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 салона красоты с использованием технологии средств диагностики»

Выполнил студент Голодок Анастасия Юрьевна
(Ф.И.О.)

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

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

Курсовой проект защищен с оценкой _____

Минск 2023

Содержание

Введение

Для эффективной работы любой организации необходим быстрый доступ к информации. Осуществить это можно с помощью баз данных способных удовлетворять уникальные потребности по хранению, управлению и администрированию данных.

На текущий момент существует разнообразие технологий доступа к данным и серверам баз данных, каждая из которых обладает своими уникальными характеристиками. Современные приложения для обработки данных ориентированы на обслуживание большого числа пользователей, включая тех, которые находятся в удаленных местах от основного сервера базы данных.

За основу базы данных «Салон красоты» была взята модель реляционных баз данных.

База данных — это организованная структура, предназначенная для хранения информации, систематизированная таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины. Реляционная база данных – это набор данных с предопределенными связями между ними, после сравнений современных моделей баз данных, данная модель была выбрана для реализации проекта, так как отвечает необходимым требованиям: простота организации связей и понятное хранение данных в таблицах. Каждая строка представляет отдельную запись или элемент данных в таблице, который содержит значения для каждого из столбцов.

Для работы была выбрана СУБД Oracle Database 19.

Цель данного курсового проекта – создание базы данных для хранения пользовательской информации, списков свободных и зарезервированных дат на услуги, наличие выбора сервисов для приобретения услуг, ознакомление с технологией средств диагностики и её применение в базе данных.

Основными задачами курсовой работы являются:

– проведение аналитического обзора аналогов;

– проектирование базы данных;

– реализация функциональности базы данных;

– проведение тестирования используемой технологии в базе данных;

– написание руководства пользователя.

База данных «Салон красоты» существенно упрощает работу сотрудников салона, а также предоставляет возможность администратору своевременно вносить необходимые изменения.

1 Постановка задачи

Целью данной работы является проектирование базы данных для программного средства салона красоты с технологией средств диагностики. В качестве модели данных следует использовать реляционную модель. Проектирование необходимо произвести таким образом, чтобы конечные данные соответствовали общим требованиям к информации в базе данных.

Функционал должен позволять:

– регистрировать, изменять и удалять пользователя;
– добавлять, изменять и удалять сотрудников;
– делать запись, удалять запись на услугу;
– просматривать доступные услуги;
– добавлять, изменять и удалять услуги;
– просматривать информацию о записях;
– добавлять и удалять отзывы;
– просматривать информацию об услугах и сотрудниках.

Весь доступный функционал будет представлен на UML-схеме (Приложение А). Для его реализации необходимо разработать ряд объектов базы данных. К таким объектам относятся:

– пользователи;
– таблицы;
– хранимые процедуры;
– функции.

1.1 Обзор прототипов и анализ

В процессе разработки программного обеспечения важным этапом является просмотр и изучение аналогов, поиск в них недостатков и достоинств. Перед тем как приступить к работе необходимо провести анализ и прочитать соответствующие статьи по данной теме. Для обзора были выбраны следующие сайты.

1.1.1 Культура маникюра

Рассмотрим в качестве примера сайт салона красоты «Культура маникюра». Бронь услуги осуществляется в 4 этапа:

1) Выбор услуги (рисунок 1.1.1);

Рисунок 1.1.1 – Интерфейс выбора услуги

2) Выбор доступного мастера и времени (рисунок 1.1.2);

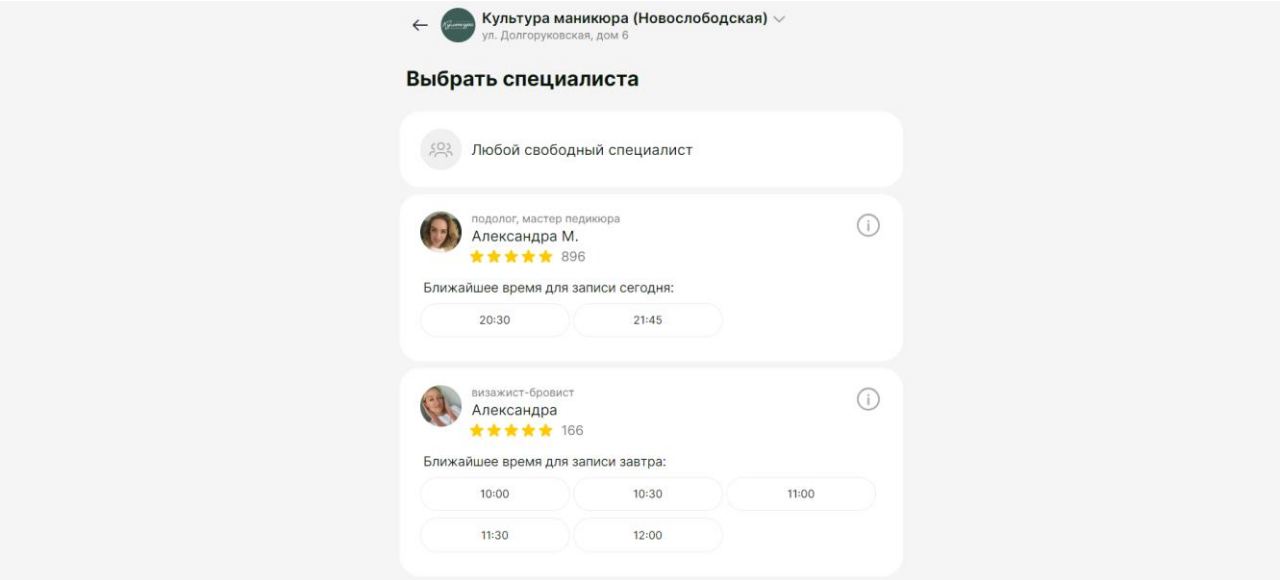


Рисунок 1.1.2 – Интерфейс выбора мастера

3) Выбор конкретного вида услуги (рисунок 1.1.3);

