 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: «База данных для компании по бронированию жилья»

**Исполнитель**

студент 2 курса 1 группы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ П.И. Буйко

   подпись, дата

**Руководитель**

    Ассистент        \_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.С. Кантарович

должность, ученая степень, ученое звание           подпись, дата

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

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

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

                                                                                                                                                        дата, подпись

Курсовой проект защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

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

                                               подпись               дата      инициалы и фамилия

Минск 2024

Содержание

[Введение 5](#_Toc167361177)

[1. Анализ требований к программному средству 6](#_Toc167361178)

[1.1 Обзор аналогов 6](#_Toc167361179)

[1.1.1 Airbnb 6](#_Toc167361180)

[1.1.2 Booking.com 9](#_Toc167361181)

[2. Разработка функциональных требований, определение вариантов использования 12](#_Toc167361182)

[Вывод 13](#_Toc167361183)

[3. Анализ и проектирование модели базы данных. Описание информационных объектов и ограничений целостности. 13](#_Toc167361184)

[3.1 UML диаграмма схемы базы данных 13](#_Toc167361185)

[3.2. Описание информационных объектов. 13](#_Toc167361186)

[3.3 Вывод по разделу. 16](#_Toc167361195)

[4. Разработка модели базы данных 17](#_Toc167361196)

[4.1 Создание необходимых объектов 17](#_Toc167361197)

[4.1.1 Таблицы 17](#_Toc167361198)

[4.1.2 Процедура 17](#_Toc167361199)

[4.1.3 Триггеры 18](#_Toc167361200)

[4.1.4 Индексы 19](#_Toc167361201)

[5. Использование Orcale SQL 21](#_Toc167361202)

[5.1 Создание ролей для разграничения доступа 21](#_Toc167361203)

[5.2 Описание процедур экспорта 22](#_Toc167361204)

[5.3 Тестирование производительности базы данных 22](#_Toc167361205)

[5.4 Вывод 23](#_Toc167361206)

[6. Применение технологии резервного копирования 24](#_Toc167361207)

[6.1 Выбор между способами копирования 24](#_Toc167361208)

[6.2 Инструменты резервного копирования 25](#_Toc167361209)

[6.3 Процесс создания резервной копии 25](#_Toc167361210)

[6.4 Вывод 26](#_Toc167361211)

Введение

Цель данной работы заключается в разработке реляционной базы данных для системы бронирования жилья, которая предоставит эффективный способ хранения и управления различными данными, поддержку функционала в виде фильтрации, поиска, управления и аналитики.

База данных в контексте применения реляционной базы данных представляет собой организованный набор структурированных данных, организованных и хранимых в соответствии с моделью реляционной базы данных. В РБД данные организованы в виде таблиц, где каждая таблица представляет собой набор строк и столбцов, где каждая строка представляет отдельную запись данных, а каждый столбец - отдельное поле данных.

База данных обеспечивает удобный механизм для хранения, управления и доступа к данным, позволяя пользователям выполнять различные операции, такие как добавление, удаление, изменение и запросы данных. Она также обеспечивает целостность данных и защиту информации путем применения соответствующих механизмов контроля целостности и безопасности.

Основные пункты для выполнения:

* Определение ролей (администратор, пользователь, арендодатель);
* Реализовать функционал для пользователя: поиск жилья по городу/стране, оценка жилья, добавление и удаление активной брони, просмотр информации о других пользователях, оценка популярности жилья, просмотр статистики выкладываемых предложений за период.
* Реализовать функционал для администратора: просмотр информации о пользователях, управление пользователями (создание, блокирование, удаление учетных записей), назначение и изменение ролей другим пользователям, оценка популярности жилья, просмотр статистики выкладываемых предложений пользователем за период.
* Реализовать функционал для арендодателя: выставление жилья на площадку, управление своими объявлениями (редактирование, удаление, добавление дополнительной информации).
* Анализ работы (оценка популярности жилья, статистика выкладываемых предложений пользователем за период)

1. Анализ требований к программному средству

1.1 Обзор аналогов

Для создания качественного проекта следует изучить существующие аналогичные решения на рынке или альтернативные подходы к решению проблемы. Это позволяет понять, какие решения уже существуют, и может дать идеи для выбора оптимального варианта.

1.1.1 Airbnb

Airbnb использует базу данных для управления своим сервисом бронирования жилья. Она содержит информацию о доступных апартаментах, домах, хозяевах, гостях и прочее.

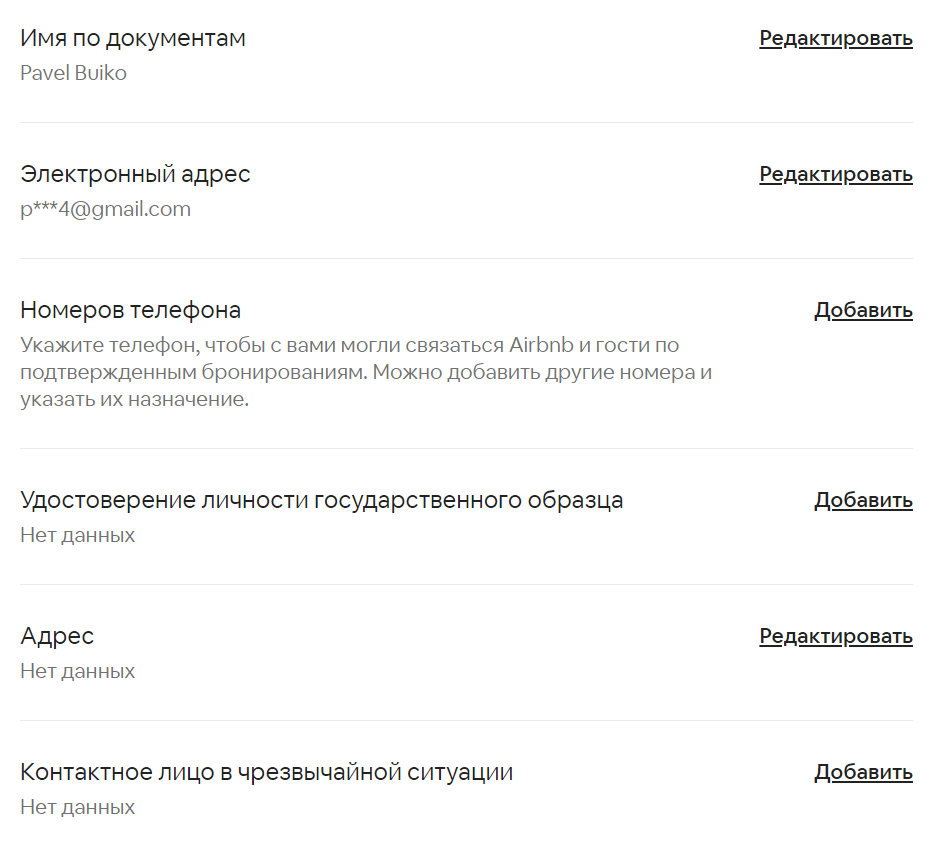


Рисунок 1.1 – Элемент интерфейса аналога AirBnb

На скриншоте можно обнаружить, что Airbnb хранит информацию, который пользователь вводит о себе, однако и позволяет её не заполнять. Хранимые данные:

