МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий
Кафедра Информационные системы и технологии
Специальность 1–40 01 01 Программное обеспечение информационных технологий

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ НА ТЕМУ:

«Реализация базы данных отеля с применением средств мониторинга состояния СУБД»

Выполнил студент Гурина Кристина Сергеевна
(Ф.И.О.)

Руководитель работы ассист. Нистюк Ольга Александровна
(учен. степень, звание, должность, Ф.И.О., подпись)

И.о. зав. кафедрой ст. преп. Блинова Е.А.
(учен. степень, звание, должность, Ф.И.О., подпись)

Курсовая работа защищена с оценкой _____

Минск 2023

Содержание

Введение

В наше современное время существует огромное спрос на хранилища информации, известные как базы данных, которые способны постоянно функционировать и обеспечивать удобный доступ к хранящейся информации. Важно также учитывать необходимость использования средств мониторинга состояния хранилища данных. В данном контексте, применение баз данных в сфере гостиничного бизнеса, в частности, отелей, для обработки и хранения информации о состоянии гостиничных комплексов и бронирования номеров, становится особенно актуальным.

В свете современных требований и актуальности данной работы, целью данного курсового проекта является разработка и внедрение базы данных Oracle и соответствующего интерфейса для отелей.

Задачи данного проекта включают:

1. Аналитический обзор литературы по теме проекта и изучение требований.
2. Определение вариантов использования базы данных в сфере гостиничного бизнеса.
3. Анализ и проектирование модели данных, включая описание информационных объектов и ограничений целостности.
4. Создание необходимых объектов в базе данных.
5. Разработка механизмов импорта и экспорта данных для обеспечения переносимости информации.
6. Описание используемой технологии и механизмов мониторинга состояния базы данных.
7. Тестирование производительности разработанной системы базы данных.
8. Формирование выводов по каждому разделу и подготовка заключения, включающего обобщение результатов проделанной работы.

Для эффективной реализации базы данных отеля, необходимо использовать комплексное программное обеспечение, известное как система управления базами данных (СУБД). СУБД служит важным интерфейсом между базой данных и пользователями или программами, обеспечивая пользователям возможность получать и обновлять информацию, а также управлять ее структурой и оптимизацией. Кроме того, СУБД обеспечивает контроль и управление данными, включая мониторинг производительности, настройку, а также резервное копирование и восстановление информации.

Сфера применения систем управления базами данных в современном мире практически бесконечна, и они используются в различных областях, включая интернет, производство, промышленность, маркетинг, мобильные устройства, финансовую и банковскую сферу, телевидение, телекоммуникации и рекламу. Внедрение средств мониторинга состояния СУБД в сфере гостиничного бизнеса, такого как отели, помогает обеспечить эффективное управление информацией, улучшить обслуживание клиентов и повысить общую эффективность бизнеса.

1. Постановка задачи

1.1 Аналитический обзор аналогов по теме

Одним из ключевых моментов в разработке программного обеспечения является просмотр и изучение различных аналогов, поиск в них недостатков и достоинств. Перед тем как приступить к работе необходимо провести анализ аналогов и прочитать соответствующие статьи по данной теме. В наши дни множество программ и сервисов по бронированию мест в гостиницах. В ходе поиска были найдены прототипы сайтов некоторых отелей.

1.1.1 Сервис Travel.Yandex

Travel.Yandex.ru представляет собой веб-платформу, предоставляющую широкий спектр услуг для поиска и бронирования гостиничных номеров, а также информации о гостиницах и местах проживания. Помимо бронирования номеров в гостиницах сервис предоставляет возможность онлайн заказа ж/д- и авиабилетов, а также туров.

Платформа обладает обширной базой данных гостиниц и предложений, что позволяет клиентам найти идеальное предложение, удовлетворяющее разным бюджетам и предпочтениям.

К особенностям данного сервиса можно отнести удобный поиск и сравнение гостиниц. Он включает в себя фильтрацию по цене, рейтингу, популярности, расположению, датам бронирования и количеству человек. Пользователи могут легко находить наилучшие варианты проживания, подходящие под их бюджет и предпочтения. Страница поиска, представленная на рисунке 1.1, так же содержит карту города с указанием стоимости проживания в гостиницах.

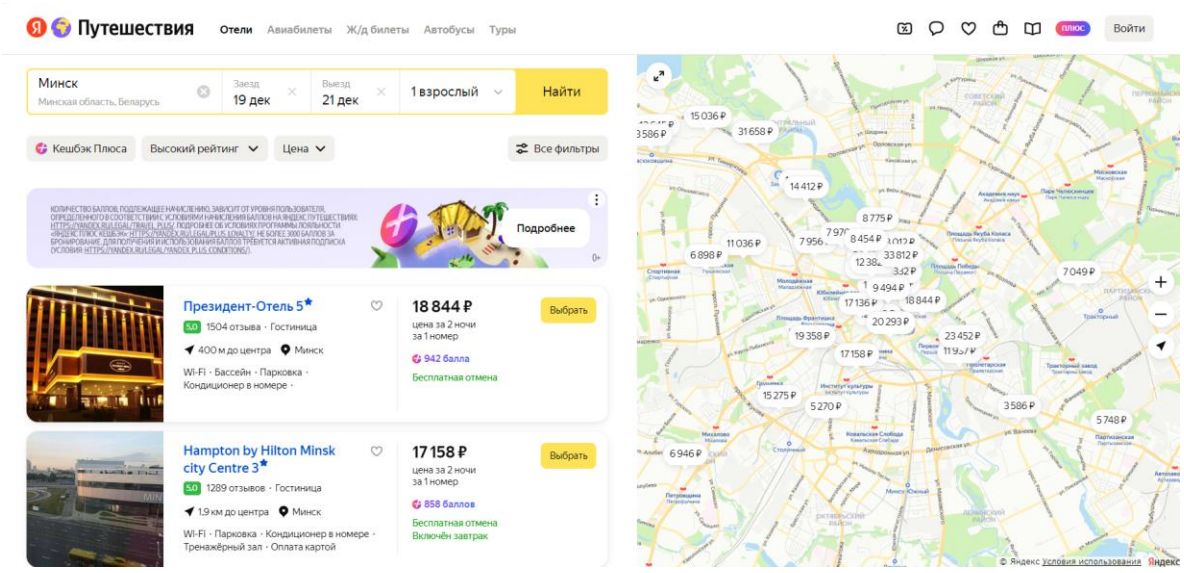


Рисунок 1.1 – Страница поиска

Также сервис предоставляет детальные описания гостиниц, включая адрес, фотографии, информацию о типах доступных номеров (рисунок 1.2). Это позволяет

клиентам более осознанно выбирать место проживания, учитывая предоставляемые гостиницей услуги и местоположение.

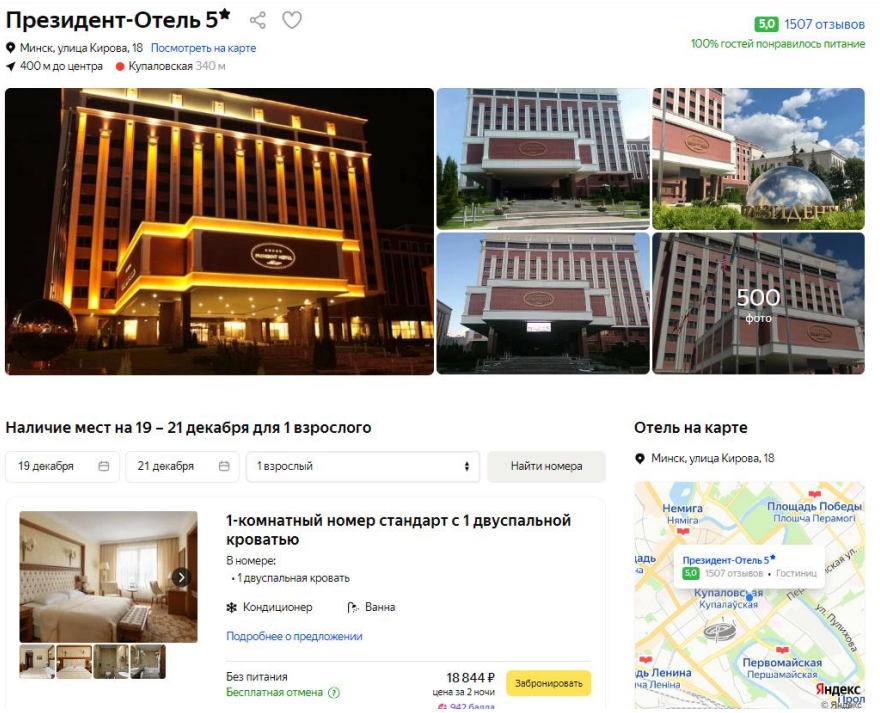


Рисунок 1.2 – Страница с детальной информацией об отеле

Travel.Yandex обеспечивает удобную систему онлайн-бронирования номеров, продемонстрированную на рисунке 1.3. Клиенты могут выбирать удобные даты и типы номеров, а также использовать разнообразные методы оплаты.

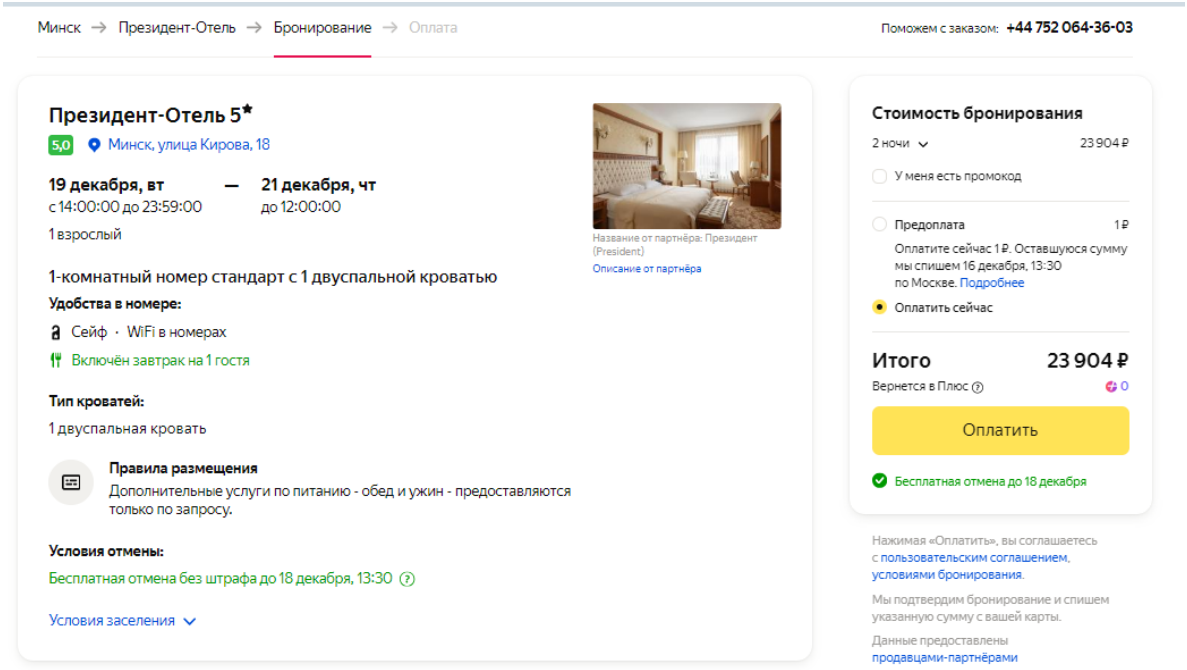


Рисунок 1.3 – Бронирование номера

При использовании данного сервиса для бронирования можно получить Кешбэк Плюса (баллы Яндекс).

Пользователи могут оставлять отзывы о гостиницах и присваивать им рейтинг, основанный на их опыте проживания. Это позволяет другим клиентам получить представление о качестве услуг и условиях в гостинице перед бронированием.

К недостаткам сервиса Travel.Yandex можно отнести то, что он не предоставляет услуги напрямую, а является посредником между клиентами и сторонними агентствами и гостиницами. Это может усложнить процесс бронирования и сопровождаться дополнительными сборами. Так же клиенты могут не узнать о каких-либо акциях и эксклюзивных предложениях конкретного отеля. Вторым недостатком можно назвать наличие рекламы, что может затруднить поиск.