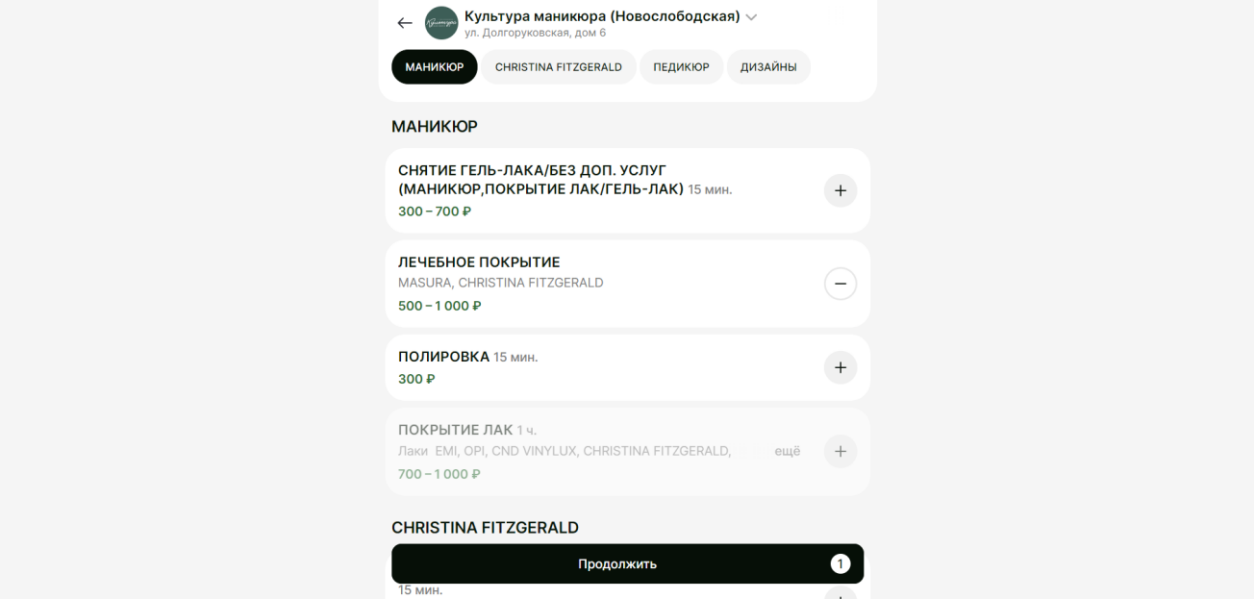


Рисунок 1.1.3 – Интерфейс выбора вида услуги

4) Заполнение личной информации (рисунок 1.1.3);

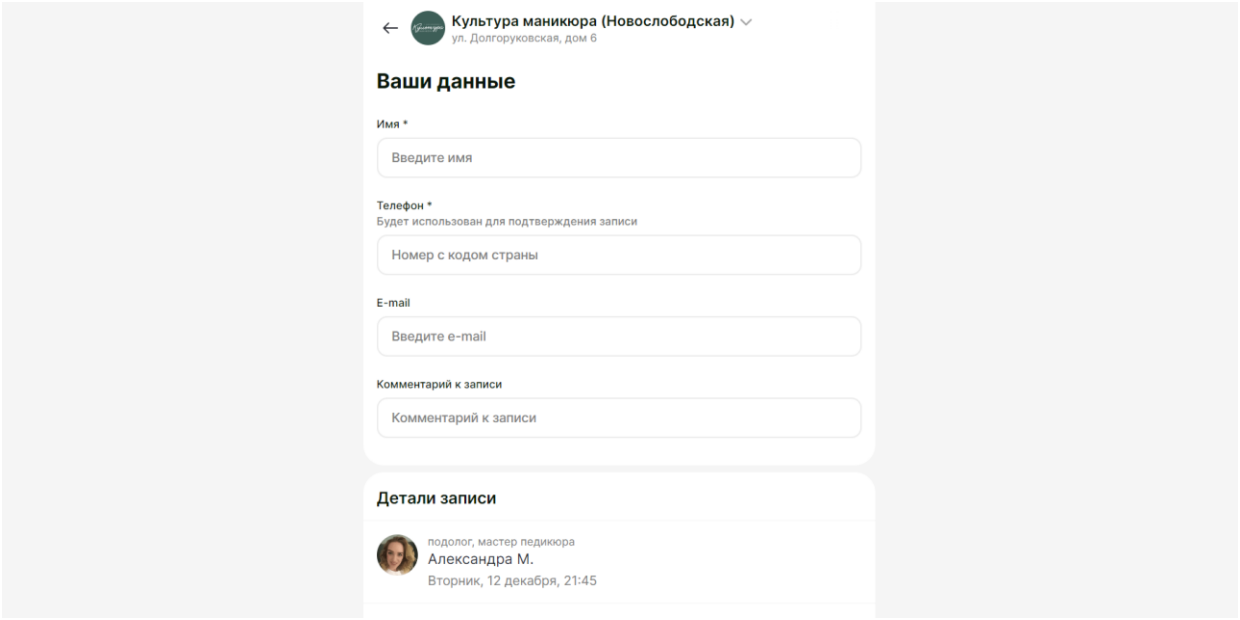


Рисунок 1.1.3 – Интерфейс заполнения формы

Сайт сделан логично, удобная последовательность при записи на услугу, осуществлена проверка на корректность введенных данных.

1.1.2 Сафина

Далее рассмотрим салон красоты «Сафина». Сайт предлагает большое количество услуг. Запись на услугу также происходит в 4 этапа (рисунок 1.1.4).

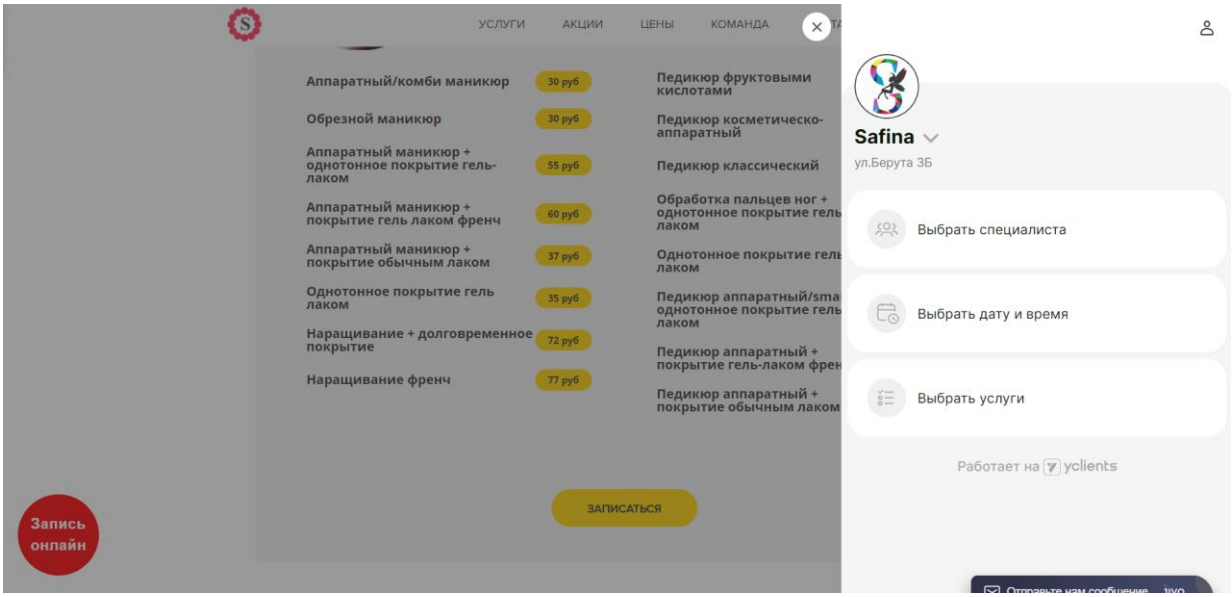


Рисунок 1.1.4 – Интерфейс страницы выбора услуги

В конце нужно указать личные данные клиента и подтвердить запись (рисунок 1.1.5).