* Имя (по документам)
* Электронный адрес
* Номер телефона
* Удостоверение личности(фото)
* Адрес
* Контактное лицо в ЧС

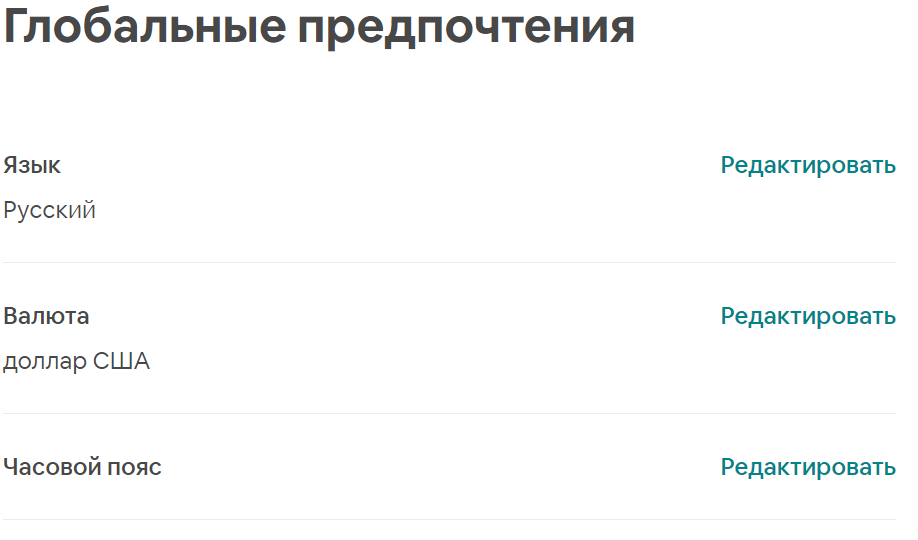


Рисунок 1.2 – Хранимая информация аналогом AirBnb

Хранит и полезную информацию для качественного опыта взаимодействия пользователя. Предпочитаемый язык, валюту, часовой пояс.

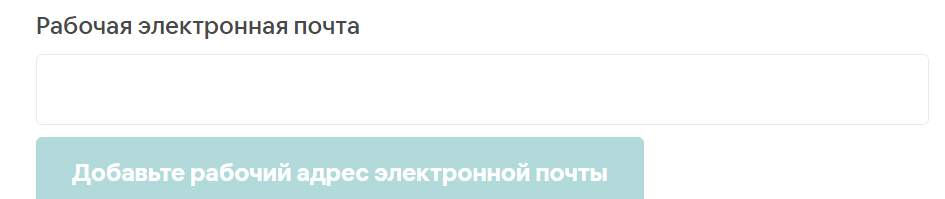


Рисунок 1.3 – Ввод адреса электронной почты

Предоставляет возможность хранения рабочей почты.

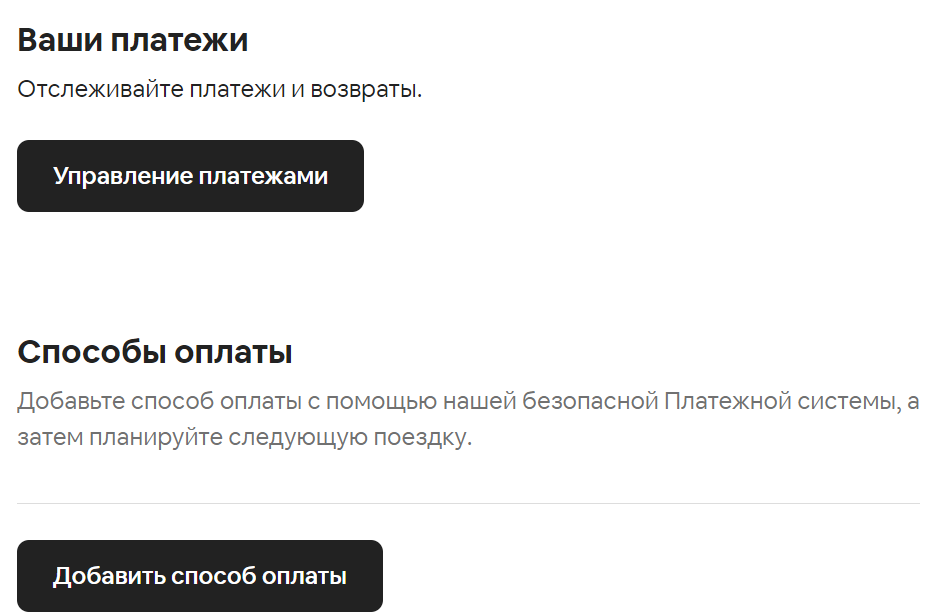


Рисунок 1.4 – Пример хранимых данных о средстве платежа

Хранит информацию о способе оплаты, а также историю платежей.

Переходя к карточке товара, можно обнаружить следующее:



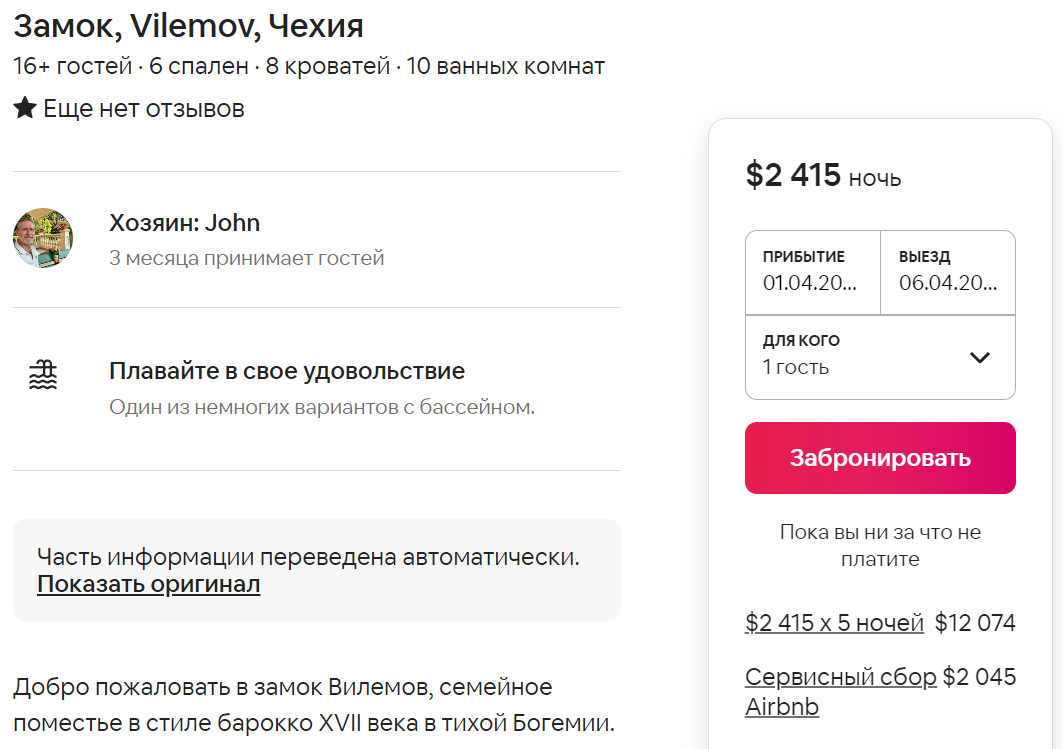


Рисунок 1.5 – Карточка товара

Сервис хранит название выставляемого жилья, фотографии, описание, местоположение, количество и наличие отзывов, цена за сутки, состав комнат, выводит информацию о хозяине.