В целом, Travel.Yandex.ru предоставляет обширные возможности для поиска и сравнения гостиничных вариантов, однако стоит учитывать его функцию посредника и оценивать дополнительные сборы и условия бронирования при выборе этого сервиса.

1.1.2 Сервис «HotelMinsk»

HotelMinsk.by представляет собой официальный веб-сайт одной конкретной гостиницы – "Гостиницы Минск". Это означает, что клиенты могут получить всю необходимую информацию о данной гостинице, включая описание номеров, предоставляемые услуги, цены, а также контактные данные для связи. Это полезно для клиентов, которые уже выбрали эту конкретную гостиницу для проживания. Главная страница сайта представлена на рисунке 1.4.

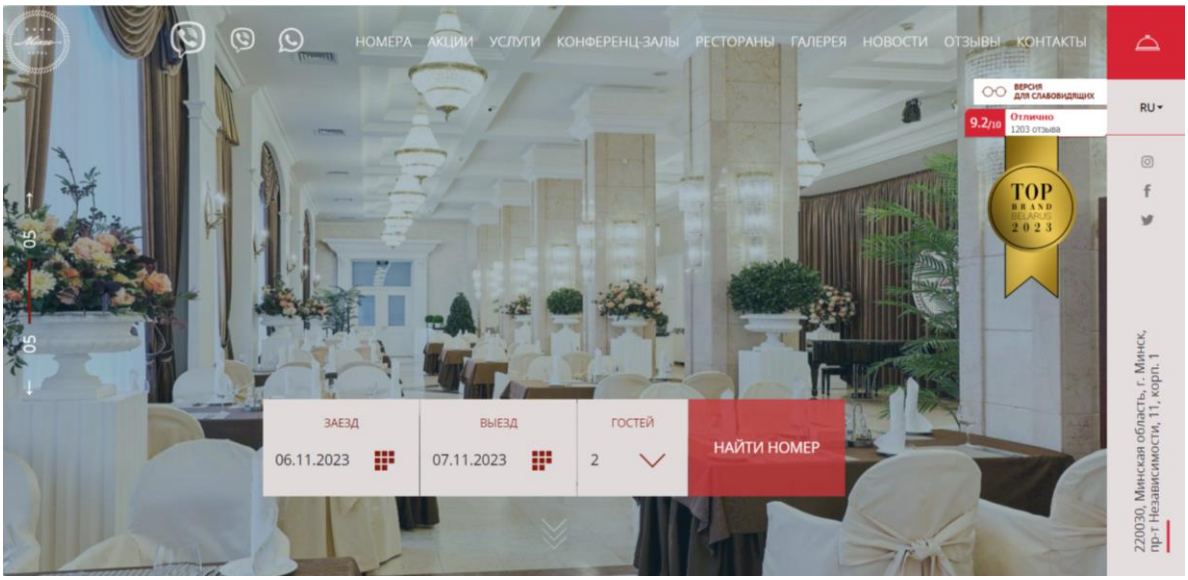


Рисунок 1.4 – Главная страница сайта HotelMinsk

Из преимуществ сайт предоставляет возможность онлайн-бронирования номеров в "Гостинице Минск" напрямую через официальный ресурс. Это удобно и обеспечивает надежность, так как клиенты имеют доступ к официальной информации и могут быть уверены в надежности бронирования. Официальный сайт

гостиницы может предоставлять эксклюзивные предложения и акции, которые недоступны на других платформах бронирования. Среди услуг, предоставляемых гостиницей, можно выделить охраняемую стоянку, аренду офисов, фитнес-центр, спа, салон красоты, казино, транспортное обслуживание и экскурсии.

При поиске комнаты можно также выбрать тариф проживания, просмотреть информацию о комнате и фото (рисунок 1.5).

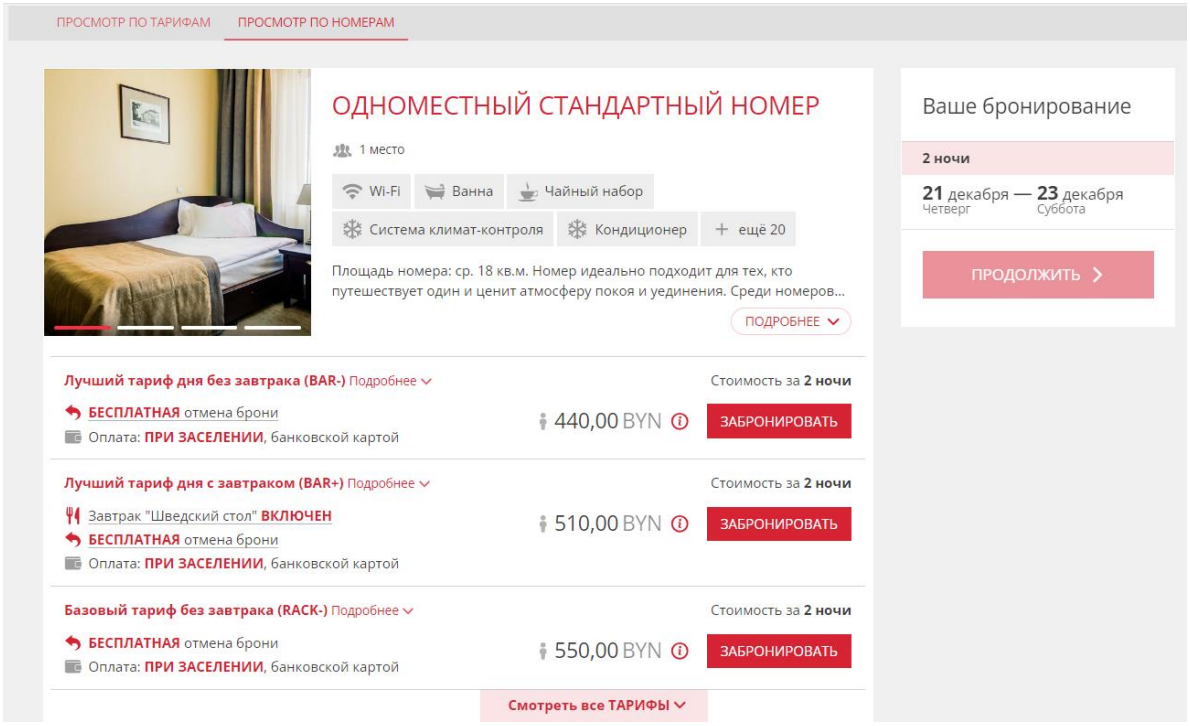


Рисунок 1.5 – Страница с выбором типа номера

К недостаткам HotelMinsk можно отнести отсутствие информации о правилах проживания, информации о заселении с детьми разных возрастов, а также животными.

В итоге, HotelMinsk.by, как официальный ресурс "Гостиницы Минск", обеспечивает удобство и доверие при бронировании номеров, а также предоставляет эксклюзивные условия для клиентов, которые выбирают данную гостиницу. Однако стоит помнить, что выбор ограничен только одним вариантом, и информация о других гостиницах ограничена на этом сайте.

1.1.3 Сервис «Beijinghotelminsk»

Представляет собой веб-сайт гостиницы «Пекин» и является прямым поставщиком услуг. Данный ресурс предоставляет детальное описание различных типов номеров, разделяя его на соответствующие категории (рисунок 1.6). Среди достоинств можно отметить наличие информации о правилах проживания, в которых оговорены важные моменты. Семьям с маленькими детьми могут предоставить всю необходимую мебель.

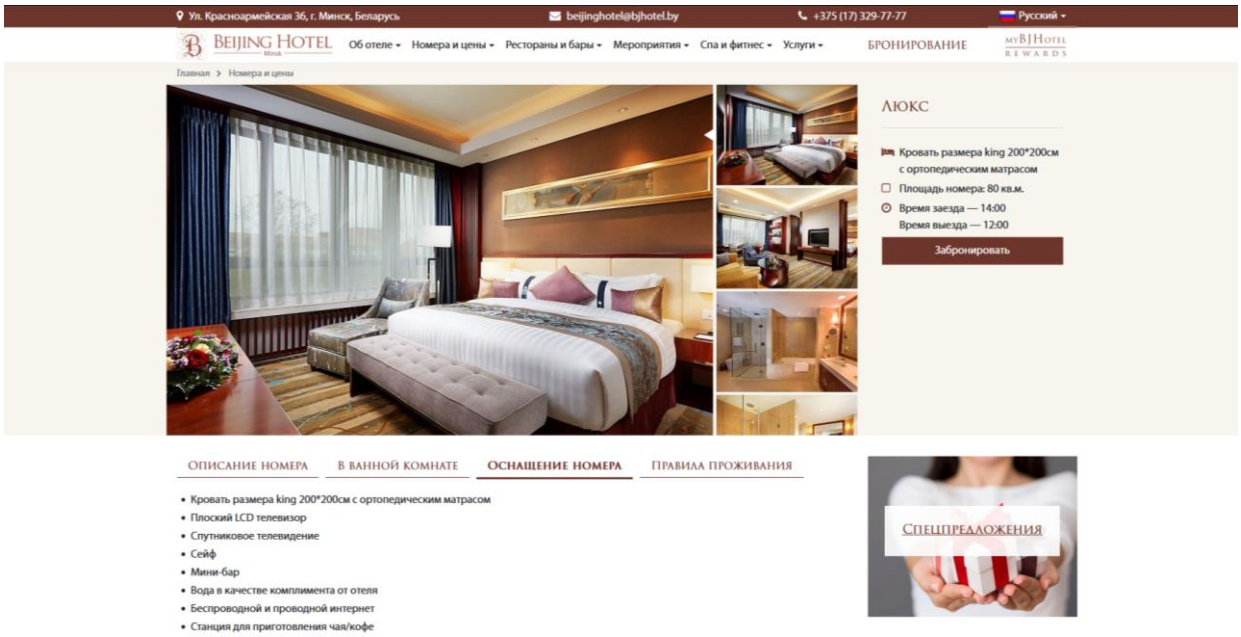


Рисунок 1.6– Описание номера

Также гостиница «Пекин» предоставлет широкий стект дополнительных услуг, таких как спортзал, спа, аренда машины, сейфа, офиса, доставка еды в номер и услуга «звонок-будильник».

Единственным недостатком можно назвать запрет проживания с домашними животными.

1.2 Определение основных требований к базе данных

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

1.2.1 Определение функциональных требований

Функциональные требования представляют собой конкретные описания функций и возможностей, которые база данных должна предоставлять для удовлетворения потребностей отеля и его клиентов. Были определены следующие требования:

– управление персоналом (добавление, удаление и изменение информации);

– управление номерами (добавление, удаление и изменение информации);

– хранение информации о сотрудниках, включая их контактные данные, должности, закрепленные сервисы;

– хранение данных о клиентах, включая их контактные данные, историю бронирований и дополнительные услуги;

– хранение информации о номерах, включая тип номера, его стоимость и описание;

– бронирование номеров;

– заказ сервисов;

– обеспечение защиты данных и доступа только авторизованных пользователей к базе данных;

Данные функциональные требования будут реализованы в курсовом проекте.

1.2.2 Определение нефункциональных требований

В качестве нефункциональных требований, можно выделить:

– безопасность: база данных должна обеспечивать безопасность данных и защиту от несанкционированного доступа, в том числе использование механизмов авторизации и аутентификации;

– производительность: база данных должна иметь высокую производительность и обеспечивать быстрый доступ к данным для обеспечения эффективной работы отеля;

– мониторинг: система должна поддерживать мониторинг состояния базы данных и предоставлять отчеты о производительности и использовании ресурсов;

– должен быть проведен импорт данных из XML файлов, экспорт данных в формат XML;

– удобство использования: база данных должна быть удобной и простой в использовании для обеспечения эффективной работы пользователей.

Эти нефункциональные требования важны для обеспечения эффективной работы базы данных отеля, ее надежности и удовлетворения потребностей клиентов.