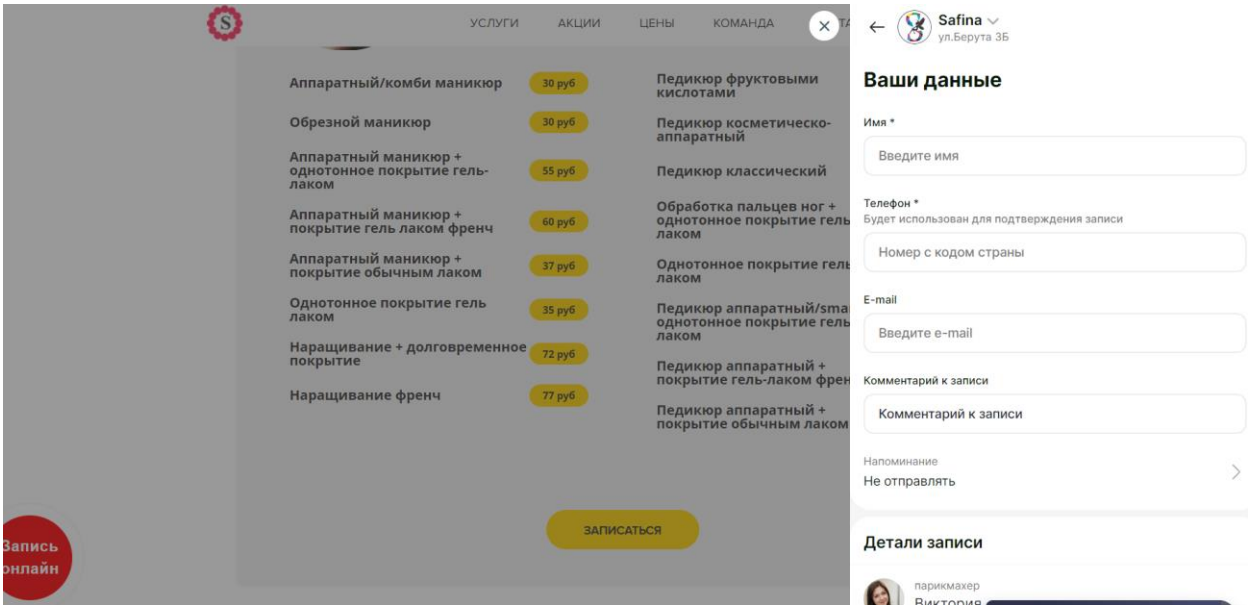


Рисунок 1.1.5 – Информация о клиенте

Функциональность сайта не отличается от предыдущего аналога, удобная последовательность при записи на процедуру.

1.2 Аналогичность и различие программных средств

Схожесть рассмотренных сайтов заключается в том, что они имеют практически одинаковый функционал: этапы записи на услугу, ввод одинаковых данных для записи, логичная последовательность пунктов для записи на услугу.

Анализируя ранее приведённые примеры, можно составить основные функциональные особенности для подобного типа приложений:

– регистрация и авторизация пользователя;
– управление услугами;
– хранимые процедуры;
– обеспечение записи на услуги.

Программа должна быть предназначена для различной аудитории пользователей. Это значит, что приложение должно быть простое и иметь доступный дизайн.

Все эти пункты и были учтены при выполнении данного курсового проекта.

2 Разработка модели базы данных

При разработке курсового проекта ниже будут описаны следующие объекты базы данных:

– таблицы;
– пользователи;
– хранимые процедуры;
– функции.

2.1 Создание таблиц

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

Для реализации работы базы данных было создано 5 таблиц.

Диаграмма базы со структурой связей представлена в приложении А.

Логически можно вывести 5 основных таблиц: CLIENTS, EMPLOYEES, REGISTRATION, SERVICES, REVIEWS.

Таблица CLIENTS представляет собой данные о пользователе, состоит из столбцов:

– clientID – идентификатор пользователя, тип number, первичный ключ;
– name – имя пользователя, тип nvarchar2(50);
– surname – фамилия пользователя, тип nvarchar2(50);
– phone – телефон пользователя, nvarchar2(20);
– email – адрес электронной почты пользователя, тип nvarchar2(100);
– password – пароль пользователя, тип nvarchar2(50).

Таблица EMPLOYEES представляет собой информацию о сотрудниках, состоит из столбцов:

– employeeID – идентификатор сотрудника, тип number, первичный ключ;
– serviceID – идентификатор услуги, тип number, внешний ключ;
– name – имя сотрудника, тип nvarchar2(50);
– surname – фамилия сотрудника, тип nvarchar2(50);
– positions – специализация сотрудника, тип nvarchar2(50);
– phone – телефон сотрудника, nvarchar2(20);
– email – адрес электронной почты сотрудника, тип nvarchar2(100).

Таблица REGISTRATION предназначена для хранения информации о записи на услугу, состоит из следующих столбцов:

– registrationID – идентификатор записи, тип number, первичный ключ;
– clientID – идентификатор клиента, тип number, внешний ключ;
– employeeID – идентификатор сотрудника, тип number, внешний ключ;
– dateTime – дата и время записи, тип TIMESTAMP;
– notes – комментарий, тип nvarchar2(255).

Таблица SERVICES представляет собой информацию о сервисах, состоит из столбцов:

– serviceID – идентификатор сервиса, тип number, первичный ключ;
– name – название сервиса, тип nvarchar2(100), внешний ключ;
– description – описание услуги, тип nvarchar2(255);
– price – цена услуги, тип decimal(10,2);
– image – изображение, тип nvarchar(255).

Таблица REVIEWS представляет собой информацию об арендованном инвентаре, состоит из столбцов:

– reviewID – идентификатор отзыва, тип number, первичный ключ;
– employeeID – идентификатор сотрудника, тип number, внешний ключ;
– rating – рейтинг сотрудника, тип number;
– comm – отзыв, тип nvarchar2(255).

3 Разработка необходимых объектов

3.1 Проектирование базы данных

Для достижения поставленной цели была разработана база данных PDB_KURSACH, которая была построена с использованием системы управления реляционными базами данных Oracle Database 19. В первую очередь необходимо было спроектировать корректную базу данных для работы. Моя база данных содержит следующие таблицы (рисунок 3.1):

– клиенты (CLIENTS);
– сотрудники (EMPLOYEES);
– запись на услуги (REGISTRATION);
– услуги (SERVICES);
– отзывы (REVIEWS).