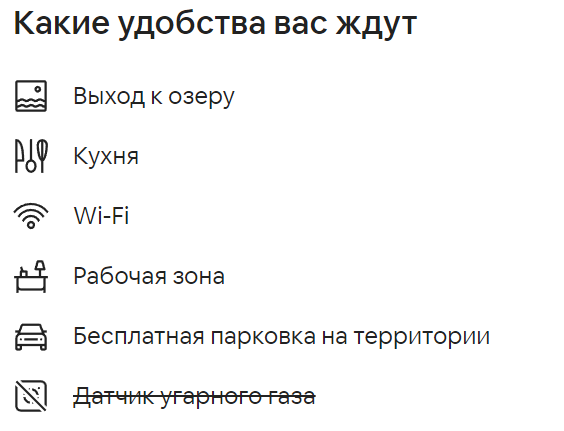


Рисунок 1.6 – Удобства жилья

На сайте так же доступна информация о каких-либо удобствах.



Рисунок 1.7 – Календарь бронирования

Можно так же заметить, что хранится информация о бронированиях на жильё.



Рисунок 1.8 – Оценочная система

Отдельное оценивание аспектов жилья так же представлено на сайте.

1.1.2 Booking.com

Можно подчеркнуть некоторые важные моменты и в системе Booking.com. В некоторых фактах можно убедиться и понять, что они обязательны при разработке базы данных.

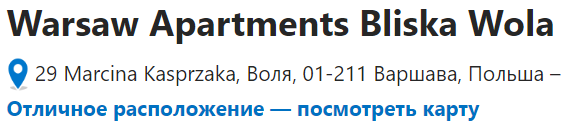


Рисунок 1.9 – Местоположение пользователя

Можно снова же обнаружить хранение информации о местоположении.

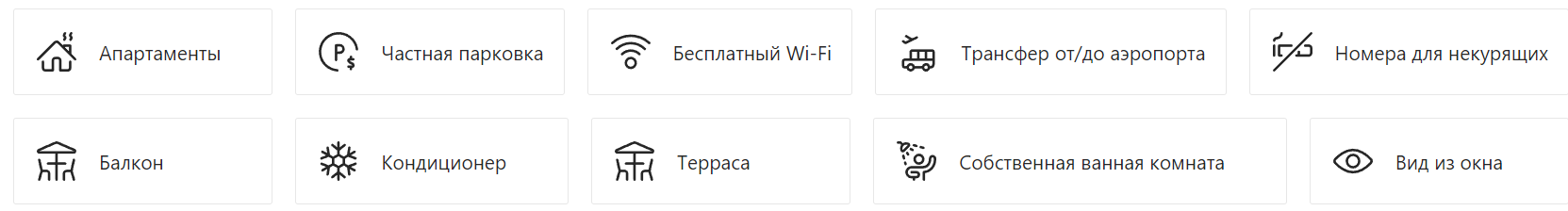


Рисунок 1.10 – Информация об удобствах

Так же можно обнаружить информацию об удобствах жилья.

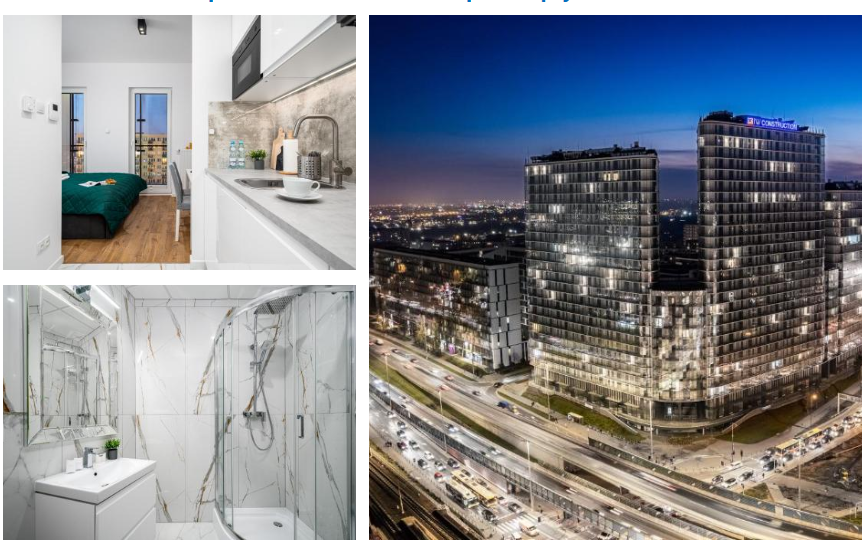


Рисунок 1.11 – Фотографии жилья

Сервис, что и предполагается, тоже хранит фотографии жилья.