1.3 Вывод

В данном разделе был проведен аналитический обзор аналогов, включающий три конкретных примера: «Travel.Yandex», «HotelMinsk» и «Beijinghotelminsk». Это позволило определить преимущества и недостатки существующих решений и использовать эту информацию при разработке нового проекта.

Сформулированная задача проекта ясно описывает его направление – реализация базы данных для управления информацией об отеле с использованием средств мониторинга состояния СУБД. Это означает стремление к созданию эффективной, масштабируемой и надежной системы, способной отслеживать и оптимизировать свою работу в реальном времени.

Также были определены функциональные и нефункциональные требования к базе данных и определены основные задачи курсового проекта. Функциональные требования четко определяют ожидаемый функционал системы, включая управление номерами, бронирование и другие аспекты гостиничного бизнеса. В то время как нефункциональные требования задают критерии производительности, безопасности и надежности.

2. Проектирование базы данных.

Для реализации задачи курсового проекта была проектирована база данных Hotel_db, состоящая из 10 таблиц: BOOKING, BOOKING_STATE, EMPLOYEES, GUESTS, PHOTO, ROOM_TYPES, ROOMS, SERVICES и SERVICE_TYPES, TARIFF_TYPES. Каждая таблица спроектирована с учетом конкретных потребностей отеля, а также для обеспечения оптимального управления информацией о бронированиях, персонале, гостях, номерах и услугах.

Диаграмма базы данных Hotel_db, спроектированной в ходе разработки представлена на рисунке 2.1.

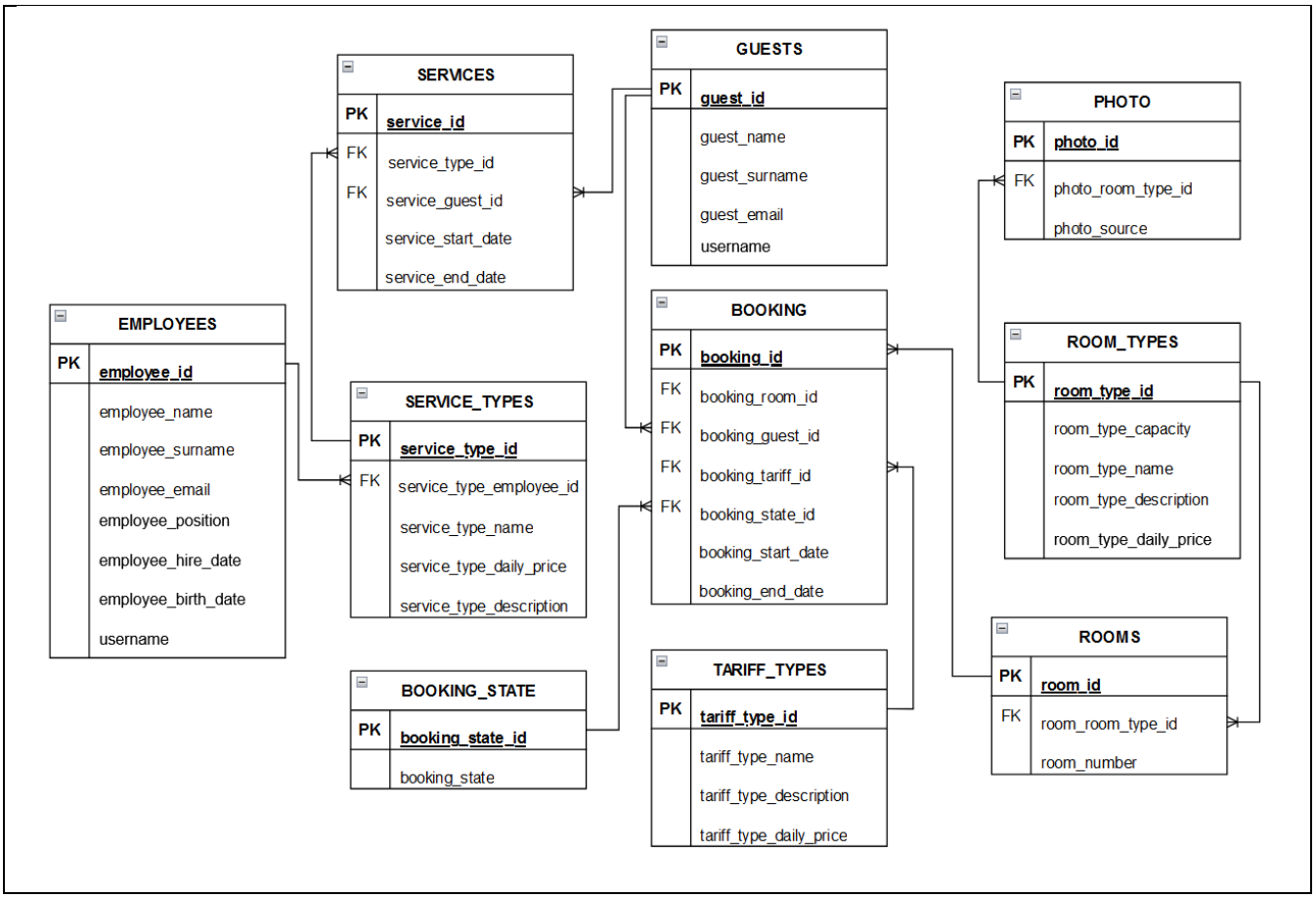


Рисунок 2.1 – Логическая структура базы данных

Диаграмма вариантов использования для пользователя администратора и гостя представлены в приложении А.

Таким образом была успешно разработана и проектирована база данных Hotel_db, представляющая собой слаженный набор из 10 таблиц. Логическая структура таблиц позволяет быстро и легко получать нужную информацию, а также расширять функциональность системы без пересмотра всей архитектуры, что позволит ускорить масштабирование системы.

3. Разработка объектов базы данных

3.1 Создание таблиц базы данных

При разработке приложения для курсового проекта была использована база данных Oracle.

Для реализации работы базы данных было создано 10 таблиц: BOOKING, BOOKING_STATE, EMPLOYEES, GUESTS, PHOTO, ROOM_TYPES, ROOMS, SERVICES, SERVICE_TYPES, TARIFF_TYPES. Каждая из созданных таблиц не только представляет собой отдельный функциональный блок для хранения информации, но и интегрируется в единое целое, что позволяет эффективно взаимодействовать между различными аспектами работы отеля.

Таблица GUESTS хранит информацию о постояльцах отеля. Состоит из следующих столбцов (таблица 3.1):

Таблица 3.1 – Столбцы таблицы GUESTS

Наименование	Описание	Тип
guest_id	Уникальный идентификатор гостя, первичный ключ	NUMBER(10)
guest_email	Электронная почта гостя	NVARCHAR2(50)
guest_name	Имя гостя	NVARCHAR2(50)
guest_surname	Фамилия гостя	NVARCHAR2(50)
username	Логин гостя	NVARCHAR2(50)

Таблица EMPLOYEES содержит информацию о работниках отеля. Состоит из столбцов (таблица 3.2):

Таблица 3.2 – Столбцы таблицы EMPLOYEES

Наименование	Описание	Тип
employee_id	Уникальный идентификатор сотрудника, первичный ключ	NUMBER(10)
employee_email	Электронная почта сотрудника	NVARCHAR2(50)
employee_name	Имя сотрудника	NVARCHAR2(50)
employee_surname	Фамилия сотрудника	NVARCHAR2(50)
employee_position	Должность	NVARCHAR2(50)
employee_hire_date	Дата найма	DATE
employee_birth_date	День рождения	DATE
username	Логин сотрудника	NVARCHAR2(50)

Таблица ROOM_TYPES содержит информацию о типах комнат в отеле. Состоит из столбцов (таблица 3.3):

Таблица 3.3 – Столбцы таблицы ROOM_TYPES

Наименование	Описание	Тип
room_type_id	Уникальный идентификатор типа комнаты, первичный ключ	NUMBER(10)
room_type_name	Название типа команты	NVARCHAR2(50)
room_type_capacity	Вместимость комнаты	NUMBER(10)
room_type_daily_price	Стоимость проживания за сутки	FLOAT(10)
room_type_description	Описание типа комнаты	NVARCHAR2(200)

Таблица ROOMS содержит информацию о комнатах отеля. Состоит из столбцов (таблица 3.4):

Таблица 3.4 – Столбцы таблицы ROOMS

Наименование	Описание	Тип
room_id	Уникальный идентификатор комнаты, первичный ключ	NUMBER(10)
room_room_type_id	ID типа комнаты, внешний ключ	NUMBER(10)
room_number	Номер комнаты	NVARCHAR2(50)

Таблица PHOTO содержит информацию о фотографиях комнат отеля. Состоит из столбцов, представленных в таблице 3.5. Использование типа данных BLOB для столбца photo_data обеспечивает эффективное хранение изображений в базе данных. Этот подход позволяет управлять фотографиями комнат, ассоциированными с конкретными номерами, предоставляя возможность динамического обновления и визуализации данных в приложении отеля.

Таблица 3.5 – Столбцы таблицы PHOTO

Наименование	Описание	Тип
photo_id	Уникальный идентификатор фото, первичный ключ	NUMBER(10)
photo_room_type_id	ID типа комнаты, внешний ключ	NUMBER(10)
photo_source	Фото	BLOB

Таблица SERVICE_TYPES содержит информацию о типах услуг отеля. Состоит из столбцов (таблица 3.6):

Таблица 3.6 – Столбцы таблицы SERVICE_TYPES

Наименование	Описание	Тип
service_type_id	Уникальный идентификатор типа комнаты, первичный ключ	NUMBER(10)
service_type_name	Название типа комнаты	NVARCHAR2(50)
service_type_description	Описание	NVARCHAR2(200)
service_type_daily_price	Стоимость	FLOAT(10)
service_type_employee_id	ID сотрудника, внешний ключ	NUMBER(10)

Таблица SERVICES содержит информацию об услугах отеля. Состоит из столбцов (таблица 3.7):

Таблица 3.7 – Столбцы таблицы SERVICES

Наименование	Описание	Тип
service_id	Уникальный идентификатор сервиса, первичный ключ	NUMBER(10)
service_type_id	ID типа сервиса, внешний ключ	NUMBER(10)
service_guest_id	ID гостя, внешний ключ	NUMBER(10)
service_start_date	Дата начала	DATE
service_end_date	Дата окончания	DATE

Таблица TARIFF_TYPES предоставляет основные сведения о различных тарифах, предлагаемых отелем, включая их стоимость, описание. Состоит из столбцов (таблица 3.8):

Таблица 3.8 – Столбцы таблицы TARIFF_TYPES

Наименование	Описание	Тип
tariff_type_id	Уникальный идентификатор тарифа, первичный ключ	NUMBER(10)
tariff_type_name	Название тарифа	NVARCHAR2(50)
tariff_type_description	Описание тарифа	NVARCHAR2(200)
tariff_type_daily_price	Стоимость тарифа за сутки	FLOAT(10)

Таблица BOOKING_STATE содержит информацию о статусах бронирования в отеле. Эта таблица позволяет отслеживать и управлять жизненным циклом бронирования, предоставляя информацию о его текущем состоянии. Состоит из столбцов (таблица 3.9):

Таблица 3.9 – Столбцы таблицы BOOKING_STATE

Наименование	Описание	Тип
booking_state_id	Уникальный идентификатор статуса брони, первичный ключ	NUMBER(10)
booking_state	Описание статуса брони	NVARCHAR2(100)

Таблица BOOKING содержит информацию о бронированиях номеров в отеле. Состоит из столбцов (таблица 3.10):

Таблица 3.10 – Столбцы таблицы BOOKING

Наименование	Описание	Тип
booking_id	Уникальный идентификатор брони, первичный ключ	NUMBER(10)
booking_room_id	ID комнаты, внешний ключ	NUMBER(10)
booking_guest_id	ID гостя, внешний ключ	NUMBER(10)
booking_tariff_id	ID тарифа, внешний ключ	NUMBER(10)
booking_state_ id	ID состояние брони, внешний ключ	NUMBER(1)
booking_start_date	Дата начала брони	DATE
booking_end_date	День окончания брони	DATE

Разработанная база данных, названная "Hotel_db", включает в себя 10 таблиц. Каждая из этих таблиц была создана с учетом конкретных потребностей отеля, что обеспечивает оптимальное управление информацией о бронированиях, персонале, гостях, номерах, услугах и тарифах.