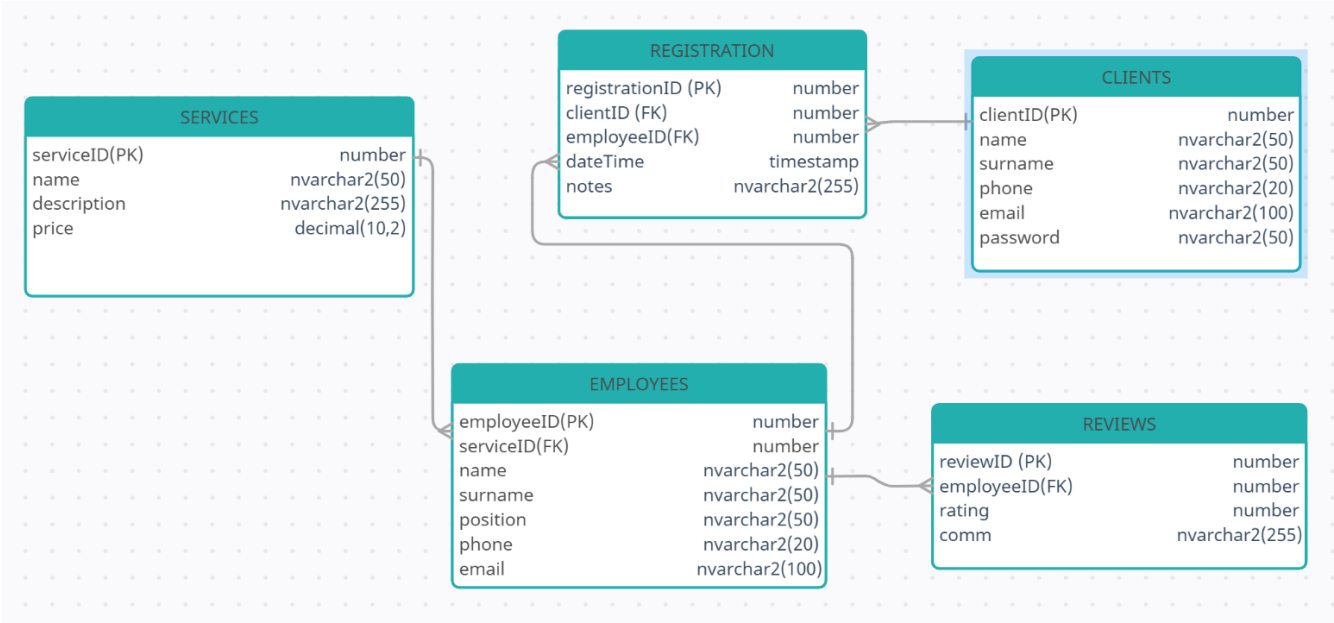


Рисунок 3.1 – Диаграмма базы данных

Листинг создания таблиц базы данных представлен в приложении Б.

3.2 Процедуры для решения поставленных задач

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

Использование хранимых процедур предоставляет возможность ограничить или полностью исключить прямой доступ пользователей к таблицам базы данных. Вместо этого пользователи получают разрешения на выполнение хранимых процедур, что обеспечивает косвенный и четко управляемый доступ к данным.

При разработке курсового проекта было создано 19 процедур для следующих целей:

1. Выборка данных из таблиц по различным полям;
2. Добавление пользователя при авторизации;
3. Удаление данных из таблиц;
4. Добавление данных в таблицы;
5. Изменение данных в таблицах;
6. Экспорт и импорт таблиц в формат json;
7. Заполнение таблицы на 100 000 строк.

Весь перечень созданных процедур будет представлен в Приложении В.

Все процедуры можно разделить на 3 категории для выполнения следующих задач:

1. CRUD-процедуры для каждой таблицы (create, update, delete);
2. Процедуры, реализующие выборку данных по запросу;
3. Процедуры выполняющие операции, которые нельзя отнести ни в одну из категорий, вынесены в специальные процедуры.

Далее мы более подробно остановимся на каждой из категорий.

3.2.1 CRUD-процедуры таблиц

Все CRUD-процедуры для каждой операции для всех таблиц построены по одному шаблону. CREATE принимают параметры для каждого поля таблицы, проверяет не существует ли уже такого значения, проверяет на корректность введенных данных, добавляет данные в таблицу.

Для обновления информации с помощью UPDATE в таблице, нужно знать идентификатор изменяемой строки, который передается в параметры процедуры вместе с другими параметрами. Каждый параметр соответствует полю таблицы.

DELETE-процедуры принимают один параметр – уникальный идентификатор записи таблицы, после чего осуществляют поиск по ней, найдя запись для удаления – удаляют её. Пример скрипта приведён в листинге 3.3.

3.2.2 Процедуры регистрации и авторизации

Для регистрации нового пользователя с помощью процедуры RegisterClient необходимо заполнить следующие поля: name, surname, phone, email, password. Осуществляется проверка корректности ввода адреса электронной почты, проверка на уникальность почты и пароля.

В случае успешной регистрации процедура создаст нового пользователя с ролью Client. Для дальнейшей работы необходимо авторизоваться с помощью процедуры ClientLogin. В качестве входных параметров используются p_email и p_password.

Параметры p_email и p_password проверяются для поиска данных клиента. В случае отсутствия пользователя с соответствующим адресом электронной почты и пароля возникает и обрабатывается ошибка. В случае успешной авторизации пользователю будет доступна информация об его аккаунте, об услугах и записях.

3.2.3 Предоставление и изменение информации о пользователе

Информацию о пользователе (имя, фамилия, номер телефона, адрес электронной почты, пароль) заполняет сам пользователь при регистрации. При необходимости любое поле может быть изменено, с помощью процедуры UpdateClient.

Процедура поиска информации о клиенте по его id – GetClientInfoByID выводит всю информацию (имя, фамилия, номер телефона, адрес электронной почты) о клиенте.

3.3 Пользователи

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

При проектировании базы данных было использовано 2 пользователя.

Первый пользователь – Client_Salon – зарегистрированный пользователь, имеет доступ к своим личным данным, может просматривать список услуг, осуществлять запись, оставлять отзывы. (листинг 3.1).

```
CREATE USER CLIENT_SALON IDENTIFIED BY CLIENTPASS
  PROFILE ANYPERSON_BASE_PROFILE
  account unlock;
GRANT CLIENT_ROLE TO CLIENT_SALON;
```

Листинг 3.1 – Скрипт создания пользователя Client_Salon

Второй пользователь – SYSADMIN – обладает иным перечнем прав. Имеет практически все права для работы с процедурами (листинг 3.9).

```
CREATE USER SYSADMIN IDENTIFIED BY SYSADMINPASS
  PROFILE ANYPERSON_BASE_PROFILE
  account unlock;
GRANT SYSADMIN_ROLE TO SYSADMIN;
```

Листинг 3.2 – Скрипт создания пользователя SYSADMIN

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

3.4 Функции

Элементы кода, которые повторяются во многих процедурах, были объединены в функции, для оптимизации кода.

Для проверки корректности введенного email используется функция IsEmailValid, представленная в листинге 3.3

```
CREATE OR REPLACE FUNCTION IsEmailValid(p_Email CLIENTS.email%type)
RETURN BOOLEAN
IS
    v_Pattern VARCHAR2(100) := '^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-
]+\.[a-zA-Z]{2,}$';
BEGIN
    RETURN REGEXP_LIKE(p_Email, v_Pattern);
END;
```

Листинг 3.3 – Скрипт создания функции IsEmailValid

Функции IsEmployeeExists, IsReviewExists, IsServiceExists используются для проверки, существует ли пользователь. Поскольку функции имеют одинаковый функционал, то ниже представлена только одна функция IsEmployeeExists (листинг 3.4).

```
CREATE OR REPLACE FUNCTION IsEmployeeExists(p_employeeID
EMPLOYEES.employeeID%type) RETURN BOOLEAN
IS
    v_Count NUMBER;
BEGIN
    SELECT COUNT(*) INTO v_Count FROM EMPLOYEES WHERE employeeID =
p_employeeID;
    RETURN v_Count > 0;
END;
```

Листинг 3.2 – Скрипт создания пользователя IsEmployeeExists

Функции облегчают работу с кодом, делают его более читабельным за счет уменьшения объема и разделения по файлам.

