'
            ELSE 'Доступен'
        END
    ORDER BY
        CASE
            WHEN EXISTS (
                SELECT 1 FROM Bookings b
                WHERE b.room_id = r.room_id
                AND b.status NOT IN ('Отменено', 'Выполнено')
                AND NOT (p_check_out <= b.check_in_date OR p_check_in
>= b.check_out_date)
            ) THEN 1
            WHEN r.is_available = 0 THEN 2
            ELSE 0
        END,
        r.price_per_night;

    RETURN v_cursor;
END;
/

CREATE OR REPLACE FUNCTION get_room_reviews(
    p_room_id IN NUMBER,
    p_user_id IN NUMBER DEFAULT NULL
) RETURN SYS_REFCURSOR AS
    v_cursor SYS_REFCURSOR;
    v_is_blocked NUMBER := 0;
```

```
BEGIN
    IF p_user_id IS NOT NULL THEN
        BEGIN
            SELECT is_blocked INTO v_is_blocked
            FROM Users
            WHERE user_id = p_user_id;

            IF v_is_blocked = 1 THEN
                DBMS_OUTPUT.PUT_LINE('Предупреждение: Пользователь
заблокирован, но может просматривать отзывы');
            END IF;
        EXCEPTION
            WHEN NO_DATA_FOUND THEN
                NULL;
        END;
    END IF;

    OPEN v_cursor FOR
    SELECT u.name, r.rating, r.comment_text, r.created_at
    FROM Reviews r
    JOIN Users u ON r.user_id = u.user_id
    WHERE r.room_id = p_room_id
    ORDER BY r.created_at DESC;
    RETURN v_cursor;
END;
/

CREATE OR REPLACE FUNCTION get_user_bookings(
    p_user_id IN NUMBER
) RETURN SYS_REFCURSOR AS
    v_cursor SYS_REFCURSOR;
    v_is_blocked NUMBER;
BEGIN
    SELECT is_blocked INTO v_is_blocked
    FROM Users
    WHERE user_id = p_user_id;

    IF v_is_blocked = 1 THEN
        DBMS_OUTPUT.PUT_LINE('Предупреждение: Пользователь заблокиро-
ван, но может просматривать свои бронирования');
    END IF;

    OPEN v_cursor FOR
    SELECT
        b.booking_id,
        r.room_id,
        rt.room_type_name,
        b.check_in_date,
        b.check_out_date,
        b.status,
```

```
        b.total_price,
        (SELECT LISTAGG(rs.service_name || ' (' || bs.quantity ||
'x)', ', ')
         WITHIN GROUP (ORDER BY rs.service_name)
         FROM BookingServices bs
         JOIN RoomServices rs ON bs.service_id = rs.service_id
         WHERE bs.booking_id = b.booking_id) AS services_list
    FROM Bookings b
    JOIN Rooms r ON b.room_id = r.room_id
    JOIN RoomTypes rt ON r.room_type_id = rt.room_type_id
    WHERE b.user_id = p_user_id
    ORDER BY b.check_in_date DESC;
    RETURN v_cursor;
END;
/

CREATE OR REPLACE FUNCTION get_all_services
RETURN SYS_REFCURSOR AS
    v_cursor SYS_REFCURSOR;
BEGIN
    OPEN v_cursor FOR
    SELECT
        service_id,
        service_name,
        price,
        description,
        'Доступна' AS availability_status
    FROM
        RoomServices
    ORDER BY
        service_name;

    RETURN v_cursor;
END;
/
```

# ПРИЛОЖЕНИЕ Г.
## Запросы для создания триггеров

```
CREATE OR REPLACE TRIGGER trg_check_booking_dates
BEFORE INSERT OR UPDATE ON Bookings
FOR EACH ROW
DECLARE
    v_current_date DATE := TRUNC(SYSDATE);
BEGIN
    IF :NEW.check_in_date < v_current_date THEN
        RAISE_APPLICATION_ERROR(-20010, 'Дата заезда не может быть в
прошлом');
    END IF;