3.2 Индексы

Индекс — объект базы данных, создаваемый с целью повышения производительности поиска данных.

Таблицы в базе данных могут иметь большое количество строк, которые хранятся в произвольном порядке, и их поиск по заданному критерию путём последовательного просмотра таблицы строка за строкой может занимать много времени. Индексы в базе данных используются для ускорения выполнения запросов, особенно при поиске, сортировке и объединении данных. Индексы обеспечивают эффективный доступ к данным, уменьшая время выполнения запросов за счет организации структуры данных в определенном порядке.

Для таблицы BOOKING были созданы индексы на столбце booking_room_id, а также на столбцах booking_start_date и booking_end_date, так как эти столбцы используются в условиях WHERE для процедур и представлений связанных с поиском свободных и занятых комнат отеля. Так же эти столбцы используются в проверке наличия брони на выбранные даты при заказе услуг. Создание индекса представлено в листинге 3.1.

```
CREATE INDEX booking_room_index ON BOOKING(booking_room_id);
CREATE INDEX booking_dates_index ON BOOKING(booking_start_date,
booking_end_date);
```

Листинг 3.1 – Индексы для таблицы BOOKING

Также были созданы необходимые индексы для других таблиц в базе данных, учитывая их структуру и особенности использования в запросах.

Правильное использование индексов в базе данных существенно повышает производительность системы, особенно при выполнении сложных запросов и операций сортировки. Важно соблюдать баланс между количеством индексов и операциями вставки/обновления данных, так как индексы требуют дополнительных ресурсов при выполнении этих операций.

3.3 Представления

Для базы данных отеля было создано несколько представлений, которые предоставляют удобный и эффективный способ получения информации из базы данных.

Представление SERVICE_TYPE_VIEW создано на основе данных из таблиц SERVICES, SERVICE_TYPES и EMPLOYEES. Это представление обеспечивает удобный и структурированный доступ к информации о типах услуг в отеле, включая идентификатор услуги, тип услуги, её описание. Создание представления продемонстрировано в листинге 3.2.

```
CREATE or replace VIEW SERVICE_TYPE_VIEW AS
SELECT
    S.service_id,
    ST.service_type_id,
    ST.service_type_name,
    ST.service_type_description,
    ST.service_type_daily_price,
    E.employee_name,
    E.employee_surname
FROM
    SERVICES S
    JOIN SERVICE_TYPES ST ON S.service_type_id = ST.service_type_id
    JOIN EMPLOYEES E ON ST.service_type_employee_id = E.employee_id;
```

Листинг 3.2 – Представление SERVICE_TYPE_VIEW

В дополнение к SERVICE_TYPE_VIEW, в базе данных также реализованы другие представления. Для отображения заказанных гостями услуг используется представление SERVICE_VIEW, для отображения информации о бронировании номеров – BOOKING_DETAILS_VIEW, для получения фотографий комнат – GET_ROOM_PHOTO, для вывода информации о комнатах и их типах используется представление ROOM_INFO_VIEW.

Все эти представления играют важную роль в обеспечении эффективного и удобного взаимодействия с базой данных, упрощая запросы и предоставляя пользователю необходимую информацию в более удобной форме.

3.4 Процедуры и функции

Процедура в Oracle – это объект базы данных, который представляет собой набор SQL-инструкций, которые могут быть вызваны для выполнения определенной задачи.

Администратор отеля обладает расширенными правами и имеет доступ к выполнению CRUD-операций (Create, Read, Update, Delete) во все таблицы базы данных. Для обеспечения удобства и безопасности управления данными, были созданы соответствующие процедуры. Пример процедуры создания гостя представлен в приложении Б.

Для сотрудников доступны следующие процедуры:

– GET_BOOKING_DETAILS_BY_ID возвращает информацию о бронировании по указанному идентификатору бронирования;

– GET_SERVICE_INFO возвращает информацию о сервисах с указанным идентификатором; если параметр не указан, выводит информацию о всех сервисах;

– BIRTHDAY_REPORT формирует отчет о днях рождениях сотрудников;

– GET_MY_SERVICES возвращает информацию о сервисах, связанных с текущим пользователем или сотрудником;

– FIND_GUEST выводит информацию о госте по указанному идентификатору гостя;

– QUIT_JOB удаляет текущего сотрудника.

Для гостей отеля доступны следующие процедуры:

– GET_AVAILABLE_ROOMS получает список доступных номеров в отеле с заданной вместимостью и периодом пребывания;

– BOOKING_NOW бронирует номер в отеле на заданный период и с указанным тарифом;

– PRE_BOOKING создает предварительное бронирование номера в отеле на указанный период и с заданным тарифом;

– GET_BOOKINGDETAILS_BY_ID получает подробную информацию о бронировании по его идентификатору;

– EDIT_BOOKING редактирует информацию о существующем бронировании, включая номер, даты пребывания и тариф;

– DENY_BOOKING отменяет бронирование с указанным идентификатором;

– RESTORE_BOOKING восстанавливает отмененное бронирование с указанным идентификатором;

– ORDER_SERVICE заказывает услугу для существующего бронирования с указанным типом услуги и периодом предоставления услуги;

– EDIT_SERVICE редактирует информацию о существующей услуге, включая тип услуги и период предоставления услуги;

– GET_SERVICE_INFO получает информацию о типе услуги по её идентификатору;

– GET_TARIFF_INFO получает информацию о тарифе по его идентификатору;

– GET_ROOM_INFO получает информацию о номере по его идентификатору;

– CHECK_OUT выселяет гостя, завершая указанное бронирование.

– GET_MY_SERVICES получает список услуг, заказанных пользователем.

– GET_MY_BOOKINGS получает список бронирований, совершенных пользователем.

Для расчета стоимости проживания была разработана функция CALCULATE_STAY_COST, принимающая в качестве параметра ID брони и возвращающая итоговую стоимость.

3.5 Пакеты

В рамках базы данных были созданы три пакета: HotelAdminPack, UserPack и EmployeePack, предназначенные для организации функционала и процедур, соответствующих ролям в системе. Пакеты используются для группировки логически связанных элементов, предоставляя модульность, сокрытие деталей реализации и повторное использование кода.

Пакет HotelAdminPack содержит процедуры и функции, предназначенные для выполнения административных задач и CRUD-операций с таблицами. Пакет UserPack содержит функции и процедуры, предоставляющие функционал для гостей отеля. Спецификация пакета для гостей отеля представлена в приложении В. Пакет EmployeePack содержит функционал и процедуры, предназначенные для использования сотрудниками отеля.

Таким образом использование пакетов в базе данных обеспечивает высокую степень инкапсуляции кода, позволяя объединить процедуры, функции и типы данных в одной единице.

3.6 Триггеры

Триггеры обеспечивают автоматизацию действий и реакцию на изменения данных, предоставляя возможность выполнения дополнительных операций перед или после события.

В разрабатываемой базе данных был создан триггер UPDATE_GUEST_XML_TRIGGER, который срабатывает после вставки, удаления или обновления записей в таблице GUESTS. Этот триггер используется для автоматического обновления данных о гостях с помощью экспорта информации о гостях в файл. Реализация представлена в листинге 3.3.

```
create or replace trigger UPDATE_GUEST_XML_TRIGGER
    after insert or delete or update
    on GUESTS
begin
    EXPORT_TO_FILE('select * from Guests', 'Guests');
    DBMS_OUTPUT.PUT_LINE('Данные о гостях успешно обновлены');
EXCEPTION
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('Произошла ошибка при экспорте гостей: ' ||
SQLERRM);
end;
```

Листинг 3.3 – Создание триггера UPDATE_GUEST_XML_TRIGGER

Данный триггер предоставляет эффективное средство для обновления информации о гостях с использованием экспорта в файл, что повышает надежность и актуальность данных в системе.

3.7 Синонимы

В базе данных были созданы синонимы для обеспечения удобного и безопасного доступа к пакетам функций и процедур для различных ролей. Применение синонимов упрощает синтаксис запросов и скрывает детали реализации, делая структуру базы данных более гибкой и поддерживаемой. Создание синонимов представлено на листинге 3.4.

```
CREATE public SYNONYM HOTEL_ADMIN FOR ADMIN.HOTELADMINPACK;
CREATE public SYNONYM GUEST FOR ADMIN.USERPACK;
CREATE public SYNONYM EMPLOYEE FOR ADMIN.EMPLOYEEPACK;
```

Листинг 3.4 – Создание синонимов

Создание синонимов для пакетов помогает упростить работу с базой данных для администраторов, гостей и сотрудников.

3.8 Роли и пользователи

Роль представляет собой набор привилегий и прав доступа, который может быть присвоен одному или нескольким пользователям. В базе данных было предусмотрено разделение пользователей на три основные роли: «Hotel_admin_role», «Employee_role» и «Guest_role». Пример создания роли для гостя и выдачи необходимых привилегий приведен в листинге 3.5.

```
CREATE ROLE Guest_role;
GRANT CREATE SESSION TO Guest_role ;
GRANT EXECUTE ON ADMIN.UserPack TO Guest_role;
GRANT READ ON DIRECTORY MEDIA_DIR TO Guest_role;
```

Листинг 3.5 – Создание роли Guest_role и выдача ей привилегий

Профили безопасности используются для управления ресурсами и ограничения использования системных ресурсов для каждого пользователя. Пример создания профиля безопасности для пользователей приведен в листинге 3.6.

```
CREATE PROFILE PF_USER LIMIT
    PASSWORD_LIFE_TIME 180
    SESSIONS_PER_USER 3
    FAILED_LOGIN_ATTEMPTS 7
    PASSWORD_LOCK_TIME 1
    PASSWORD_REUSE_TIME 5
    CONNECT_TIME 180
    IDLE_TIME 45;
```

Листинг 3.6 – Создание профиля безопасности

Создание пользователей и присвоение им ролей осуществляется с помощью соответствующих процедур. Фрагмент такой процедуры представлен в листинге 3.7.

```
EXECUTE IMMEDIATE 'CREATE USER ' || p_username ||
                      ' IDENTIFIED BY ' || p_username ||
                      ' DEFAULT TABLESPACE HOTEL_TS' ||
                      ' TEMPORARY TABLESPACE HOTEL_TEMP_TS'||
                      ' PROFILE PF_USER' ||
                      ' ACCOUNT UNLOCK'||
                      ' PASSWORD EXPIRE';
EXECUTE IMMEDIATE 'GRANT Guest_role TO ' || p_username;
```

Листинг 3.7 – Создание пользователя

При создании пользователя устанавливается пароль, совпадающий с именем пользователя. С помощью PASSWORD EXPIRE активируется механизм смены пароля при первой аутентификации пользователя. Этот процедурный подход обеспечивает гибкость в управлении пользователями и ролями, а также поддерживает базовые меры безопасности, такие как смена пароля при первой аутентификации.

3.9 Планировщик

Планировщик используется для автоматизации ряда задач, таких как ежедневные обслуживание, регулярные проверки данных, сбор статистики и т.д. В базе данных отеля планировщик используется для автоматизации процесса выселения гостей. Пример создания задачи представлен в листинге 3.8.

```
begin
dbms_scheduler.create_schedule(
  schedule_name => 'DAILY_CHECKOUT_SCHEDULE',
  start_date => SYSTIMESTAMP,
    end_date => NULL,
  repeat_interval => 'FREQ=DAILY; BYHOUR=12; BYMINUTE=0; BYSECOND=0',
  comments => 'DAILY_CHECKOUT_SCHEDULE starts now');
end;
begin
dbms_scheduler.create_program(
  program_name => 'DAILY_CHECKOUT_PROGRAM',
  program_type => 'STORED_PROCEDURE',
  program_action => 'ADMIN.HOTELADMINPACK.CHECK_OUT_GUESTS',
  number_of_arguments => 0,
  enabled => true,
  comments => 'DAILY_CHECKOUT_PROGRAM');
end;
begin
    dbms_scheduler.create_job(
            job_name => 'DAILY_CHECKOUT_JOB',
            program_name => 'DAILY_CHECKOUT_PROGRAM',
            schedule_name => 'DAILY_CHECKOUT_SCHEDULE',
            enabled => true);
end;
```

Листинг 3.8 – Создание планировщика

Также были созданы задачи для автоматизации процесса регистраци гостей и удаления отмененных броней. Такой подход улучшает обслуживание и обеспечивает более эффективное управление данными в отеле, повышая общую производительность системы.