4 Описание процедур импорта и экспорта

Для выполнения задачи по импорту и экспорту необходимо было указать каталог для сохранения JSON-файла. Название каталога «UTL_DIR», путь 'C:\json'. Экспорт таблицы SERVICES в формате json осуществляется с помощью разработанной процедуры ExportToJSON, представленной ниже в листинге 4.1.

```
CREATE OR REPLACE PROCEDURE ExportToJSON
IS
    v_file sys.UTL_FILE.FILE_TYPE;
    v_row SERVICES%ROWTYPE;
BEGIN
    v_file :=
sys.UTL_FILE.FOPEN('UTL_DIR','BEAUTY_SALON_SERVICES.json','W');
    sys.UTL_FILE.PUT_LINE(v_file, '[');
    FOR v_row in (select JSON_OBJECT(
        'serviceID' is serviceID,
        'name' is CAST(name as nvarchar2(100)),
        'description' is CAST(description as nvarchar2(255)),
        'price' is CAST(price as decimal(10,2)),
        'image' is CAST(image as nvarchar2(255))
    ) AS JSON_DATA from SERVICES)
    LOOP
        sys.UTL_FILE.PUT_LINE(v_file ,v_row.JSON_DATA || ',');
    END LOOP;
    sys.UTL_FILE.PUT_LINE(v_file, ']');
    sys.UTL_FILE.FCLOSE(v_file);
    EXCEPTION
        WHEN OTHERS THEN
            sys.DBMS_OUTPUT.PUT_LINE('Error: ' || SQLCODE || ' - ' ||
SQLERRM);
    RAISE;
END;
```

Листинг 4.1 – Скрипт создания процедуры ExportToJSON

Код представляет собой процедуру ExportToJSON, которая выполняет экспорт данных из таблицы SERVICES в формат JSON и сохраняет результат в файл 'BEAUTY_SALON_SERVICES.json'. Процедура использует UTL_FILE для работы с файлами и создает JSON-объекты для каждой строки таблицы, записывая их в файл в виде массива JSON. В случае ошибки выводится сообщение с кодом и текстом ошибки.

Для импорта данных в таблицу SERVICES из файла формата json была разработана процедура ImportFromJSON. Данная процедура представлена в листинге 4.2:

```
CREATE OR REPLACE PROCEDURE ImportFromJSON
IS
BEGIN
    FOR json_rec IN (
        SELECT serviceID, name, description, price, image
        FROM JSON_TABLE(BFILENAME('UTL_DIR',
'BEAUTY_SALON_SERVICES.json'), '$[*]' COLUMNS (
            serviceID number PATH '$.serviceID',
            name varchar2(20) PATH '$.name',
            description varchar2(20) PATH '$.description',
            price varchar2(100) PATH '$.price',
            image varchar2(30) PATH '$.image'
        ))
    )
    LOOP
        BEGIN
            INSERT INTO SERVICES (serviceID, name, description,
price, image)
            VALUES (json_rec.serviceID, json_rec.name,
json_rec.description, json_rec.price, json_rec.image);
        EXCEPTION
            WHEN DUP_VAL_ON_INDEX THEN
                ROLLBACK;
                sys.DBMS_OUTPUT.PUT_LINE('Service with the id already
exists.');
            WHEN OTHERS THEN
                ROLLBACK;
                sys.DBMS_OUTPUT.PUT_LINE('Error inserting service: '
|| SQLERRM);
                RAISE;
        END;
    END LOOP;
END;
```

Листинг 4.2 – Скрипт создания процедуры ImportFromJSON

Процедура использует JSON_TABLE для извлечения данных из JSON-массива. Для каждой записи в массиве выполняется вставка данных в таблицу SERVICES. В случае возникновения ошибки дублирования ключа (DUP_VAL_ON_INDEX), происходит откат транзакции и выводится сообщение об уже существующей услуге с заданным идентификатором. В случае других ошибок также происходит откат транзакции, и выводится сообщение об ошибке с SQLERRM.

Таким образом происходит быстрое и удобное экспортирование и импортирование данных в таблицу.

5 Описание технологий

5.1 Средства диагностики

Oracle Database — это широко используемая реляционная система управления базами данных, которая предоставляет надежное и эффективное хранение, управление и обработку данных. Для обеспечения стабильной и производительной работы баз данных необходимо иметь средства диагностики, которые обеспечивают мониторинг, выявление проблем, анализ производительности и предоставление ценной информации для администраторов баз данных (DBA) и разработчиков.

Для того, чтобы начать диагностику необходимо открыть InstanceViewer во вкладке Database Status.

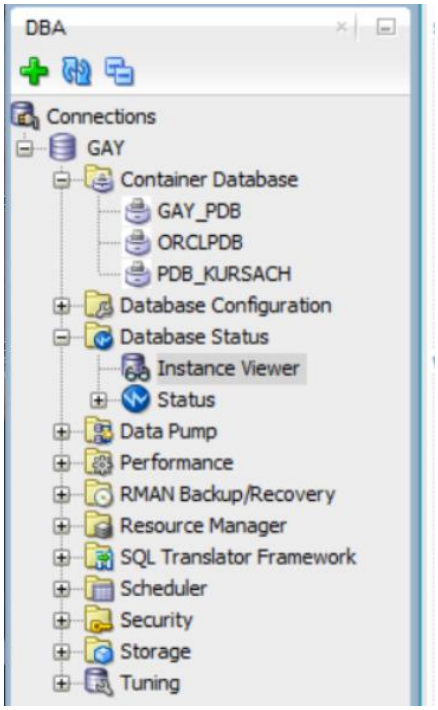


Рисунок 5.1 – Авторизация через DBA

Главная панель Instance Viewer представлена на рисунке 5.2. Данное средство предоставляет следующую информацию:

– позволяет видеть текущий статус базы данных, такой как запущена она или остановлена;

– отображает текущие настройки базы данных, такие как размер буфера, настройки кэша и другие параметры конфигурации;

– предоставляет информацию о событиях, происходящих в экземпляре, а также о блокировках, возможно, с указанием соответствующих SQL-запросов

– может включать данные о производительности, такие как загрузка CPU, использование памяти, статистика I/O и другие метрики;

– отображает активные сеансы и выполняющиеся SQL-запросы в базе данных;

– предоставляет доступ к журналам и событиям, что полезно для отслеживания и анализа происходящих событий.