    IF :NEW.check_out_date < v_current_date THEN
        RAISE_APPLICATION_ERROR(-20011, 'Дата выезда не может быть в
прошлом');
    END IF;

    IF :NEW.check_in_date >= :NEW.check_out_date THEN
        RAISE_APPLICATION_ERROR(-20012, 'Дата выезда должна быть позже
даты заезда');
    END IF;
END;
/

CREATE OR REPLACE TRIGGER trg_check_room_availability
BEFORE INSERT OR UPDATE OF room_id, check_in_date, check_out_date ON
Bookings
FOR EACH ROW
DECLARE
    v_is_available NUMBER;
    v_conflict_count NUMBER;
BEGIN
    BEGIN
        SELECT is_available INTO v_is_available
        FROM Rooms WHERE room_id = :NEW.room_id;
    EXCEPTION
        WHEN NO_DATA_FOUND THEN
            RAISE_APPLICATION_ERROR(-20013,   'Номер   с   ID   '   ||
:NEW.room_id || ' не найден');
    END;

    SELECT COUNT(*) INTO v_conflict_count
    FROM Bookings
    WHERE room_id = :NEW.room_id
    AND booking_id != NVL(:NEW.booking_id, -1)
    AND status NOT IN ('Отменено', 'Выполнено')
    AND NOT (:NEW.check_out_date <= check_in_date OR :NEW.check_in_date
>= check_out_date);

    IF v_conflict_count > 0 THEN
```

```
        RAISE_APPLICATION_ERROR(-20014, 'Номер уже забронирован на
указанные даты');
    END IF;
END;
/

CREATE OR REPLACE TRIGGER trg_check_review
BEFORE INSERT ON Reviews
FOR EACH ROW
DECLARE
    v_has_completed_booking NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_has_completed_booking
    FROM Bookings
    WHERE user_id = :NEW.user_id
    AND room_id = :NEW.room_id
    AND status = 'Выполнено';

    IF v_has_completed_booking = 0 THEN
        RAISE_APPLICATION_ERROR(-20015, 'Вы не можете оставить отзыв о
номере, в котором не останавливались');
    END IF;

    IF :NEW.rating NOT BETWEEN 1 AND 5 THEN
        RAISE_APPLICATION_ERROR(-20016, 'Рейтинг должен быть от 1 до
5');
    END IF;
END;
/

CREATE OR REPLACE TRIGGER trg_prevent_room_deletion
BEFORE DELETE ON Rooms
FOR EACH ROW
DECLARE
    v_active_bookings NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_active_bookings
    FROM Bookings
    WHERE room_id = :OLD.room_id
    AND status NOT IN ('Отменено', 'Выполнено');

    IF v_active_bookings > 0 THEN
        RAISE_APPLICATION_ERROR(-20017, 'Нельзя удалить номер с актив-
ными бронированиями');
    END IF;
END;
/

CREATE OR REPLACE TRIGGER trg_prevent_service_deletion
BEFORE DELETE ON RoomServices
FOR EACH ROW
DECLARE
    v_used_services NUMBER;
BEGIN
```

```
    SELECT COUNT(*) INTO v_used_services
    FROM BookingServices
    WHERE service_id = :OLD.service_id;

    IF v_used_services > 0 THEN
        RAISE_APPLICATION_ERROR(-20018, 'Нельзя удалить услугу, свя-
занную с бронированиями');
    END IF;
END;
/

CREATE OR REPLACE TRIGGER check_booking_status_change
BEFORE UPDATE OF status ON Bookings
FOR EACH ROW
BEGIN
        IF (:OLD.status = 'Подтверждено' AND :NEW.status = 'Ожидание')
THEN
            RAISE_APPLICATION_ERROR(-20019, 'Нельзя вернуть подтвер-
жденное бронирование в статус "Ожидание"');