3.10 Вывод

В этом разделе было описано создание основных объектов базы данных, предназначенные для обеспечения её функциональности, производительности и безопасности.

Были разработаны и созданы таблицы базы данных для хранения информации о гостях, услугах, типах услуг, сотрудниках и других важных сущностях. Определены первичные и внешние ключи для обеспечения целостности данных и связей между таблицами. Для улучшения производительности запросов к базе данных были созданы различные индексы. Разработаны представления, облегчающие выполнение сложных запросов и предоставляющие удобный интерфейс для получения нужной информации. Разработаны пакеты, включающие в себя набор процедур и функций, предназначенных для конкретных пользователей.

Использование синонимов к объектам базы данных, что упрощает использование и обеспечивает удобство при работе с базой данных. Определены роли и пользователи для обеспечения безопасности базы данных. Роли разграничивают доступ к различным частям данных и функциональности, а пользователи получают определенные привилегии в соответствии с их ролью.

В итоге, создание объектов базы данных в рамках данного раздела позволило структурировать и организовать базу данных для эффективного хранения, обработки и использования информации в рамках разрабатываемого проекта. Созданные объекты обеспечивают необходимую функциональность, производительность и безопасность для удовлетворения потребностей бизнес-логики системы.

4. Описание процедур импорта и экспорта

В данном курсовом проекте реализованы процедуры экспорта и импорта данных из XML файла в базу данных таблиц GUESTS и EMPLOYи наоборот.

4.1 Процедура экспорта данных

Один из способов экспорта таблиц методом работы с типом данных CLOB, который и будет формировать большой файл XML с данными из таблицы. Процедура EXPORT_TO_FILE представлена в приложении Г.

Процедура принимает два параметра p_query и p_filename. P_query принимает строку выполняемого запроса. Таким образом мы можем не просто экспортировать только таблицу, но и некоторые выборки данных, которые можно использовать для анализа. Параметр p_filename создаст файл с именем переданной строки, в которой будет хранится результат экспорта.

4.2 Процедура импорта данных

Для реализации импорта были созданы две процедуры: FILE_TO_CLOB и IMPORT_GUESTS_XML, представленные в приложении Г.

Процедура FILE_TO_CLOB отвечает за преобразование содержимого XML-файла в формат CLOB, который затем используется для обработки данных. Эта процедура читает файл строка за строкой и объединяет его содержимое в CLOB.

Процедура IMPORT_GUESTS_XML использует FILE_TO_CLOB для преобразования XML-файла в CLOB и далее в XMLTYPE. Затем она извлекает необходимые данные из XML и вставляет их в таблицу GUESTS_XML. Процедура обеспечивает корректное чтение и обработку данных, а также обрабатывает возможные ошибки в процессе выполнения.

4.3 Вывод

В данном разделе курсового проекта мы рассмотрели один из способов реализации процедур импорта и экспорта данных, а также использование формата XML для обмена данными между различными базами данных и приложениями. Процедура экспорта данных, которая представлена в приложении Г, работает с типом данных CLOB и позволяет экспортировать данные из таблицы или выполнить выборку данных для анализа. Ее преимуществом является возможность работы с большими объемами данных, которые можно сохранить в одном файле.

5. Тестирование производительности

Для проверки производительности базы данных необходимо заполнить ее большим количеством различных данных и узнать время выполнения одного запроса. Для данной задачи была создана соответствующая процедура, представленная в приложении Д.

```
ADMIN> Call INSERT_SERVICE_TYPES()
[2023-12-17 17:00:00] completed in 5 s 731 ms
```

Рисунок 5.1 –Заполнение таблицы большим количеством данных

Для анализа эффективности работы запросов к таблице с большим количеством строк использовался план запросов в SQL Developer. План запроса до создания индексов к таблице SERVICE_TYPES представлен на рисунке 5.2.

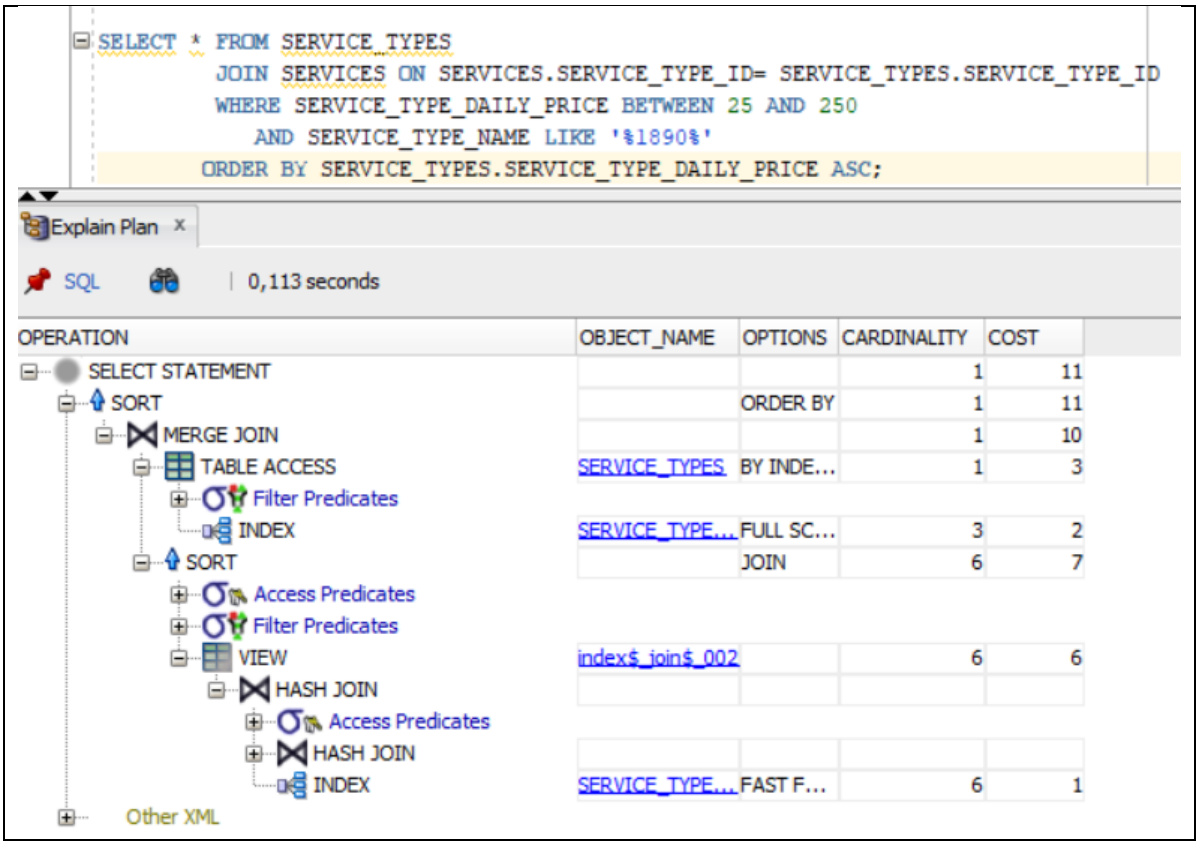


Рисунок 5.2 –Заполнение таблицы большим количеством данных

Из анализа видно, что запрос выполняется с использованием полного сканирования таблицы, что может сказаться на производительности при большом объеме данных.

После создания индексов для таблицы SERVICE_TYPES, время выполнения запроса значительно сократилось, что можно заметить на рисунке 5.3.

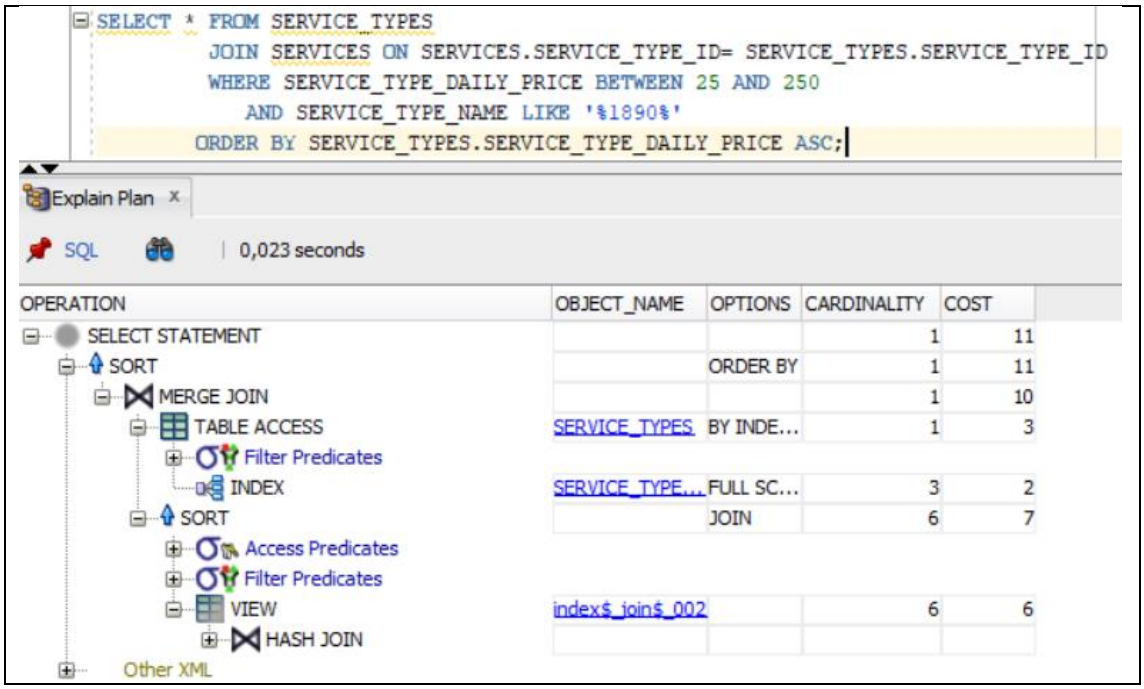


Рисунок 5.3 –Заполнение таблицы большим количеством данных

Проведенный анализ производительности базы данных с таблицей SERVICE_TYPES, содержащей большое количество строк, позволяет сделать следующие выводы. Первоначально, до создания индексов, запрос к таблице осуществлялся полным сканированием, что может привести к замедлению при работе с большим объемом данных.

Создание индексов на соответствующих столбцах таблицы существенно улучшило время выполнения запроса. Индексы позволяют эффективнее организовывать доступ к данным, снижая стоимость выполнения операций. Таким образом, база данных оказывается более подготовленной к обработке больших объемов данных, что является важным аспектом в условиях активной работы системы с множеством запросов и операций.

6. Описание технологии и ее применения в базе данных

6.1 Средства мониторинга состояния СУБД

Мониторинг состояния СУБД является важным инструментом для обеспечения высокой производительности и надежности базы данных. Он позволяет анализировать работу базы данных в реальном времени и выявлять проблемы, которые могут влиять на ее работу. Мониторинг включает в себя отслеживание различных метрик, таких как использование ресурсов, активность пользователей, сетевые запросы и другие ключевые показатели, предоставляя администраторам баз данных ценную информацию для оперативного реагирования на возможные угрозы или сбои.

В данном проекте используется Oracle Enterprise Manager Database Express (EM Express). Он представляет собой встроенный веб-интерфейс для администрирования базы данных Oracle и доступен из любого браузера по ссылке, настраиваемой администратором, что значительно упрощает доступ для мониторинга состояния базы данных.

Oracle Enterprise Manager Database Express 21 предлагает множество возможностей для работы с базами данных.