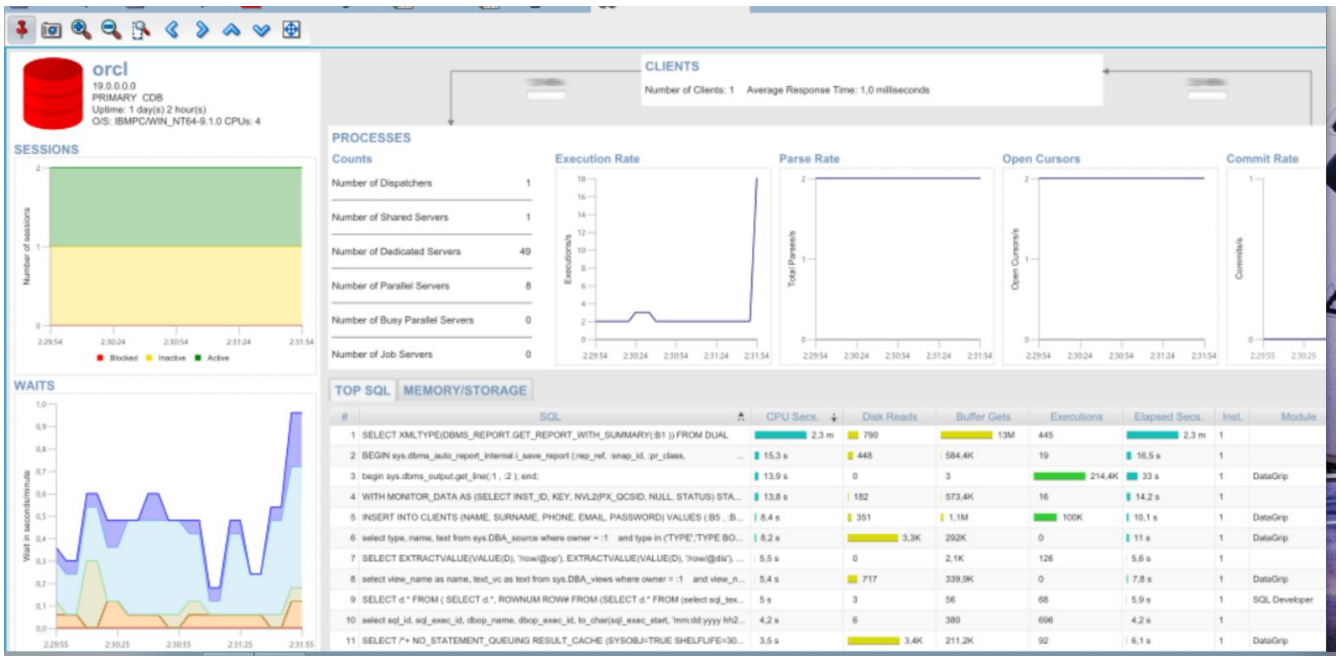


Рисунок 5.2 – Главный экран средства

Sessions (сеансы) – количество открытых сеансов пользователей или приложений, которые взаимодействуют с базой данных. Это может дать представление о нагрузке на базу данных. Представлено на рисунке 5.3.

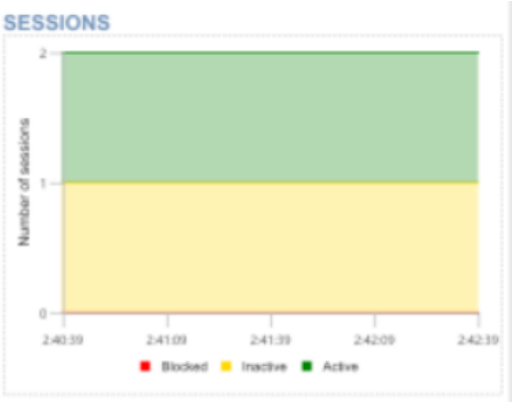


Рисунок 5.3 – Показатель Sessions

Waits (ожидания) – основной показатель, который указывает на нересурсоемкие операции. Если этот показатель высок, то система может замедляться, рисунок 5.4

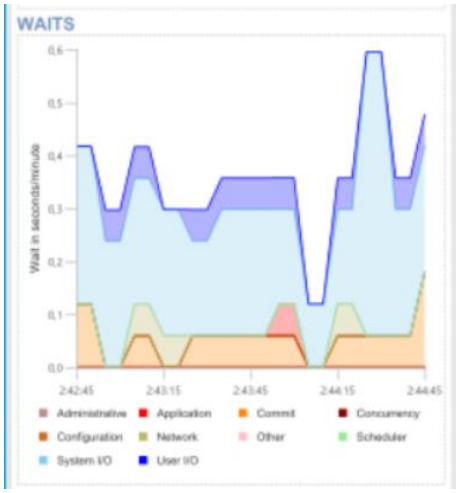


Рисунок 5.4 – Показатель Waits

Отношение DB CPU (CPU базы данных) представляет собой соотношение использования процессорного времени базы данных к общему процессорному времени системы. Этот параметр предоставляет индикатор того, насколько интенсивно база данных использует вычислительные ресурсы системы для достижения максимальной производительности. Отображено на рисунке 5.5.

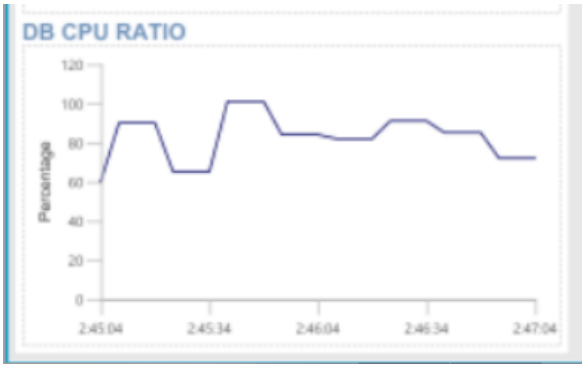


Рисунок 5.5 – Показатель DB CPU Ratio

Processes (процессы) – количество открытых процессов базы данных, к которым обращаются пользователи и приложения. Это может быть полезно при выявлении проблем с производительностью базы данных. Графики приведены на рисунке 5.6.

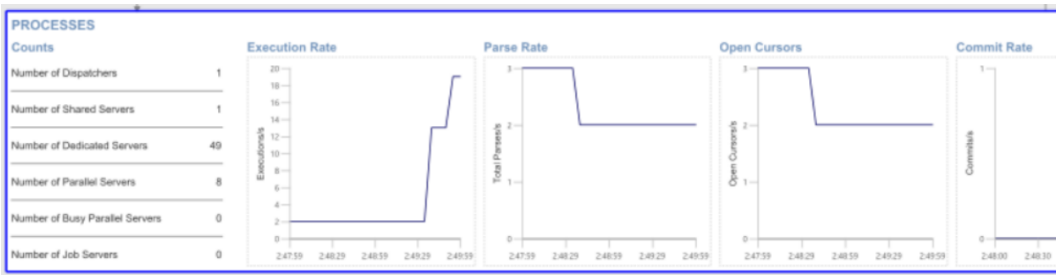


Рисунок 5.6 – Показатель Processes

Clients (клиенты) – количество открытых клиентов базы данных. Это может быть полезно при определении, сколько пользователей и приложений взаимодействуют с базой данных, рисунок 5.7.

Рисунок 5.7 – Показатель Clients

Top SQL (верхние SQL-запросы) – это список самых ресурсоемких запросов, которые находятся в очереди на выполнение. Это может помочь определить проблемы с производительностью базы данных и оптимизировать SQL-запросы. Приведен на рисунке 5.8.

Рисунок 6.9 – Показатель Top SQL

Все эти средства обеспечивают удобную диагностику базы данных, что позволяет работать с большими объемами информации.

6 Тестирование производительности

Для проверки производительности базы данных необходимо заполнить ее большим количеством данных и вычислить время выполнения одного запроса. К таблице CLIENTS была разработана процедура REGISTERCLIENT для заполнения таблицы строками в количестве 100000, представленная ниже в листинге 6.1:

```
CREATE OR REPLACE PROCEDURE RegisterClientTEST
is
    v_name CLIENTS.name%TYPE;
    v_surname CLIENTS.surname%TYPE;
    v_phone CLIENTS.phone%TYPE;
    v_email CLIENTS.email%TYPE;
    v_password CLIENTS.email%TYPE;
    v_count NUMBER := 100000;
BEGIN
    for i in 1..v_count loop
        --генерация чисел
        v_name := 'Имя';
        v_surname := 'Фамилия';
        v_phone := 'Номер' || i;
        v_email := 'Email' || i || '@example.com' ;--
        v_password := 'Пароль' || i;
    GOL_PDB_A.REGISTERCLIENT(
        p_Name => v_Name,
        p_surname => v_surname,
        p_phone => v_phone,
        p_email => v_email,
        p_password => v_password
        );
  end loop;
    end;
```

Листинг 6.1 – Описание процедуры REGISTERCLIENT

Таким образом было добавлено 100000 строк в таблицу. Примерное время выполнения – 282 миллисекунд (представлено на рисунке 6.1).