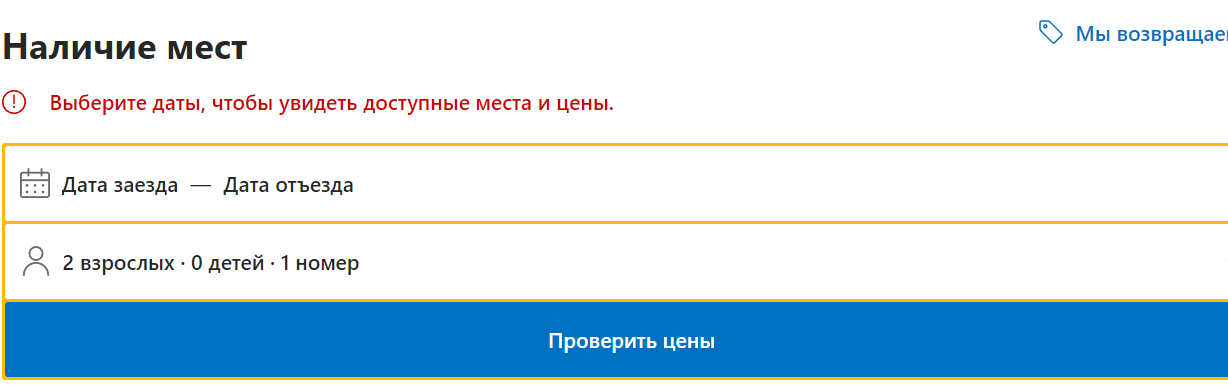


Рисунок 1.12 – Система бронирования

Забронированные и свободные даты так же являются хранимой информацией

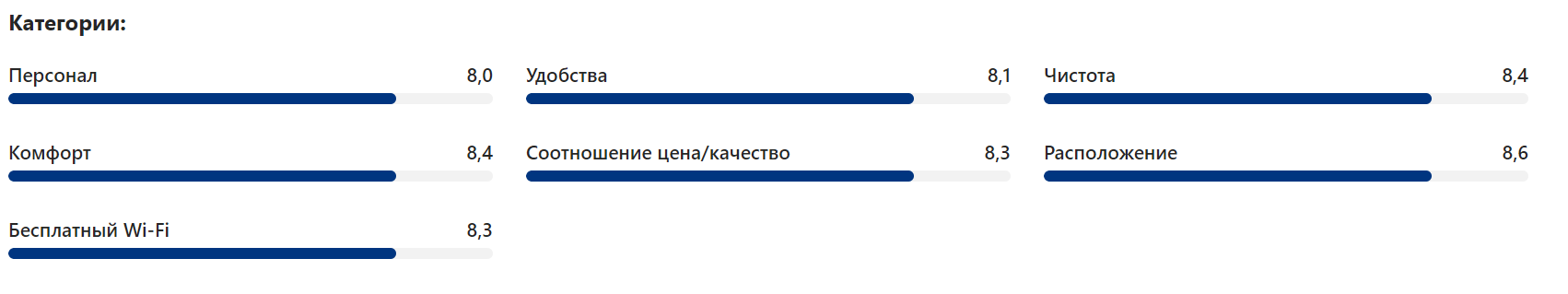


Рисунок 1.12 – Разделенные оценки

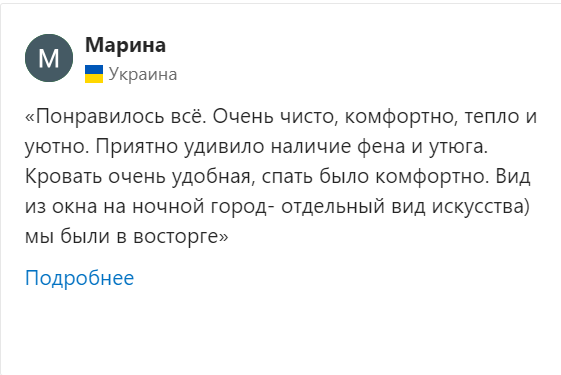


Рисунок 1.13 – Отдельный отзыв пользователя

Отзывы так же являются частью хранимой информации.

Категории хранимой информации, являющиеся общими для баз данных сервисов бронирования жилья, включают:

Предложения по размещению:

* Информация об отелях, апартаментах, домах и других видов жилья.
* Описание помещений, фотографии, видео.
* Список удобств и услуг.
* Цены и доступность.

Клиенты:

* Информация о клиентах, включая их контактные данные.
* История платежей.

Бронирования:

* Даты заезда и выезда.
* Дополнительные услуги (например, завтрак, трансфер).
* Специальные запросы и комментарии.

Хозяева (для платформ, где предоставляется жилье от частных лиц):

* Информация о хозяевах, включая их контактные данные.
* Описание жилья, цены, правила проживания.
* Отзывы и оценки:
* Отзывы от клиентов о размещении.
* Оценки качества и уровня сервиса.

Финансовая информация:

* Платежные данные клиентов.
* Счета, квитанции, история платежей.

2. Разработка функциональных требований, определение вариантов использования

Функциональное требование — это заявление о том, как должна вести себя система. Он определяет, что система должна делать, чтобы удовлетворить потребности или ожидания пользователя. Функциональные требования можно рассматривать как функции, которые обнаруживает пользователь. Они отличаются от нефункциональных требований, которые определяют, как система должна работать внутри (например, производительность, безопасность и т. д.).

Функциональные требования могут различаться в зависимости от роли:

1. Администратор:

* Создание, редактирование и удаление учетных записей пользователей.
* Управление правами доступа и ролями пользователей в системе.
* Добавление, редактирование и удаление информации об объектах размещения (отелях, апартаментах).
* Модерация отзывов и комментариев пользователей.
* Просмотр статистики и отчетов о деятельности сервиса.

1. Обычный пользователь (Клиент):

* Поиск доступных объектов размещения по различным критериям (цена, местоположение, удобства и т.д.).
* Бронирование жилья на определенные даты с указанием количества гостей и других параметров.
* Редактирование личной информации (имя, контактная информация и т.д.).
* Просмотр истории бронирований и отзывов.
* Оценка и оставление отзывов о местах размещения после проживания.

1. Арендодатель:

* Добавление новых апартаментов в систему с указанием описания, фотографий, цены и доступности.
* Редактирование информации о своих апартаментах.
* Просмотр и управление бронированиями для своих апартаментов.
* Регистрация и аутентификация в системе как владелец апартаментов.
* Редактирование личной информации (имя, контактная информация и т.д.).
* Просмотр истории бронирований и отзывов о своих апартаментах.
* Просмотр отзывов и комментариев о своих апартаментах.
* Возможность отвечать на отзывы и реагировать на негативные комментарии.

Каждый тип пользователя имеет свои уникальные ожидания от функционала сервиса, и система должна обеспечивать соответствующий функционал для удовлетворения их потребностей. Это будет достигаться за счёт реализации ролей и создания процедур и таблиц, доступных лишь конкретной роли.