1. Администрирование базы данных: Oracle Enterprise Manager Database Express позволяет выполнять административные задачи, такие как управление безопасностью пользователей и управление памятью и хранилищем базы данных.
2. Управление пространством: Инструменты для управления табличными пространствами базы данных, а также просмотр статистики по использованию табличных пространств и информации об их физическом расположении в памяти компьютера.
3. Управление производительностью: Oracle Enterprise Manager Database Express значительно упрощает диагностику производительности базы данных, объединяя соответствующие экраны производительности базы данных в единый вид, называемый Performance Hub.
4. Мониторинг: Мониторинг SQL в реальном времени и исторический, а также мониторинг операций базы данных.
5. Мониторинг операций базы данных: Мониторинг операций базы данных позволяет отслеживать ресурсоёмкие запросы базы данных и отображает подробную информацию об используемых ресурсах.
6. Active Session History (ASH) Analytics: Аналитика истории активных сеансов.

Когда пользователи впервые входят в EM Express, они видят домашняя страница, на которой отображается информация о базу данных, а также данные о производительности, такие как использование ресурсов, среднее количество активных сеансов и отслеживаемые операторы SQL. Домашняя страница изображена на рисунке 6.1.

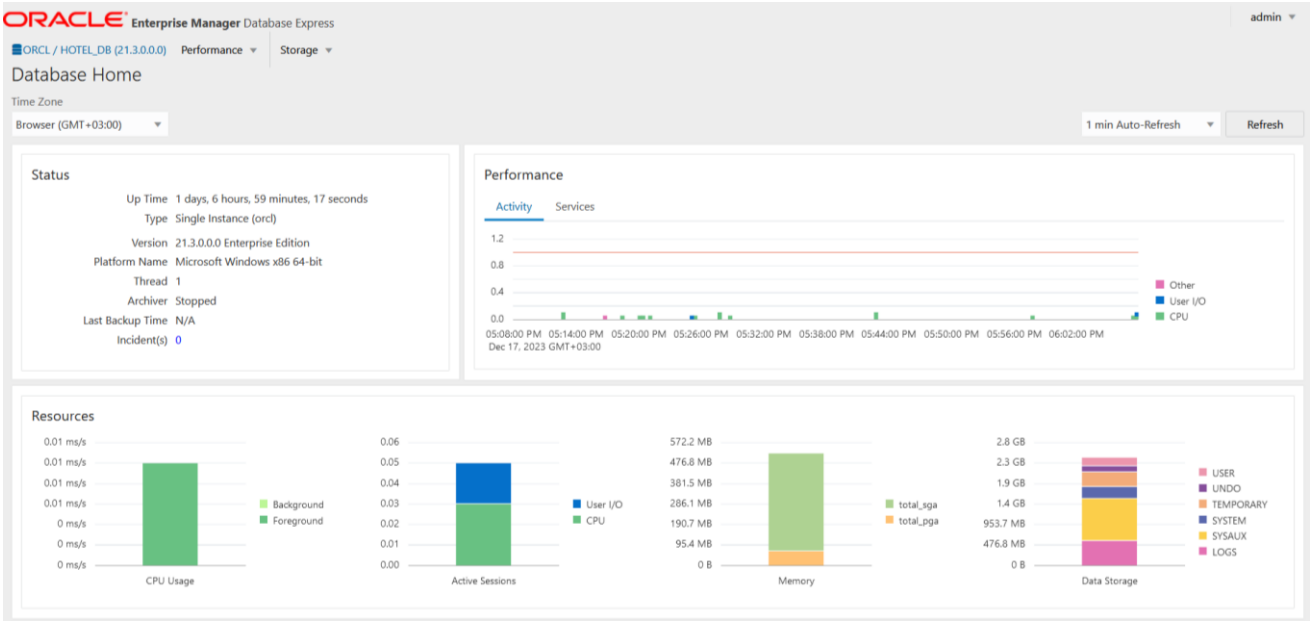


Рисунок 6.1 – Oracle Enterprise Manager

В разделе «Status» находится информация о текущем состоянии базы данных, версии EM Express, имени устройства, с которого осуществляется мониторинг, статус архивирования для базы данный, информация о бэкапах и инцидентах.

В разделе «Perfomance» отображается информация о классах ожидания, таких как пользовательский ввод/вывод, параллелизм, процессор и другие, и их активности в определенное время.

Ниже в разделе «Resourses» находится информация об используемых ресурсах процессора, активных сессий, памяти и хранилища данных.

На вкладке «Performance Hub» в разделе «Workload» можно просмотреть статистику использования процессора, время ожидания, активность пользователей и количество сессий за определённый промежуток времени

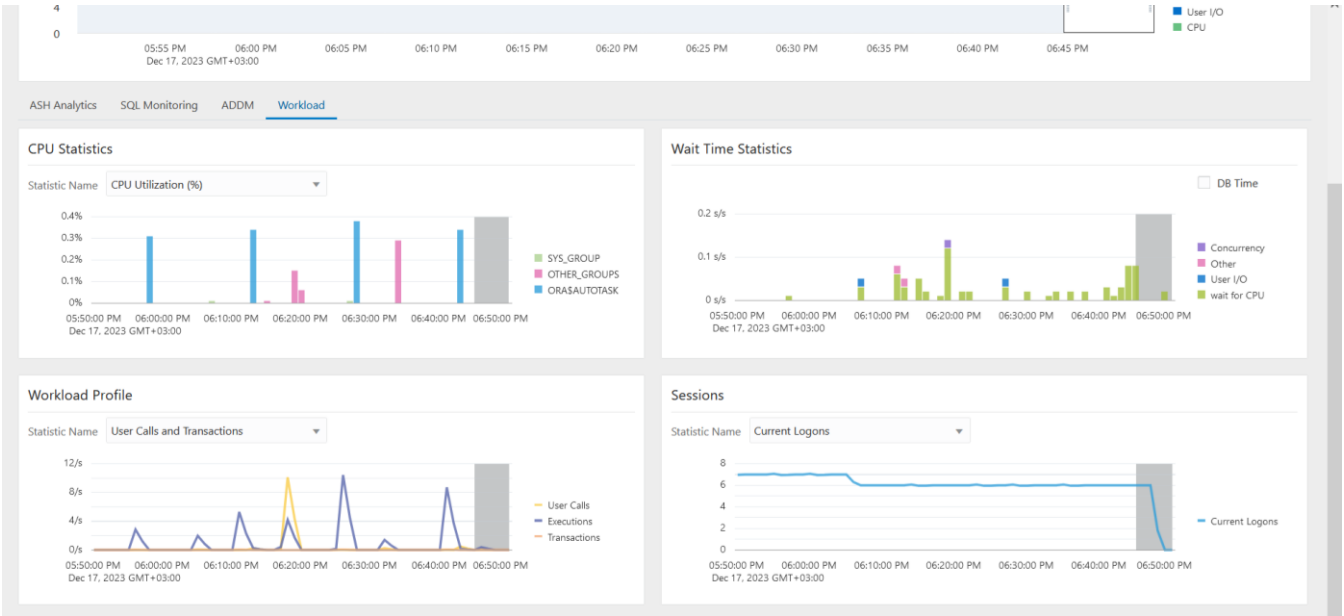


Рисунок 6.2 – Performance Hub

Раздел «SQL Monitor», представленный на рисунке 6.3, в Oracle Enterprise Manager Database Express предназначен для мониторинга и анализа выполнения SQL-запросов в реальном времени. С его помощью можно просматривать детальную информацию о планах выполнения, статистике, ожиданиях и параллелизме SQL-запросов, которые запущены в данный момент или были запущены в прошлом.

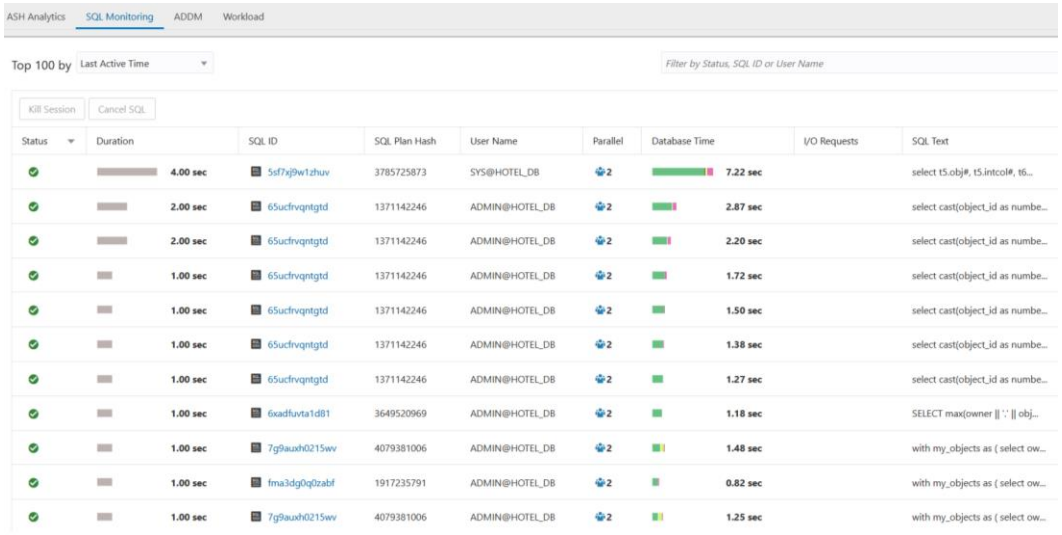


Рисунок 6.3 – SQL Monitor

SQL Monitor также позволяет сравнивать планы выполнения разных SQL-запросов и определять причины их различной производительности. Детальная информация о том, каким образом выполняется запрос, включая порядок выполнения операций, стоимость операций и использование индексов доступна по нажатия на интересующий запрос и представлена на рисунке 6.4

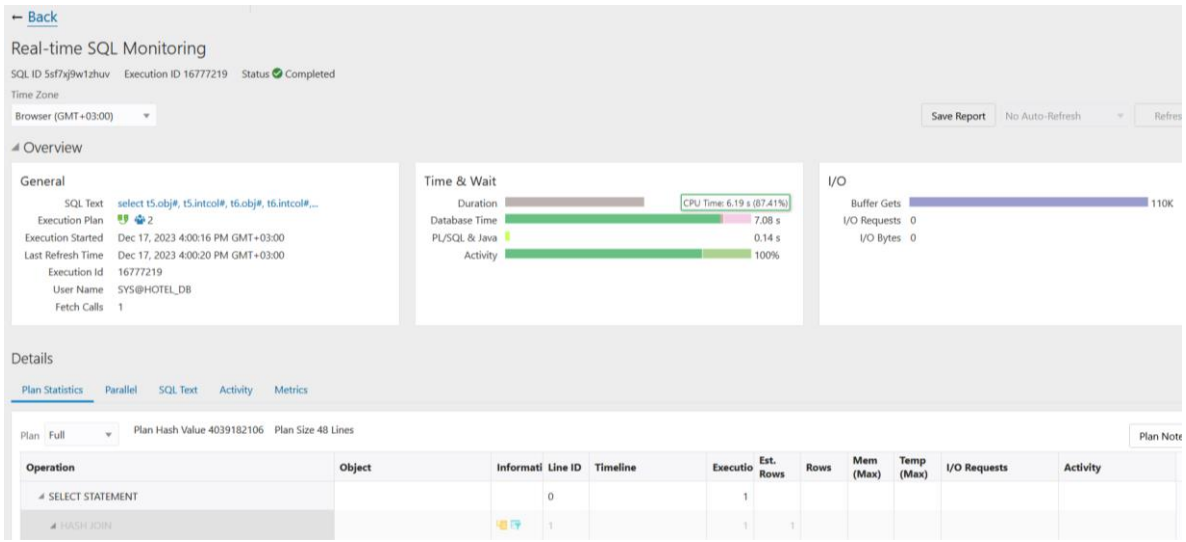
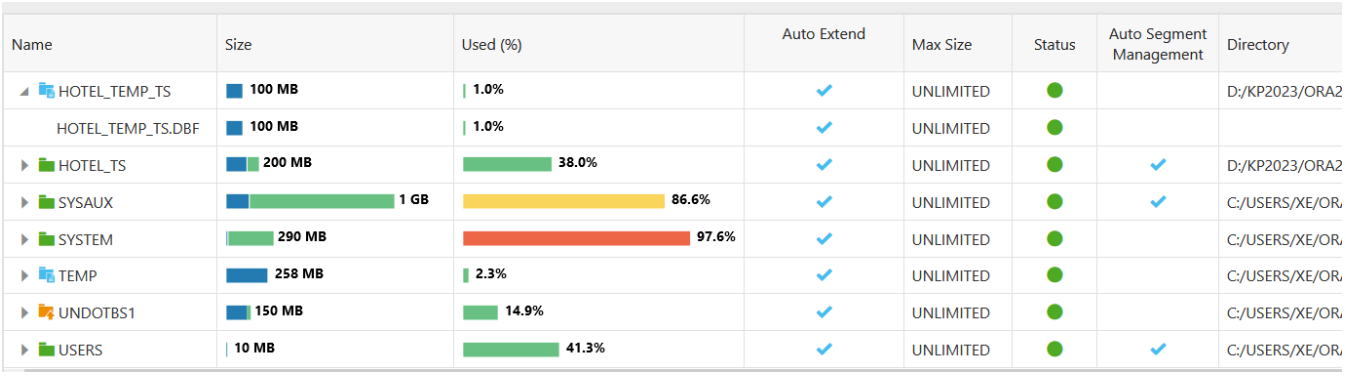


Рисунок 6.4 – SQL Monitor Details

Также в Oracle Enterprise Manager Database Express позволяет просмотреть информацию о табличных пространствах используемых в базе данных. На

рисунке 6.5 можно увидеть список табличных пространств, их размер, процент использования, статус, максимальный размер и путь к физическому файлу.