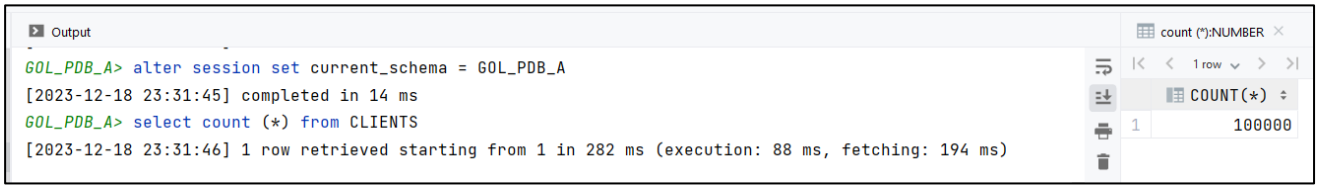


Рисунок 6.1 – Время выполнения запроса

7 Руководство пользователя

Для корректного использования написанной базы данных пользователем предусмотрен порядок выполнения доступных ему процедур. Полный список процедур с их входными параметрами представлен в приложении В.

В первую очередь пользователю необходимо зарегистрироваться с помощью процедуры RegisterClient. Далее процедура авторизации – ClientLogin. Если пользователь хочет изменить данные своего аккаунта, он может вызвать процедуру UpdateClient. Для удаления аккаунта достаточно использовать процедуру DeleteClient.

Для просмотра информации об услугах и сотрудниках можно вызвать процедуры GetServiceList, GetStaffList, GetSreviceInfoByName.

После того как клиент ознакомился с услугами и мастерами, предоставляемыми эти услуги, он может создать запись на услугу с помощью процедуры AddRegistration. Если пользователь решит перезаписаться, или отменить запись, он может использовать процедуру удаления старой записи – DeleteRegistration. Для просмотра информации о созданной записи следует использовать процедуру GetClientRegistrations.

Также пользователю доступна процедура для написания отзыва, делается это с помощью AddReview. Чтобы удалить отзыв нужно использовать DeleteReview. Для просмотра отзывов о сотруднике и среднего рейтинга, можно вызвать процедуры GetAverageRatingForEmployee, GetReviewsForEmployee.

Заключение

В ходе выполнения курсовой работы была разработана база данных для работы салона красоты на базе СУБД Oracle 19. В результате выполнения задач был углубленно изучен теоретический материал, а также проанализированы и разобраны различные готовые решения для разнообразных задач. Были созданы, описаны и успешно применены процедуры для экспорта и импорта данных. Функционально были выполнены следующие задачи:

– регистрация, изменение и удаление пользователя;
– добавление, изменение и удаление сотрудников;
– создание, удаление записи на услугу;
– просмотр доступных услуг;
– добавление, изменение и удаление услуг;
– просмотр информации о записях;
– добавление и удаление отзывов;
– просмотр информации об услугах и сотрудниках.

Корректно обрабатывается ошибочный ввод данных (например, ввод данных некорректного формата, попытка оставить поля пустыми), а также некоторые внутренние ошибки.

В соответствии с полученным результатом работы программы, можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объёме.

Список используемых источников

1. Oracle Documentation [Электронный ресурс] / Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/19/ Дата доступа: 10.11.2023

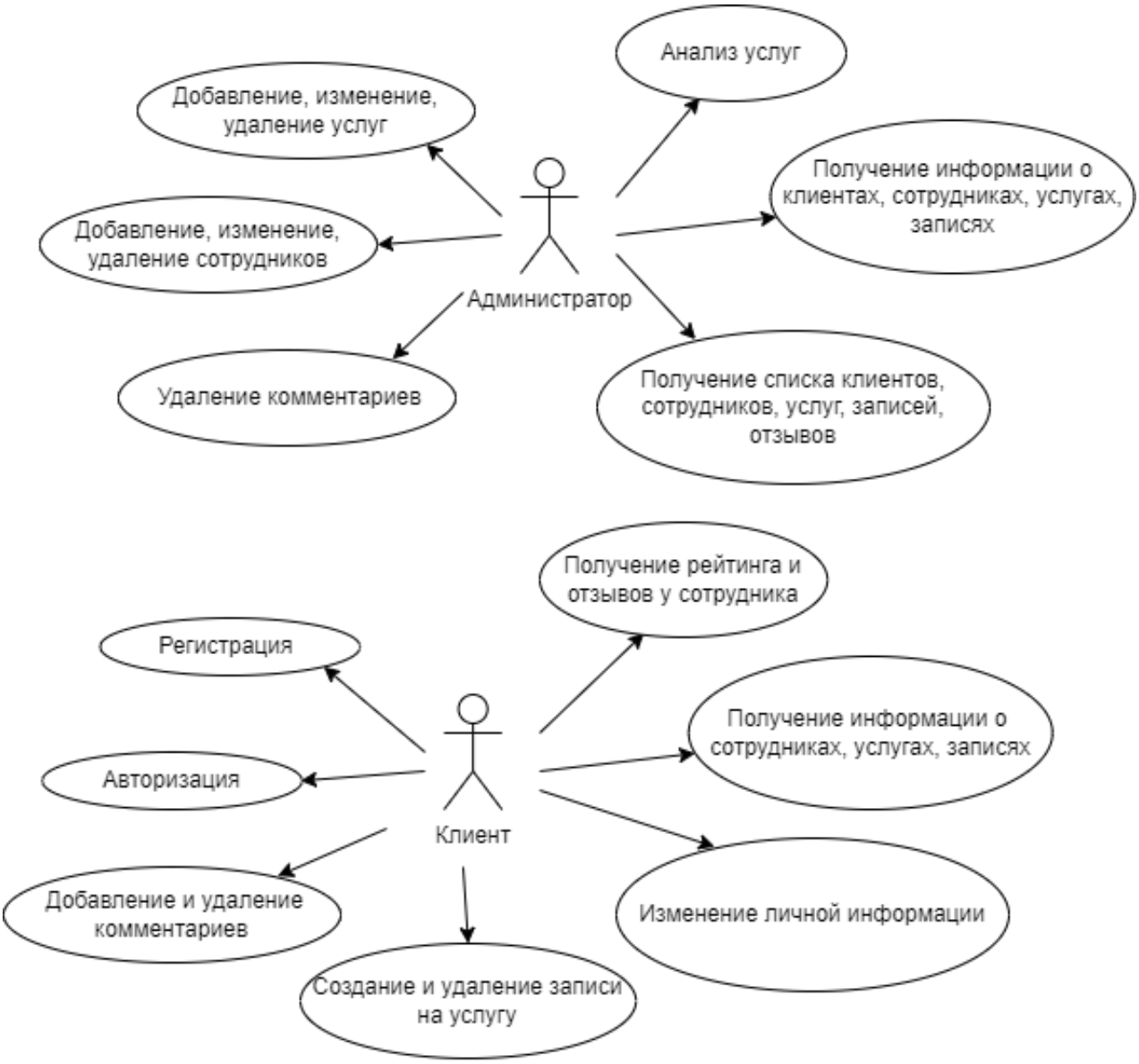
2. Oracle-dba.ru [Электронный ресурс] / Режим доступа: https://oracle-dba.ru Дата доступа: 23.11.2023

3. stackoverflow.com [Электронный ресурс] / Режим доступа: https://stackoverflow.com Дата доступа: 27.11.2023

4. DBMS_LOB [Электронный ресурс] / Режим доступа: https://docs.oracle.com/database/121/ARPLS/d_lob.htm#ARPLS600 Дата доступа: 1.12.2022