        ELSIF (:OLD.status = 'Отменено' AND :NEW.status IN ('Ожидание',
'Подтверждено')) THEN
            RAISE_APPLICATION_ERROR(-20020, 'Отмененное бронирование
нельзя восстановить');

        ELSIF (:OLD.status = 'Выполнено' AND :NEW.status IN ('Ожида-
ние', 'Подтверждено', 'Отменено')) THEN
            RAISE_APPLICATION_ERROR(-20021, 'Выполненное бронирование
нельзя изменить');

        ELSIF (:NEW.status = 'Подтверждено' AND :OLD.status IN ('Отме-
нено', 'Выполнено')) THEN
            RAISE_APPLICATION_ERROR(-20022, 'Нельзя подтвердить отме-
ненное или выполненное бронирование');

        ELSIF (:NEW.status = 'Отменено' AND :OLD.status = 'Выполнено')
THEN
            RAISE_APPLICATION_ERROR(-20023, 'Нельзя отменить выполнен-
ное бронирование');
        END IF;
END;
/
```

# ПРИЛОЖЕНИЕ Д.
## Запросы для создания процедуры импорта данных

```
CREATE OR REPLACE PROCEDURE import_from_file IS
  l_file       UTL_FILE.FILE_TYPE;
  l_json       CLOB := EMPTY_CLOB();
  l_buffer     VARCHAR2(32767);
  l_dest_offset NUMBER := 1;
  l_src_offset  NUMBER := 1;
  l_ctx         NUMBER := DBMS_LOB.DEFAULT_CSID;
BEGIN
  l_file := UTL_FILE.FOPEN('JSON_DIR', 'bookings.json', 'r', 32767);

  DBMS_LOB.CREATETEMPORARY(l_json, TRUE);

  LOOP
    BEGIN
      UTL_FILE.GET_LINE(l_file, l_buffer);
      DBMS_LOB.WRITEAPPEND(l_json, LENGTH(l_buffer) + 1, l_buffer ||
CHR(10));
    EXCEPTION
      WHEN NO_DATA_FOUND THEN
        EXIT;
    END;
  END LOOP;

  UTL_FILE.FCLOSE(l_file);

INSERT INTO Bookings (
  user_id, room_id, check_in_date, check_out_date,
  status, total_price, created_at
)
SELECT
  jt.user_id,
  jt.room_id,
  TO_DATE(jt.check_in_date, 'YYYY-MM-DD'),
  TO_DATE(jt.check_out_date, 'YYYY-MM-DD'),
  jt.status,
  jt.total_price,
  TO_TIMESTAMP(jt.created_at, 'YYYY-MM-DD"T"HH24:MI:SS')
FROM JSON_TABLE(
  l_json,
  '$[*]'
  COLUMNS (
    user_id        NUMBER PATH '$.userId',
    room_id        NUMBER PATH '$.roomId',
    check_in_date  VARCHAR2(20) PATH '$.checkInDate',
    check_out_date VARCHAR2(20) PATH '$.checkOutDate',
    status         VARCHAR2(20) PATH '$.status',
    total_price    NUMBER PATH '$.totalPrice',
    created_at     VARCHAR2(30) PATH '$.createdAt'
  )
) jt;
```

```
  COMMIT;
EXCEPTION
  WHEN OTHERS THEN
    IF UTL_FILE.IS_OPEN(l_file) THEN
      UTL_FILE.FCLOSE(l_file);
    END IF;
    RAISE;
END;
/
```

# ПРИЛОЖЕНИЕ Е.
## Запросы для создания ролей, пользователей и доступа к определенным объектам

```
CREATE ROLE c##app_user;
CREATE ROLE c##hotel_manager;
CREATE ROLE c##hotel_admin;

GRANT EXECUTE ON register_user TO PUBLIC;
GRANT EXECUTE ON get_available_rooms TO PUBLIC;
GRANT EXECUTE ON get_room_reviews TO PUBLIC;