Name	Size	Used (%)	Auto Extend	Max Size	Status	Auto Segment Management	Directory
HOTEL_TEMP_TS	100 MB	1.0%	✓	UNLIMITED	●		D:/KP2023/ORA2
HOTEL_TEMP_TS.DBF	100 MB	1.0%	✓	UNLIMITED	●		
HOTEL_TS	200 MB	38.0%	✓	UNLIMITED	●	✓	D:/KP2023/ORA2
SYSAUX	1 GB	86.6%	✓	UNLIMITED	●	✓	C:/USERS/XE/ORA
SYSTEM	290 MB	97.6%	✓	UNLIMITED	●		C:/USERS/XE/ORA
TEMP	258 MB	2.3%	✓	UNLIMITED	●		C:/USERS/XE/ORA
UNDOTBS1	150 MB	14.9%	✓	UNLIMITED	●		C:/USERS/XE/ORA
USERS	10 MB	41.3%	✓	UNLIMITED	●	✓	C:/USERS/XE/ORA

Рисунок 6.5 – Tablespace

Мониторинг состояния базы данных с использованием Oracle Enterprise Manager Database Express (EM Express) предоставляет обширные возможности для администрирования и оптимизации баз данных Oracle. EM Express обеспечивает удобный веб-интерфейс, который позволяет администраторам следить за состоянием базы данных в реальном времени и принимать меры для улучшения ее производительности.

Кроме того, EM Express предоставляет информацию о табличных пространствах, ресурсах процессора, активных сессиях и других ключевых аспектах работы базы данных. Такой обширный функционал делает EM Express не только инструментом мониторинга, но и эффективным средством администрирования и анализа производительности баз данных Oracle.

6.2 Хранение мультимедийных типов данных

Для работы с данными большого объёма СУБД Oracle предоставляет типы данных BLOB, CLOB, NCLOB и BFILE. Здесь LOB означает large object, или большой объект, и далее по тексту термины LOB и "большой объект" взаимозаменяемы. По сути, большой объект – это абстрактный тип для манипуляции данными большого объема внутри БД, а типы BLOB, CLOB, NCLOB и BFILE - его конкретные реализации.

В курсовом проекте технология мультимедийных типов данных отражена в виде хранимых изображений комнат. Изображения хранятся в таблице PHOTO.

Для начала необходимо создать директорию для хранения фотографий, как это показано на листинге 6.1.

```
create directory MEDIA_DIR as 'E:\CourseProj\photo';
```

Листинг 6.1 – Создание директории

В курсовом проекте используется технология BFILE для хранения фотографий. BFILE – это объект, предназначенный для работы с бинарными файлами, такими как изображения, звуки, видео и другие мультимедийные данные.

BFILE представляет собой специальный тип данных, который ссылается на внешний двоичный файл в операционной системе. Процедура InsertPhoto предназначена для вставки мультимедийных данных (фотографий) в базу данных и представлена на листинге 6.1.

```
PROCEDURE InsertPhoto(
    p_photo_room_type_id NUMBER,
    p_photo_source VARCHAR2
) AS
    v_room_type_count NUMBER;
    v_photo_id NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_room_type_count
    FROM ROOM_TYPES
    WHERE room_type_id = p_photo_room_type_id;

    IF v_room_type_count = 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Тип комнаты с указанным ID не
найден.');
    END IF;

    INSERT INTO PHOTO (photo_room_type_id, photo_source)
    VALUES (p_photo_room_type_id, BFILENAME('MEDIA_DIR', p_photo_source))
returning photo_id into v_photo_id;
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Фото успешно добавлено. ID: ' || v_photo_id);

EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Произошла ошибка: ' || SQLERRM);
        ROLLBACK;
END InsertPhoto;
```

Листинг 6.2 – Процедура InsertPhoto

В данном процедуре функция BFILENAME создает объект BFILE, указывая на файл с именем p_photo_source в директории MEDIA_DIR. Это обеспечивает привязку мультимедийного файла, хранящегося в операционной системе, к объекту BFILE в базе данных. Вставка происходит в таблицу PHOTO с использованием созданного объекта BFILE. Это позволяет хранить только путь к файлу в базе данных, сохраняя сам мультимедийный файл в операционной системе. Это может быть эффективным способом управления большими объемами мультимедийных данных, не загружая их напрямую в базу данных.

Таким образом, в данной процедуре используется технология BFILE для хранения фотографий, что позволяет эффективно управлять мультимедийными данными, предоставляя ссылку на внешние файлы вместо их полного хранения в базе данных.

7. Руководство пользователя

Так как база данных, поддерживает 3 типа пользователей, то есть и 3 сценария использования. Первый – при входе от администратора, второй – от сотрудника, третий – от гостя.

7.1 Сторона администратора отеля

Перед началом работы необходимо авторизироваться как пользователь Hotel_admin. После этого станут доступны функции администрирования базой данных отеля.

Для добавления нового типа услуг необходимо вызвать процедуру InsertServiceType и передать ей следующие параметры: название услуги, описание, суточная стоимость и идентификатор работника, закрепленного за этой услугой. После создания нового типа услуги его можно изменить с помощью процедуры UpdateServiceType, которая принимает 1 обязательный параметр – идентификатор типа услуги, а остальные параметры соответствуют параметрам процедуры InsertServiceType и не являются обязательными. Это значит, что по идентификатору типа услуги можно обновить только некоторые столбцы.

Для удаления типа услуги используется процедура DeleteServiceType, принимающая 1 параметр – идентификатор удаляемой услуги.

Аналогичные процедуры используются для работы со всеми остальными таблицами.

Также администратор занимается регистрацией гостей и сотрудников с помощью функций InsertGuest и InsertEmployee соответственно. В результате выполнения этих функций создается пользователь базы данных с определенными правами и ограничениями и возможностью аутентификации и авторизации в системе.

7.2 Сторона посетителя отеля

Перед началом работы необходимо пройти регистрацию через пользователя Hotel_admin. После этого необходимо пройти аутентификацию и сменить пароль, далее пользователь сможет самостоятельно подключаться к базе данных.

Перед бронированием места в отеле пользователь может просмотреть информацию о свободных комнатах на определенные даты с помощью процедуры GET_AVAILABLE_ROOMS. Для просмотра существующих в отеле тарифов проживания используется процедура Get_Tariff_Info, а для ознакомления с подробным описанием комнат – Get_Room_Info.

Для предварительного бронирования номера необходимо вызвать процедуру PRE_BOOKING и передать ей дату начала и окончания брони и выбранный тариф проживания.

Для подтверждения брони необходимо в первый день бронирования пройти регистрацию с помощью процедуры CHECK_IN, после чего статус брони измениться на «Одобрено администратором» и станет доступен заказ

дополнительных услуг. Со списком предоставляемых услуг можно ознакомиться с помощью процедуры Get_Service_Info, а заказать услугу с использованием процедуры Order_Service. Просмотреть список заказанных текущим пользователем услуг можно вызовом процедуры GET_MY_SERVICES.

При необходимости возможно изменить текущее бронирование, например сменить комнату или тариф, с помощью процедуры Edit_Booking.

Для выселения из отеля раньше окончания брони используется процедура Check_Out, которая также выставляет гостю счет за проживание и используемые услуги.

7.3 Сторона сотрудника отеля

Перед началом работы необходимо пройти регистрацию через пользователя Hotel_admin. После этого необходимо пройти аутентификацию и сменить пароль, далее пользователь сможет самостоятельно подключаться к базе данных.

Для получения доступа к информации о бронировании необходимо вызвать процедуру Get_Booking_Details_By_Id с передачей ей идентификатора брони. Для просмотра информации об услугах отеля используется процедура Get_Service_Info, а для просмотра заказанных услуг, за которыми закреплен текущий сотрудник – GET_MY_SERVICES.

Если для предоставления услуги необходимо получить информацию о месте проживания конкретного пользователя можно воспользоваться процедурой FIND_GUEST.

Для сотрудников доступна процедура QUIT_JOB, которая при отсутствии закрепленных за ними сервисов, предоставляет возможность увольнения.

Заключение

В рамках данной курсовой работы была разработана и реализована база данных для отеля, используя СУБД Oracle. Процесс начался с тщательного проектирования структуры базы данных, включая определение сущностей, их атрибутов и взаимосвязей. Модель базы данных была разработана с учетом основных потребностей отеля, включая информацию о бронированиях, персонале, гостях, номерах и предоставляемых услугах.

Создание объектов базы данных включало в себя создание 10 ключевых таблиц: BOOKING, BOOKING_STATE, EMPLOYEES, GUESTS, PHOTO, ROOM_TYPES, ROOMS, SERVICES, SERVICE_TYPES и TARIFF_TYPES. Каждая из этих таблиц была тщательно спроектирована для оптимального хранения и управления соответствующей информацией.

Кроме того, были созданы представления, индексы и ограничения, обеспечивающие эффективность работы базы данных и ее целостность. Применение технологии Oracle включало использование различных системных пакетов и утилит, предоставляемых СУБД, для оптимизации запросов, управления сеансами и обеспечения безопасности данных.

Особое внимание уделялось безопасности системы. С использованием ролей и профилей безопасности был создан надежный механизм управления доступом к данным, а соответствующие процедуры обеспечивают создание пользователей и присвоение им необходимых привилегий.

Работа также включала этап тестирования производительности базы данных. Проведенные тесты подтвердили стабильность и эффективность разработанной системы при работе с большим количеством данных.

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

Использование технологии мониторинга базы данных с применением Oracle Enterprise Manager Database Express (EM Express) в ходе разработки базы данных для гостиничного бизнеса оказалось весьма значимым и эффективным. EM Express предоставил нам ценные инструменты для наблюдения, анализа и управления работой базы данных в реальном времени.

Список используемых источников

1. Официальный сайт Oracle [Электронный ресурс] / Режим доступа – URL: https://www.oracle.com/ – Дата доступа: 05.10.2023.