5. Oracle Documentation: JSON Developer's Guide [Электронный ресурс] / Режим доступа: https://docs.oracle.com/en/database/oracle/oracle-database/19/adxjs/index.html Дата доступа: 1.12.2023

Приложение А



Приложение Б

```
create table CLIENTS (
clientID NUMBER GENERATED BY DEFAULT AS IDENTITY NOT NULL,
name NVARCHAR2(50) NOT NULL,
surname NVARCHAR2(50) NOT NULL,
phone NVARCHAR2(20) NOT NULL,
email NVARCHAR2(100) UNIQUE NOT NULL,
password NVARCHAR2(50) UNIQUE NOT NULL,
CONSTRAINT PK_Clients PRIMARY KEY (clientID)
);

create table EMPLOYEES (
employeeID NUMBER GENERATED BY DEFAULT AS IDENTITY NOT NULL,
name NVARCHAR2(50) NOT NULL,
surname NVARCHAR2(50) NOT NULL,
positions NVARCHAR2(50) NOT NULL,
serviceID number not null,
phone NVARCHAR2(20) NOT NULL,
email NVARCHAR2(100) UNIQUE NOT NULL,
CONSTRAINT PK_Employees PRIMARY KEY (employeeID)
);

create table SERVICES (
serviceID NUMBER GENERATED BY DEFAULT AS IDENTITY NOT NULL,
name NVARCHAR2(100) unique NOT NULL,
description NVARCHAR2(255),
price DECIMAL(10,2) NOT NULL CHECK (price >= 0),
image NVARCHAR2(255),
CONSTRAINT PK_Services PRIMARY KEY (serviceID)
);

CREATE TABLE REVIEWS (
reviewID NUMBER GENERATED BY DEFAULT AS IDENTITY NOT NULL,
employeeID NUMBER NOT NULL,
rating NUMBER CHECK (rating >= 1 AND rating <= 5),
comm NVARCHAR2(255),
CONSTRAINT PK_Reviews PRIMARY KEY (reviewID)
);

CREATE TABLE REGISTRATION (
registrationID NUMBER GENERATED BY DEFAULT AS IDENTITY NOT
NULL,
clientID NUMBER NOT NULL,
employeeID NUMBER NOT NULL,
```

```
dateTime TIMESTAMP NOT NULL,
notes NVARCHAR2(255),
CONSTRAINT PK_Registration PRIMARY KEY (registrationID)
);

 ------------------------------------------------------------

ALTER TABLE REVIEWS ADD CONSTRAINT FK_Reviews_Employee FOREIGN
KEY (employeeID) REFERENCES EMPLOYEES(employeeID) ON DELETE
CASCADE;
-----------------------------------------------------
ALTER TABLE REGISTRATION ADD CONSTRAINT FK_Registration_Client
FOREIGN KEY (clientID) REFERENCES CLIENTS(clientID) ON DELETE
CASCADE;
ALTER TABLE REGISTRATION ADD CONSTRAINT
FK_Registration_Employee FOREIGN KEY (employeeID) REFERENCES
EMPLOYEES(employeeID) ON DELETE CASCADE;
-----------------------------------------------------
ALTER TABLE EMPLOYEES add constraint FK_Employee_Service
foreign key (serviceID) references SERVICES(serviceID) on
delete cascade;
```

Приложение В

Название процедуры	Описание	Принимаемые параметры
RegisterClientTEST	Процедура заполнения таблицы CLIENTS на 100000 строк	
AddService	Добавление новой услуги	p_name, p_description, p_price, p_image
UpdateService	Обновление услуги	p_serviceID, p_name, p_description, p_price, p_image
DeleteService	Удаление услуги	p_serviceID
GetServiceList	Получение списка услуг	
GetServiceInfoByName	Получение информации об услуге	p_serviceName
AddEmployee	Добавление сотрудника	p_name, p_surname, p_positions, p_phone, p_email, p_serviceID
UpdateEmployee	Изменение сотрудника	P_employeeID, p_name, p_surname, p_positions, p_phone, p_email, p_serviceID
DeleteEmployee	Удаление сотрудника	p_employeeID
GetEmployeeList	Получение списка сотрудников	
GetStaffInfo	Получение информации о сотруднике	p_employeeID
AddReview	Добавление отзыва	p_employeeID, p_rating, p_comm
DeleteReview	Удаление отзыва	p_reviewID
GetAverageRatingForEmployee	Получение рейтинга сотрудника	p_employeeName, p_employeeSurname
GetReviewsForEmployee	Получение списка отзывов для сотрудника	p_employeeName, p_employeeSurname
AddRegistration	Добавление записи	p_clientID, p_employeeID, p_dateTime, p_notes
DeleteRegistration	Удаление записи	p_registrationID

GetClientRegistrations	Получение информации о записи	p_clientID
GetClientList	Получение списка клиентов	
AnalyzeServicesByPeriod	Анализ количества проведенных услуг по периодам	p_StartDate, p_EndDate
AnalyzePopularServices	Получение популярных услуг	
RegisterClient	Регистрация пользователя	p_name, p_surname, p_phone, p_email, p_password
UpdateClient	Изменение клиента	p_clientID, p_name, p_surname, p_phone, p_email, p_password
DeleteClient	Удаление клиента	p_clientID
ClientLogin	Авторизация клиента	p_email, p_password
GetClientInfoByID	Получение информации о клиенте	p_clientID
ExportToJSON	Экспорт данных	
ImportFromJSON	Импорт данных	

Приложение Г

Скрипт распределения доступа по ролям:

```
CREATE ROLE SYSADMIN_ROLE;
CREATE ROLE CLIENT_ROLE;

GRANT CREATE SESSION TO SYSADMIN_ROLE;
GRANT EXECUTE ANY PROCEDURE TO SYSADMIN_ROLE;
GRANT CREATE SEQUENCE TO SYSADMIN_ROLE;
grant dba to SYSADMIN_role;

GRANT CREATE SESSION TO CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.AddReview TO CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.DeleteReview TO CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.AddRegistration TO CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.DeleteRegistration TO CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.GetClientRegistrations TO
CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.GetServiceList TO CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.GETEMPLOYEEList TO CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.GetServiceInfoByName TO CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.GetAverageRatingForEmployee TO
CLIENT_ROLE;
GRANT EXECUTE ON GOL_PDB_A.GetReviewsForEmployee TO
CLIENT_ROLE;
grant execute on GOL_PDB_A.RegisterClient to CLIENT_role;
grant execute on GOL_PDB_A.UpdateClient to CLIENT_role;
grant execute on GOL_PDB_A.DeleteClient to CLIENT_role;
grant execute on GOL_PDB_A.ClientLogin to CLIENT_role;
grant execute on GOL_PDB_A.GetClientInfoByID to CLIENT_role;

select * from dba_SYS_PRIVS where GRANTEE = 'SYSADMIN_ROLE';
select * from dba_SYS_PRIVS where GRANTEE = 'CLIENT_ROLE';

CREATE PROFILE ANYPERSON_BASE_PROFILE LIMIT
  PASSWORD_LIFE_TIME 180
  FAILED_LOGIN_ATTEMPTS 5
  PASSWORD_LOCK_TIME 1
  PASSWORD_REUSE_TIME 10
  PASSWORD_GRACE_TIME DEFAULT
  CONNECT_TIME 180
  IDLE_TIME 30;
COMMIT;
```