GRANT EXECUTE ON book_room TO c##app_user;
GRANT EXECUTE ON cancel_booking TO c##app_user;
GRANT EXECUTE ON add_review TO c##app_user;
GRANT EXECUTE ON add_service TO c##app_user;
GRANT EXECUTE ON get_user_bookings TO c##app_user;
GRANT EXECUTE ON get_all_services TO c##app_user;

GRANT EXECUTE ON change_booking_status TO c##hotel_manager;
GRANT EXECUTE ON update_room_price TO c##hotel_manager;
GRANT EXECUTE ON remove_service_from_booking TO c##hotel_manager;
GRANT EXECUTE ON get_user_bookings TO c##hotel_manager;
GRANT EXECUTE ON get_all_services TO c##hotel_manager;

GRANT EXECUTE ON toggle_user_block TO c##hotel_admin;
GRANT EXECUTE ON assign_manager_role TO c##hotel_admin;
GRANT EXECUTE ON delete_user TO c##hotel_admin;
GRANT EXECUTE ON add_room TO c##hotel_admin;
GRANT EXECUTE ON update_room TO c##hotel_admin;
GRANT EXECUTE ON delete_room TO c##hotel_admin;
GRANT EXECUTE ON add_new_service TO c##hotel_admin;
GRANT EXECUTE ON update_service TO c##hotel_admin;
GRANT EXECUTE ON delete_service TO c##hotel_admin;
GRANT EXECUTE ON get_all_services TO c##hotel_admin;

CREATE USER c##client1 IDENTIFIED BY "UserPassword123";
CREATE USER c##manager1 IDENTIFIED BY "ManagerPass123";
CREATE USER c##admin1 IDENTIFIED BY "AdminPass123";

GRANT c##app_user TO c##client1;
GRANT c##hotel_manager TO c##manager1;
GRANT c##hotel_admin TO c##admin1;

GRANT CREATE SESSION TO c##app_user, c##hotel_manager, c##hotel_admin;

GRANT SELECT ON Rooms TO c##app_user, c##hotel_manager;
GRANT SELECT ON RoomTypes TO c##app_user, c##hotel_manager;
GRANT SELECT ON RoomServices TO c##app_user, c##hotel_manager;
GRANT SELECT, INSERT, UPDATE ON Bookings TO c##hotel_manager;
GRANT SELECT, INSERT ON Reviews TO c##app_user;
GRANT SELECT, INSERT ON BookingServices TO c##app_user, c##hotel_man-
ager;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON Users TO c##hotel_admin;
GRANT SELECT, INSERT, UPDATE, DELETE ON Rooms TO c##hotel_admin;
GRANT SELECT, INSERT, UPDATE, DELETE ON RoomTypes TO c##hotel_admin;
GRANT SELECT, INSERT, UPDATE, DELETE ON RoomServices TO c##hotel_admin;
GRANT SELECT, INSERT, UPDATE, DELETE ON Bookings TO c##hotel_admin;
GRANT SELECT, INSERT, UPDATE, DELETE ON BookingServices TO c##hotel_ad-
min;
GRANT SELECT, INSERT, UPDATE, DELETE ON Reviews TO c##hotel_admin;

GRANT EXECUTE ON book_room TO c##client1;
GRANT EXECUTE ON cancel_booking TO c##client1;
GRANT EXECUTE ON add_review TO c##client1;
GRANT EXECUTE ON add_service TO c##client1;
GRANT EXECUTE ON get_user_bookings TO c##client1;

GRANT EXECUTE ON change_booking_status TO c##manager1;
GRANT EXECUTE ON update_room_price TO c##manager1;
GRANT EXECUTE ON remove_service_from_booking TO c##manager1;
GRANT EXECUTE ON get_user_bookings TO c##manager1;