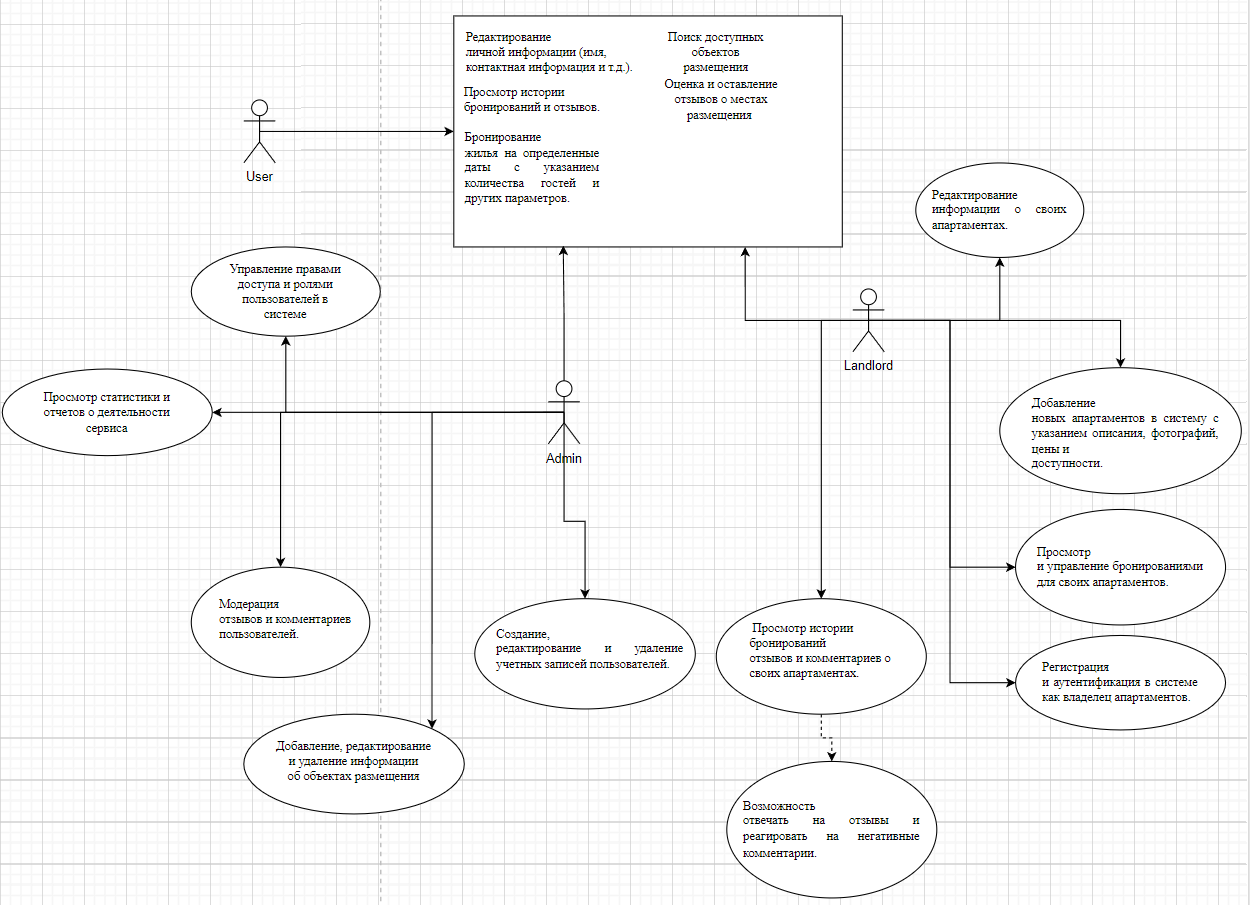


Рисунок 2.1 – Система ролей UML-диаграмма

Вывод

Описание функциональных требований подчеркивает важность удовлетворения различных потребностей пользователей системы через конкретные функции. Администраторы управляют учетными записями и контентом, обычные пользователи могут искать и бронировать жилье, а арендодатели добавляют и управляют своими объектами. Каждый тип пользователя имеет уникальные ожидания, которые система должна учитывать, обеспечивая соответствующий функционал для каждой роли. Это достигается через реализацию специфических ролей и ограничений доступа.

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

3.1 UML диаграмма схемы базы данных

В проектируемой базе данных располагается 9 таблиц, 8 являются хранилищами сущностей и 1 используется для реализации ManyToMany. Схема базы данных указана в приложении 1

3.2. Описание информационных объектов и ограничений целостности.

Далее будет описана структура каждой таблицы. В таблице users хранятся данные для идентификации пользователей, зарегистрированных в приложении, и пользователей, имеющих права модератора, addresses – адреса каких-либо помещений, properties – таблица, хранящая данные о жилье,payments– таблица, хранящая данные об способах оплаты пользователя, ratings– хранит оценки жилья, оставленные пользователями, bookings – хранит информацию о бронированиях, сделанных пользователями на конкретном жилье. Photos – URL адреса картинок, которые присваиваются какому либо жилью.

3.2.1 Таблица Users

Содержит информацию о пользователях системы.

* id: Уникальный идентификатор пользователя, автоматически генерируемый.
* first\_name: Имя пользователя, не может быть пустым.
* last\_name: Фамилия пользователя, не может быть пустой.
* email: Электронная почта пользователя, уникальное значение, не может быть пустым.
* password: Пароль пользователя (стоит рассмотреть использование более безопасного типа данных для пароля).
* age: Возраст пользователя.
* role: Роль пользователя в системе, может быть admin, user или landlord.

3.2.2 Таблица payments

Содержит информацию о платежах, связанных с пользователями.

* id: Уникальный идентификатор платежа, автоматически генерируемый.
* user\_id: Идентификатор пользователя, к которому относится платеж, внешний ключ к таблице users.
* card\_number: Номер карты для оплаты.
* card\_owner: Имя владельца карты.
* expiration\_date: Дата окончания действия карты.
* cvv\_code: CVV код карты.

3.2.3 Таблица addresses

Содержит информацию об адресах недвижимости.

* id: Уникальный идентификатор адреса, автоматически генерируемый.
* street\_title: Название улицы.
* street\_address: Номер дома на улице.
* city: Город.
* state: Штат или область.
* postal\_code: Почтовый индекс.
* country: Страна.
* apartment\_number: Номер квартиры (если есть).

3.2.4 Таблица properties

Содержит информацию о недвижимости.

* id: Уникальный идентификатор недвижимости, автоматически генерируемый.
* user\_id: Идентификатор пользователя-владельца, внешний ключ к таблице users.
* address\_id: Идентификатор адреса, внешний ключ к таблице addresses.
* title: Заголовок объявления о недвижимости.
* type: Тип недвижимости (квартира, дом, комната).
* description: Описание недвижимости.
* rooms: Количество комнат.
* is\_available: Статус доступности недвижимости.
* price: Цена за аренду недвижимости.

3.2.5 Таблица amenities

Содержит информацию об удобствах, доступных в недвижимости.

* id: Уникальный идентификатор удобства, автоматически генерируемый.
* name: Название удобства, должно быть уникальным.

3.2.6 Таблица photoes

Содержит фотографии недвижимости.

* id: Уникальный идентификатор фотографии, автоматически генерируемый.
* property\_id: Идентификатор недвижимости, к которой относится фотография, внешний ключ к таблице properties.
* url: URL фотографии.
* order: Порядок отображения фотографии.

3.2.7 Таблица bookings

Содержит информацию о бронированиях.

* id: Уникальный идентификатор бронирования, автоматически генерируемый.
* user\_id: Идентификатор пользователя, совершившего бронирование, внешний ключ к таблице users.
* property\_id: Идентификатор недвижимости, которую забронировали, внешний ключ к таблице properties.
* payment\_id: Идентификатор платежа, связанного с бронированием, внешний ключ к таблице payments.
* check\_in: Дата заселения.
* check\_out: Дата выселения.
* guests: Количество гостей.
* total\_price: Общая стоимость бронирования.
* status: Статус бронирования (approved, denied, waiting).

3.2.8 Таблица ratings

Содержит отзывы и рейтинги для недвижимости.

* id: Уникальный идентификатор рейтинга, автоматически генерируемый.
* user\_id: Идентификатор пользователя, оставившего рейтинг, внешний ключ к таблице users.
* property\_id: Идентификатор недвижимости, для которой оставлен рейтинг, внешний ключ к таблице properties.
* rating: Рейтинг недвижимости (от 1 до 5).

Уникальное сочетание user\_id и property\_id, чтобы один пользователь мог оставить только один отзыв для одной недвижимости.

3.3 Вывод по разделу.

В данном разделе была разработана архитектура проекта, составлена обобщенная структура управления приложением и диаграмма UML. Были описаны все таблицы базы данных и ограничения целостности к строкам таблицы.

4. Разработка модели базы данных

4.1 Создание необходимых объектов

4.1.1 Таблицы

Таблица представляет собой набор взаимосвязанных данных, организованных в структурированном виде в базе данных. Она включает столбцы и строки. В рамках данного курсового проекта база данных включает 8 таблиц, которые детально описаны в главе 3.

|  |
| --- |
| CREATE TABLE users (  id NUMBER(20) GENERATED BY DEFAULT ON NULL AS IDENTITY PRIMARY KEY,  first\_name VARCHAR2(30) NOT NULL,  last\_name VARCHAR2(50) NOT NULL,  email VARCHAR2(150) NOT NULL UNIQUE,  password NUMBER(10) NOT NULL,  age NUMBER(3) NOT NULL,  role VARCHAR2(10) NOT NULL CHECK (role IN ('admin', 'user', 'landlord'))  ); |

Листинг 4.1 – Пример создания таблицы

4.1.2 Процедура

Процедуры — это именованные блоки кода на языке PL/pgSQL, хранящиеся в базе данных и вызываемые по мере необходимости. Они предназначены для группировки и повторного использования кода, который должен выполняться многократно, а также для снижения сетевой нагрузки при взаимодействии с базой данных. Процедуры могут включать SQL-запросы, циклы, условные операторы, переменные и другие элементы.