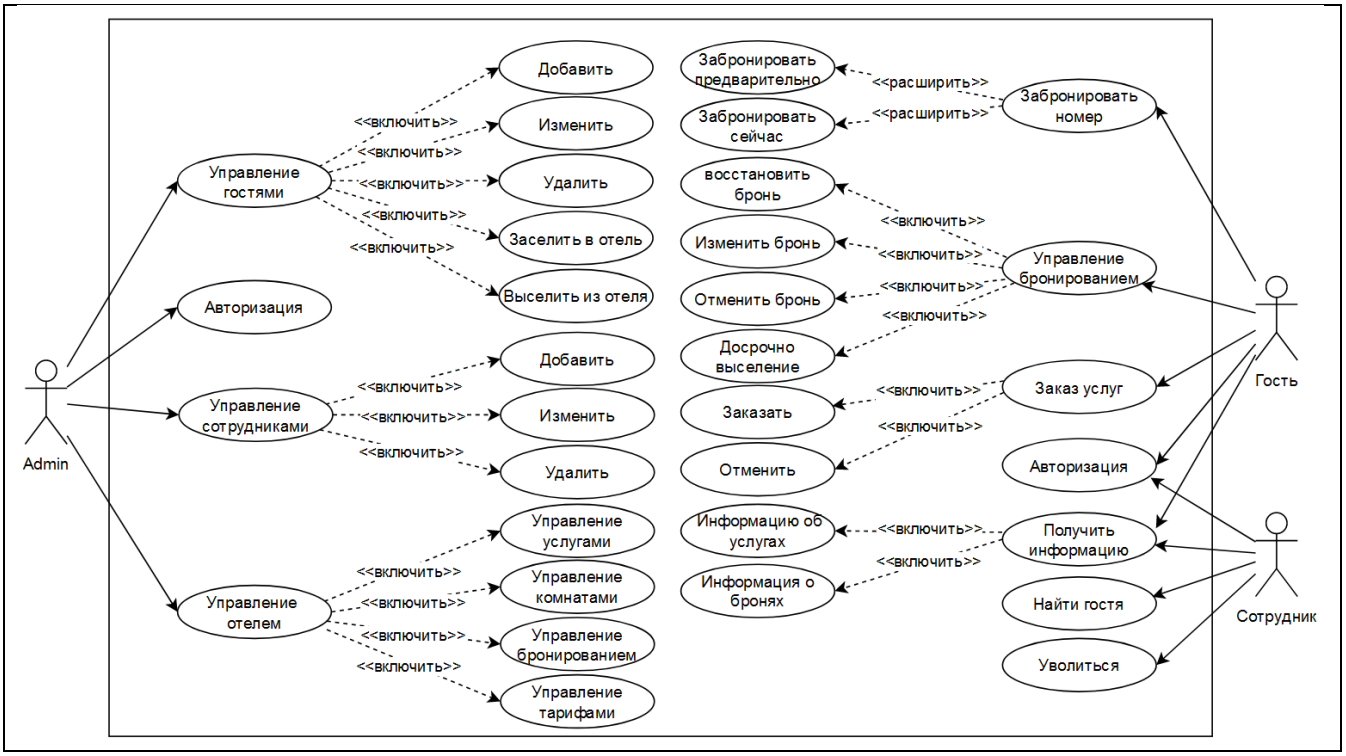
2. Официальная документация Oracle [Электронный ресурс] / Режим доступа – URL: https://docs.oracle.com/en/ – Дата доступа: 07.10.2023.

3. Информационный портал Oracle-patches [Электронный ресурс] / Режим доступа – URL: https://oracle-patches.com/ – Дата доступа: 10.10.2023.

4. Статья по Oracle Enterprise manager [Электронный ресурс] / Режим доступа – URL: https://habr.com/ru/companies/tinkoff/articles/525436/ – Дата доступа: 20.10.2023.

5. Работа с файлами в Oracle, [Электронный ресурс] / Режим доступа: https://docs.oracle.com/cd/F49540_01/DOC/server.815/a68001/utl_file.htm. – Дата доступа: 22.10.2023.

ПРИЛОЖЕНИЕ А



ПРИЛОЖЕНИЕ Б

```
   PROCEDURE InsertGuest(
       p_email NVARCHAR2,
       p_name NVARCHAR2,
       p_surname NVARCHAR2,
       p_username NVARCHAR2
   )
   AS
       v_username_exists NUMBER;
       v_guest_id NUMBER;
   BEGIN
       SELECT COUNT(*) INTO v_username_exists FROM ALL_USERS
       WHERE USERNAME = UPPER(p_username);

       IF v_username_exists > 0 THEN
           RAISE_APPLICATION_ERROR(-20002,'Ошибка:   Пользователь   с
таким именем ' || p_username || ' уже существует.');
       END IF;

       IF REGEXP_LIKE(p_email, '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-
Z|a-z]{2,4}$') = FALSE THEN
           RAISE_APPLICATION_ERROR(-20002,    'Неправильный    формат
email.');
       END IF;

       INSERT  INTO  GUESTS  (guest_email,  guest_name,  guest_surname,
USERNAME)
       VALUES   (p_email,   p_name,   p_surname,p_username)   RETURNING
guest_id INTO v_guest_id;
       COMMIT;

       EXECUTE IMMEDIATE 'CREATE USER ' || p_username ||
                         ' IDENTIFIED BY ' || p_username ||
                         ' DEFAULT TABLESPACE HOTEL_TS' ||
                         ' TEMPORARY TABLESPACE HOTEL_TEMP_TS'||
                         ' PROFILE PF_USER' ||
                         ' ACCOUNT UNLOCK'||
                         ' PASSWORD EXPIRE';
       EXECUTE IMMEDIATE 'GRANT Guest_role TO ' || p_username;

       DBMS_OUTPUT.PUT_LINE('Гость    успешно    создан.    Ваш    ID:
'||v_guest_id||' Логин: '|| p_username || ' Пароль: '|| p_username);

   EXCEPTION
       WHEN OTHERS THEN
           DBMS_OUTPUT.PUT_LINE('Произошла ошибка: ' || SQLERRM);
           ROLLBACK;
   END InsertGuest;
```

ПРИЛОЖЕНИЕ В

```
CREATE OR REPLACE PACKAGE UserPack AS
    PROCEDURE GET_AVAILABLE_ROOMS(
        p_capacity NUMBER,
        p_start_date DATE,
        p_end_date DATE);
    PROCEDURE BOOKING_NOW(
        p_room_id NUMBER,
        p_end_date DATE,
        p_tariff_id NUMBER);
    PROCEDURE PRE_BOOKING(
        p_room_id NUMBER,
        p_start_date DATE,
        p_end_date DATE,
        p_tariff_id NUMBER);
    PROCEDURE Get_BookingDetails_By_Id(
        p_booking_id NUMBER);
    PROCEDURE Edit_Booking(
        p_booking_id NUMBER,
        p_room_id NUMBER DEFAULT NULL,
        p_start_date DATE DEFAULT NULL,
        p_end_date DATE DEFAULT NULL,
        p_tariff_id NUMBER DEFAULT NULL);
    PROCEDURE Deny_Booking(p_booking_id NUMBER);
    PROCEDURE Restore_Booking(p_booking_id NUMBER);

    PROCEDURE Order_Service(
       p_service_type_id NUMBER,
       p_service_start_date DATE,
       p_service_end_date DATE);
    PROCEDURE Edit_Service(
       p_service_id NUMBER,
       p_service_type_id NUMBER DEFAULT NULL,
       p_service_start_date DATE DEFAULT NULL,
       p_service_end_date DATE DEFAULT NULL);

  PROCEDURE Get_Service_Info(p_id NUMBER DEFAULT NULL);
  PROCEDURE Get_Tariff_Info(p_id NUMBER DEFAULT NULL);
  PROCEDURE Get_Room_Info(p_id NUMBER DEFAULT NULL);
  FUNCTION Calculate_Stay_Cost(p_booking_id IN NUMBER) RETURN FLOAT;
  PROCEDURE Check_Out(p_booking_id NUMBER);
  PROCEDURE GET_MY_SERVICES;
  PROCEDURE GET_MY_BOOKINGS;

END UserPack;
/
```

ПРИЛОЖЕНИЕ Г

```
create or replace PROCEDURE EXPORT_TO_FILE(
    p_query IN NVARCHAR2,
    p_filename IN NVARCHAR2
)AS
    v_clob NCLOB;
    v_file UTL_FILE.FILE_TYPE;
BEGIN
    SELECT DBMS_XMLGEN.GETXML(p_query) INTO v_clob FROM DUAL;
    v_file := UTL_FILE.FOPEN('XML_DIR', p_filename || '.xml', 'w');
    BEGIN
        UTL_FILE.PUT(v_file, v_clob);
    EXCEPTION
        WHEN UTL_FILE.WRITE_ERROR THEN
            DBMS_OUTPUT.PUT_LINE('Ошибка записи в файл.');
    END;
    UTL_FILE.FCLOSE(v_file);
EXCEPTION
    WHEN NO_DATA_FOUND THEN
        DBMS_OUTPUT.PUT_LINE('Запрос не вернул данных.');
    WHEN UTL_FILE.INVALID_PATH THEN
        DBMS_OUTPUT.PUT_LINE('Неверный путь к директории.');
    WHEN UTL_FILE.INVALID_MODE THEN
        DBMS_OUTPUT.PUT_LINE('Неверный режим записи файла.');
    WHEN UTL_FILE.INVALID_FILEHANDLE THEN
        DBMS_OUTPUT.PUT_LINE('Неверный идентификатор файла.');
    WHEN UTL_FILE.INVALID_OPERATION THEN
        DBMS_OUTPUT.PUT_LINE('Неверная операция с файлом.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Произошла ошибка: ' || SQLERRM);
END;

begin
    EXPORT_TO_FILE('select * from employees', 'Employees');
end;

-- FILE_TO_CLOB
create PROCEDURE FILE_TO_CLOB(
        p_file_name IN NVARCHAR2,
        p_clob OUT CLOB
    )
    AS
        v_file     UTL_FILE.FILE_TYPE;
        v_filename NVARCHAR2(100);
        v_buffer   NVARCHAR2(32767);
BEGIN
    v_filename := p_file_name || '.xml';

    v_file := UTL_FILE.FOPEN('XML_DIR', v_filename, 'r');
    LOOP
        UTL_FILE.GET_LINE(v_file, v_buffer);
        IF v_buffer = '</ROWSET>' THEN
```

```
                p_clob := p_clob || v_buffer;
                EXIT;
            ELSE
                p_clob := p_clob || v_buffer;
            END IF;
        END LOOP;
        UTL_FILE.FCLOSE(v_file);
    END;

    -- Импорт гостей
    CREATE OR REPLACE PROCEDURE IMPORT_GUESTS_XML(p_file IN NVARCHAR2)
AS
        v_clob CLOB;
        v_xml XMLTYPE;-- := XMLTYPE(p_file);
    BEGIN
            FILE_TO_CLOB(p_file, v_clob);
        v_xml:= XMLTYPE(v_clob);
     FOR item IN (
        SELECT extractvalue(value(r), '/ROW/GUEST_ID') AS id,
               extractvalue(value(r), '/ROW/GUEST_EMAIL') AS email,
               extractvalue(value(r), '/ROW/GUEST_NAME') AS name,
               extractvalue(value(r), '/ROW/GUEST_SURNAME') AS surname,
               extractvalue(value(r), '/ROW/USERNAME') AS username

        FROM TABLE (XMLSEQUENCE(EXTRACT(v_xml, '/ROWSET/ROW'))) r
     )

     LOOP
       INSERT INTO GUESTS_XML (guest_email, guest_name, guest_surname,
USERNAME)
       VALUES (item.email, item.name, item.surname, item.username);
     END LOOP;

     COMMIT;
    EXCEPTION
      WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Произошла ошибка: ' || SQLERRM);
    END;
```

ПРИЛОЖЕНИЕ Д

```
CREATE OR REPLACE PROCEDURE INSERT_SERVICE_TYPES AS
BEGIN
    FOR i IN 1..100000 LOOP
        DECLARE
            v_daily_price FLOAT := DBMS_RANDOM.VALUE(5, 200);
            v_employee_id NUMBER;
        BEGIN
            SELECT employee_id
            INTO v_employee_id
            FROM EMPLOYEES
            WHERE ROWNUM = 1
            ORDER BY DBMS_RANDOM.VALUE;

            INSERT INTO SERVICE_TYPES (
                service_type_name,
                service_type_description,
                service_type_daily_price,
                service_type_employee_id
            ) VALUES (
                'service ' || i,
                'service description ' || i,
                v_daily_price,
                v_employee_id
            );
        END;
    END LOOP;
    COMMIT;

    DBMS_OUTPUT.PUT_LINE('Вставка успешно завершена.');
EXCEPTION
    WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Ошибка вставки: ' || SQLERRM);
        ROLLBACK;
END INSERT_SERVICE_TYPES;
```