GRANT EXECUTE ON toggle_user_block TO c##admin1;
GRANT EXECUTE ON assign_manager_role TO c##admin1;
GRANT EXECUTE ON delete_user TO c##admin1;
GRANT EXECUTE ON add_room TO c##admin1;
GRANT EXECUTE ON update_room TO c##admin1;
GRANT EXECUTE ON delete_room TO c##admin1;
GRANT EXECUTE ON add_new_service TO c##admin1;
GRANT EXECUTE ON update_service TO c##admin1;
GRANT EXECUTE ON delete_service TO c##admin1;

GRANT EXECUTE ON register_user TO c##client1, c##manager1, c##admin1;
GRANT EXECUTE ON get_available_rooms TO c##client1, c##manager1, c##ad-
min1;
GRANT EXECUTE ON get_room_reviews TO c##client1, c##manager1, c##ad-
min1;

GRANT CREATE SESSION TO c##client1, c##manager1, c##admin1;

GRANT SELECT ON Rooms TO c##client1;
GRANT SELECT ON RoomTypes TO c##client1;
GRANT SELECT ON RoomServices TO c##client1;
GRANT SELECT, INSERT ON Reviews TO c##client1;
GRANT SELECT, INSERT ON BookingServices TO c##client1;

GRANT SELECT ON Rooms TO c##manager1;
GRANT SELECT ON RoomTypes TO c##manager1;
GRANT SELECT ON RoomServices TO c##manager1;
GRANT SELECT, INSERT, UPDATE ON Bookings TO c##manager1;
GRANT SELECT, INSERT ON BookingServices TO c##manager1;

GRANT SELECT, INSERT, UPDATE, DELETE ON Users TO c##admin1;
GRANT SELECT, INSERT, UPDATE, DELETE ON Rooms TO c##admin1;
GRANT SELECT, INSERT, UPDATE, DELETE ON RoomTypes TO c##admin1;
GRANT SELECT, INSERT, UPDATE, DELETE ON RoomServices TO c##admin1;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON Bookings TO c##admin1;
GRANT SELECT, INSERT, UPDATE, DELETE ON BookingServices TO c##admin1;
GRANT SELECT, INSERT, UPDATE, DELETE ON Reviews TO c##admin1;
```

# ПРИЛОЖЕНИЕ Ж.

## Запросы для создания триггера активации процедуры отправки email-уведомлений

```
CREATE OR REPLACE TRIGGER trg_booking_notify
AFTER INSERT OR UPDATE
   ON Bookings
FOR EACH ROW
DECLARE
   PRAGMA AUTONOMOUS_TRANSACTION;
   v_subject VARCHAR2(200);
   v_body    VARCHAR2(1000);
   v_action  VARCHAR2(10);
BEGIN
   IF INSERTING THEN
      v_action := 'created';
   ELSIF UPDATING THEN
      v_action := 'updated';
   END IF;

   v_subject := 'Booking ' || v_action || ': ID #' || :NEW.booking_id;
   v_body    := 'Dear user,' || CHR(10) ||
                'Your booking (ID ' || :NEW.booking_id || ') has been
' || v_action || '.' || CHR(10) ||
                'Details:' || CHR(10) ||
                '   User ID      : ' || :NEW.user_id    || CHR(10) ||
                '   Room ID      : ' || :NEW.room_id    || CHR(10) ||
                '    Check-in date: ' || TO_CHAR(:NEW.check_in_date,
'YYYY-MM-DD') || CHR(10) ||
                '   Check-out date: ' || TO_CHAR(:NEW.check_out_date,
'YYYY-MM-DD') || CHR(10) ||
                '   Status       : ' || :NEW.status    || CHR(10) ||
                '    Total price   : ' || TO_CHAR(:NEW.total_price,
'FM9999990.00') || CHR(10) || CHR(10) ||
                'Thank you for using our service.' || CHR(10) ||
                'Best regards,' || CHR(10) ||
                'Your Hotel Team';

   send_gmail_fixed(
     p_to      => 'zhkharchenko@gmail.com',
     p_subject => v_subject,
     p_body    => v_body
   );

   COMMIT;
EXCEPTION
   WHEN OTHERS THEN
      NULL;
END trg_booking_notify;
/
```