Список основных процедур, разработанных в рамках курсового проекта:

* add\_user используется для добавления нового пользователя в таблицу users.
* add\_property используется для добавления нового объекта недвижимости в таблицу properties, а также для добавления связанного с ним адреса в таблицу addresses
* add\_payment используется для добавления данных о платежной карте в таблицу payments
* add\_booking используется для создания нового бронирования в таблицу bookings
* add\_property\_amenity используется для связывания недвижимости с определенным удобством.
* add\_photos\_to\_property используется для добавления фотографий к недвижимости.
* add\_rating используется для добавления оценки (отзыва) к недвижимости.
* get\_user\_details используется для получения информации о пользователе по его ID.
* update\_booking\_status используется для изменения статуса бронирования.
* get\_payment\_details используется для получения информации о платежной карте по ее ID.
* get\_address\_details используется для получения информации об адресе по его ID.
* get\_property\_details используется для получения информации об объекте недвижимости по его ID.
* get\_booking\_by\_id используется для получения информации о бронировании по его ID.
* get\_bookings\_by\_user\_id используется для получения списка бронирований, сделанных пользователем.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE update\_user(  p\_user\_id IN users.id%TYPE,  p\_first\_name IN users.first\_name%TYPE,  p\_last\_name IN users.last\_name%TYPE,  p\_email IN users.email%TYPE,  p\_age IN users.age%TYPE,  p\_role IN users.role%TYPE  )  AS  BEGIN  UPDATE users  SET first\_name = p\_first\_name,  last\_name = p\_last\_name,  email = p\_email,  age = p\_age,  role = p\_role  WHERE id = p\_user\_id;  COMMIT;  END; |

Листинг 4.2 – Пример создания процедуры

4.1.3 Триггеры

Триггер — это объект базы данных, который автоматически выполняет заданный блок кода при возникновении определенных событий, таких как вставка, обновление или удаление данных в таблице. Триггеры позволяют обеспечить целостность данных, автоматизировать обработку и отслеживать изменения в базе данных.

В базе данных для данного курсового проекта определены следующие триггеры:

* users\_trigger: Этот триггер срабатывает после вставки, обновления или удаления записи в таблице users.
* bookings\_trigger: Этот триггер срабатывает после вставки, обновления или удаления записи в таблице bookings.
* properties\_trigger: Этот триггер срабатывает после вставки, обновления или удаления записи в таблице properties.
* check\_user\_unique: Этот триггер выполняется перед вставкой новой записи в таблицу users и проверяет уникальность email.
* check\_landlord\_property: Этот триггер проверяет, имеет ли пользователь роль landlord или admin.
* check\_past\_booking\_dates: Этот триггер выполняется перед вставкой новой записи в таблицу bookings и проверяет, чтобы даты бронирования не были в прошлом.
* check\_overlapping\_bookings: Этот триггер выполняется перед вставкой или обновлением записи в таблице bookings.
* check\_guests: Этот триггер выполняется перед вставкой или обновлением записи в таблице bookings и проверяет, чтобы количество гостей было больше нуля.

|  |
| --- |
| CREATE OR REPLACE TRIGGER users\_trigger  AFTER INSERT OR UPDATE OR DELETE ON users  FOR EACH ROW  BEGIN  IF INSERTING THEN  DBMS\_OUTPUT.PUT\_LINE('New user inserted with ID: ' || :NEW.id);  ELSIF UPDATING THEN  DBMS\_OUTPUT.PUT\_LINE('User with ID: ' || :OLD.id || ' updated');  ELSIF DELETING THEN  DBMS\_OUTPUT.PUT\_LINE('User with ID: ' || :OLD.id || ' deleted');  END IF; |

Листинг 4.3 – Пример триггера

4.1.4 Индексы

Индексы являются объектами базы данных, которые ускоряют выполнение запросов на поиск, сортировку или группировку данных в таблицах. Индексы представляют собой специальные структуры данных, которые позволяют быстро находить записи в таблице на основе значений столбцов, входящих в индекс.

В базе данных для данного курсового проекта определены следующие индексы:

* idx\_properties\_type\_city\_price\_available: Индекс на таблице properties для столбцов type, address\_id, price и is\_available.
* idx\_properties\_id: Индекс на таблице properties для столбца id.
* idx\_photoes\_property\_id\_order: Индекс на таблице photoes для столбцов property\_id и order.
* idx\_bookings\_user\_id: Индекс на таблице bookings для столбца user\_id.
* idx\_bookings\_property\_id: Индекс на таблице bookings для столбца property\_id.
* idx\_ratings\_property\_id: Индекс на таблице ratings для столбца property\_id.
* idx\_addresses\_duplicate\_check: Индекс на таблице addresses для столбцов street\_title, street\_address и apartment\_number.

|  |
| --- |
| CREATE INDEX idx\_properties\_type\_city\_price\_available  ON properties (type, address\_id, price, is\_available); |

Листинг 4.3 – Пример созданного индекса

Таким образом с помощью индексов достигается повышение производительности базы данных

5. Использование Orcale SQL

Oracle Database — это объектно-реляционная система управления базами данных (СУБД), разработанная и поддерживаемая компанией Oracle Corporation.

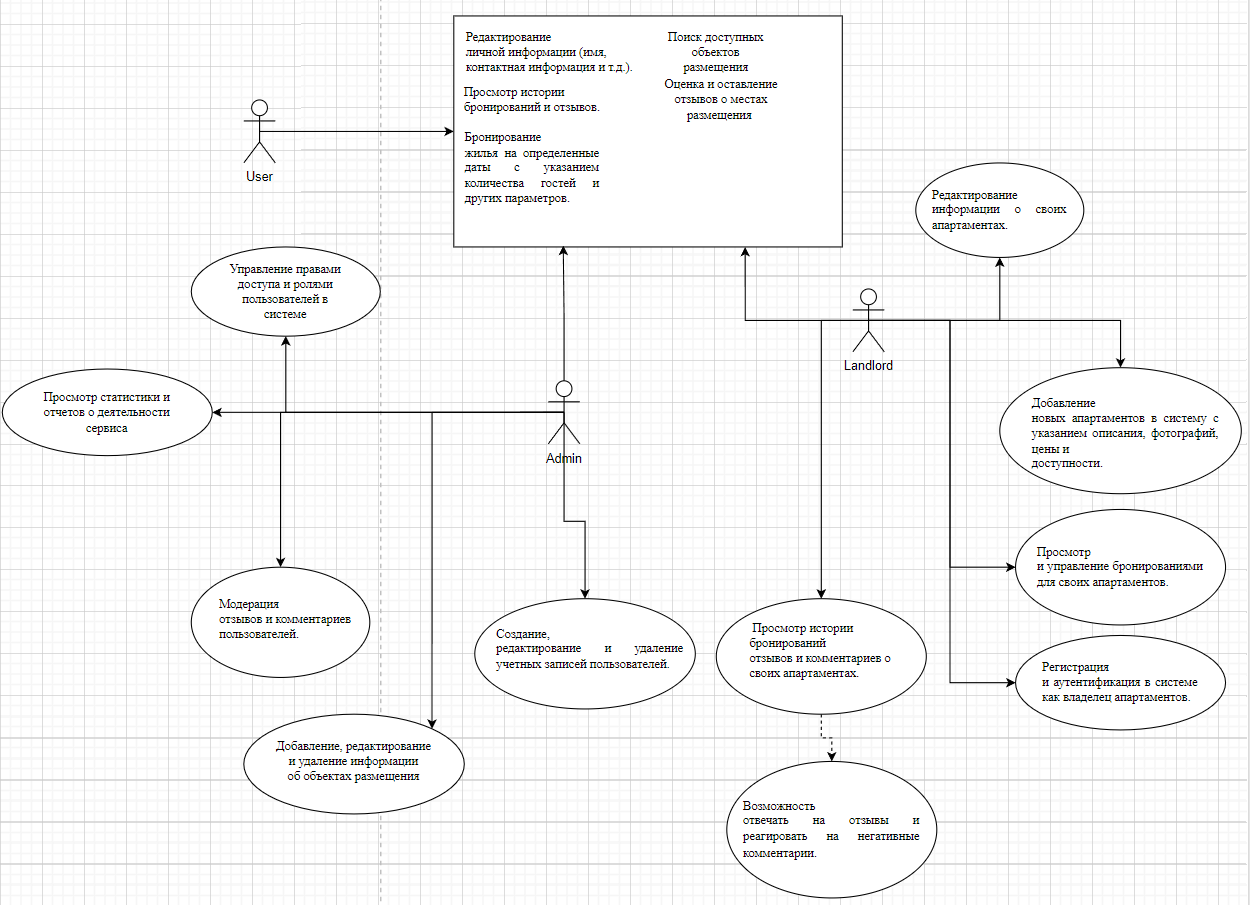


Рисунок 5.1 – диаграмма ролей пользователей

5.1 Создание ролей для разграничения доступа

Для разграничения доступа создаются три роли: пользователь, администратор, арендодатель. Для начала опишем возможности пользователя, которые являются общими для всех ролей:

* Регистрация.
* Просмотр истории бронирований.
* Просмотр отзывов.
* Бронирование жилья.
* Поиск доступных объектов жилья.
* Оценка и оставление отзывов.

Возможности администратора:

* Управление ролями пользователей.
* Просмотр статистики о работе сервиса.
* Модерация отзывов.
* Создание учётных записей.
* Манипуляции с данными объектов размещения.

Арендодатель:

* Добавление новых объектов размещения в систему.
* Редактирование записей об объектах размещения.
* Просмотр и управление бронированиями его жилья.
* Просмотр истории бронирований.

5.2 Описание процедур экспорта

JSON — это формат, который хранит структурированную информацию и в основном используется для передачи данных между сервером и клиентом.

Файл JSON представляет собой более простую и лёгкую альтернативу расширению с аналогичными функциями XML (Extensive Markup Language).

Часто возникает необходимость импортировать и экспортировать JSON-файлы, в данной курсовой работе используются функция row\_to\_json() для экспорта. Процедура экспорта используются для таблицы Posts, так как эта таблица является важнейшей в базе данных. Пример создания процедуры можно посмотреть в листинге 5.1.

|  |
| --- |
| CREATE OR REPLACE PROCEDURE export\_users\_to\_json AS  v\_json CLOB;  v\_chunk\_size CONSTANT INTEGER := 4096;  v\_start\_pos INTEGER := 1;  v\_end\_pos INTEGER;  v\_chunk VARCHAR2(4096);  file\_handle UTL\_FILE.FILE\_TYPE;  BEGIN |

Листинг 5.1 – пример создания процедуры работы с JSON

5.3 Тестирование производительности базы данных

Производительность БД является решающим фактором эффективности управленческих и коммерческих приложений. Если поиск или запись данных выполняется медленно – способность к нормальной работе приложения падает. Существует единственный путь выяснить причину плохой производительности – выполнить количественные измерения и определить, что является причиной проблемы производительности.

Для того чтобы правильно организовать процесс тестирования БД, тестировщики должны обладать хорошими знаниями SQL и DML и иметь ясное представление о внутренней структуре БД. Это самый лучший и надежный способ тестирования БД особенно для приложений с низким и средним уровнем сложности. Данный метод не только дает уверенность, что тестирование выполнено качественно, но также повышает мастерство написания SQL-запросов.

Оптимизация работы базы данных осуществляется с помощью индексов:

|  |
| --- |
| CREATE INDEX idx\_addresses\_duplicate\_check  ON addresses (street\_title, street\_address, apartment\_number); |

Листинг 5.2 – пример индекса

Этот код демонстрирует индекс, созданный по нескольким столбцам.

В данной ситуации при тестировании базы данных удалось добиться улучшения производительности на 25 процентов. Ниже представлен SELECT запрос без создания индекса:



Рисунок 5.2 – SELECT запрос без индекса

На рисунке 5.3 представлен результат SELECT запроса с использование индекса:

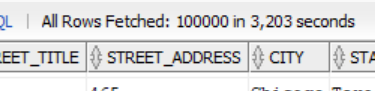


Рисунок 5.3 – SELECT запрос с использованием индекса

Остальыне индексы представлены в приложении 1.

5.4 Вывод

В этом разделе была описана выбранная СУБД, рассказано о разработанных процедурах, триггерах, функциях и индексах. Кроме того, продемонстрирован листинг экспорта данных из таблицы posts в json файл. Также была проверена работоспособность разработанной БД посредством добавления в одну из таблиц 100000 строк.

6. Применение технологии резервного копирования и восстановления

Резервным копированием называется сохранение копии данных где-то вне основного места их хранения.

Главное назначение резервного копирования – восстановление данных после их потери. В связи с этим нередко приходится слышать, что при наличии реплики базы данных с неё всегда можно восстановить данные, и резервное копирование не нужно. На самом деле резервное копирование позволяет решить как минимум три задачи, которые не могут быть решены при помощи реплики, да и реплику без резервной копии не инициализировать.

6.1 Выбор между способами копирования

Очевидная идея заключается в том, чтобы остановить базу данных и скопировать все её файлы, создавая таким образом «холодную» резервную копию. Этот способ крайне надёжный и простой, но у него есть два очевидных недостатка. Во-первых, из «холодной» резервной копии можно восстановить только то состояние базы данных, которое было в момент её останова; транзакции, сделанные после рестарта базы, в «холодную» резервную копию не попадут. Во-вторых, не у каждой базы данных есть технологическое окно, когда её можно остановить.

«Горячее» сохранение файлов выполняется путём копирования файлов базы данных без её остановки, что характерно для большинства современных систем. Однако при этом возникают несколько проблем. Во-первых, в момент начала копирования содержимое базы данных может не совпадать с содержимым файлов, так как часть информации находится в кеше и ещё не записана на диск. Во-вторых, во время копирования содержимое базы данных может изменяться: если используются изменяемые структуры данных, то меняется содержимое файлов; при использовании неизменяемых структур меняется набор файлов, с новыми файлами и удалением старых. В-третьих, из-за отсутствия синхронизации между записью данных в базу и чтением файлов БД, программа резервного копирования может прочитать некорректную страницу, где половина данных будет от старой версии страницы, а другая половина — от новой.

Таким образом можно понять, что для полноценной симуляции рабочей базы данных следует воспользоваться созданием горячей копии.

6.2 Инструменты резервного копирования

Существует множество инструментов для создания резервных копий баз данных: RMAN, Data pump, Oracle secure backup и т.д.

В рамках курсового проекта будет использоваться самое популярное решение: Oracle Recovery Manager(RMAN). RMAN широко используется благодаря своим многочисленным преимуществам:

* Интеграция с Oracle Database: RMAN является встроенным инструментом, который не требует установки дополнительных программ и тесно интегрирован с Oracle Database, что обеспечивает высокую совместимость и надежность.
* Функциональность: RMAN поддерживает как горячие, так и холодные резервные копии, инкрементальные резервные копии, архивацию redo-логов, восстановление до определенного момента времени (PITR) и другие продвинутые функции резервного копирования и восстановления.
* Автоматизация: RMAN позволяет автоматизировать процессы резервного копирования и восстановления с помощью скриптов и планировщиков задач, что снижает вероятность ошибок и упрощает управление.
* Оптимизация и производительность: RMAN оптимизирует использование ресурсов системы, поддерживает сжатие и дедупликацию данных, что помогает снизить объемы хранилища и улучшить производительность.
* Поддержка Data Guard: RMAN интегрируется с Oracle Data Guard для обеспечения защиты данных и аварийного восстановления, предоставляя дополнительные возможности для повышения отказоустойчивости системы.

Благодаря этим возможностям RMAN считается стандартом де-факто для резервного копирования и восстановления баз данных Oracle и используется большинством организаций, работающих с Oracle Database.

6.3 Процесс создания резервной копии

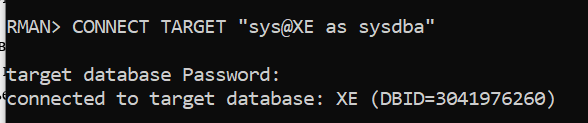


Рисунок 6.1 – Подключение к базе данных через RMAN

Подключение осуществляется стандартными командами RMAN.

Подключение обязательно осуществлять с административными привилегиями, иначе программа не даст возможности создать резервную копию.

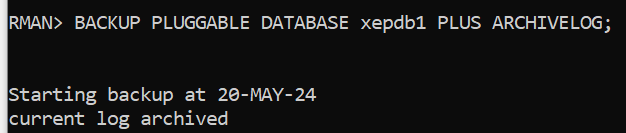


Рисунок 6.2 – Выполнение горячего архивирования базы данных

Для архивации был выбран режим, включающий архив логи ведь это хорошее решение для проекта по нескольким причинам, связанным с обеспечением надежности, целостности данных и возможности восстановления в случае сбоев:

* Горячее и полное резервное копирование базы данных (DATABASE):
* Резервное копирование архивных журналов (ARCHIVELOG)
* Резервное копирование Pluggable Database (PLUGGABLE)
* Простота и удобство использования.

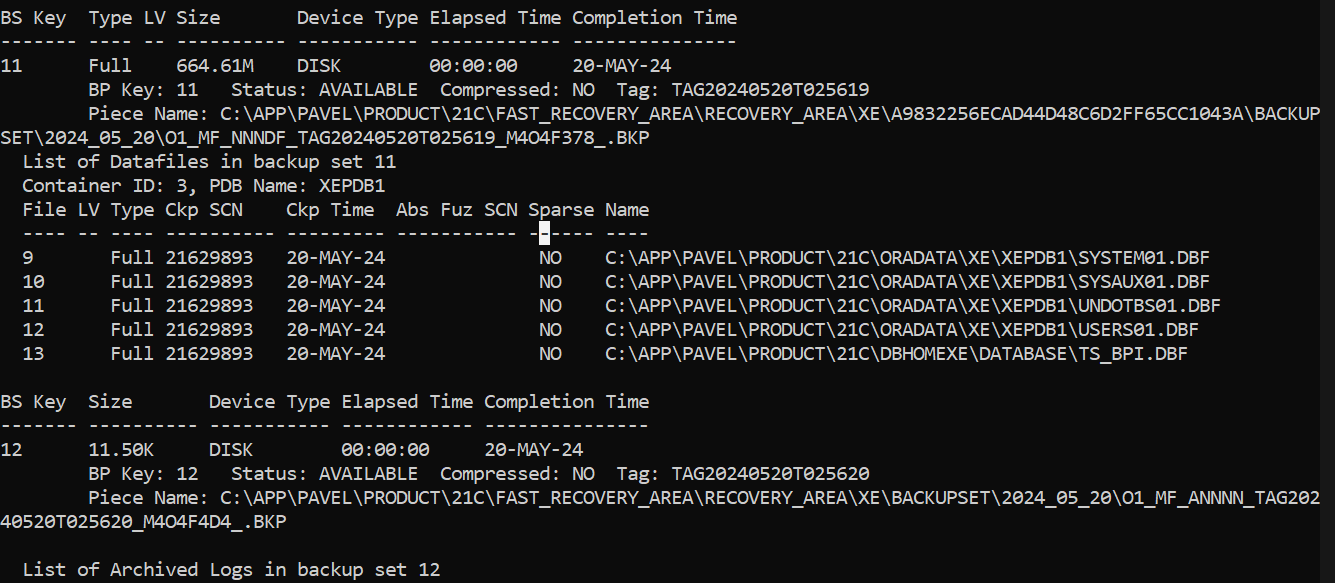


Рисунок 6.3 – Запись о копии в списке резервных копий

6.4 Вывод

На основании проведенной работы над созданием резервной копии базы данных Oracle с использованием Oracle RMAN можно сделать следующий вывод: Создание надежной и регулярно обновляемой резервной копии базы данных является критически важным аспектом для любой организации, оно обеспечивает безопасность данных и защиту при ошибках сервера, системы или проблем, вызванных человеческим фактором, проделанная работа обеспечила надежное и эффективное управление резервными копиями базы данных Oracle, что является ключевым аспектом обеспечения безопасности и целостности данных в корпоративной среде.

Список использованных источников

1.Oracle Help Centre [Электронный ресурс] / Режим доступа: URL https://docs.oracle.com/en/- Дата доступа: 22.04.2023

2. Yandex Dzen [Электронный ресурс] / Режим доступа: URL https://dzen.ru/a/ - Дата доступа: 20.04.2023

3. Stackoverflow [Электронный ресурс] / Режим доступа: URL https://stackoverflow.com/ - Дата доступа: 12.04.2023

4. Habr [Электронный ресурс] / Режим доступа: URL https://habr.com/en/ - Дата доступа: 12.03.2023

5. AirBnb [Электронный ресурс] / Режим доступа: URL https://www.airbnb.com/ - Дата доступа: 17.03.2023

6. Booking.com [Электронный ресурс] / Режим доступа: URL https://www.booking.com/ - Дата доступа: 19.03.2023