Министерство науки и высшего образования РФ
федеральное государственное автономное образовательное учреждение
высшего образования
«Омский государственный технический университет»
Факультет информационных технологий и компьютерных систем
Кафедра «Прикладная математика и фундаментальная информатика»

Расчетно-графическая работа

по дисциплине **Современные системы управления базами данных**

Студента	Гресь Владимир Игоревич
Курс	2 Группа ФИТм-241
Направление	02.04.02 Фундаментальная информатика и информационные технологии
Руководитель	доц.,к.н. Морарь Е.В.
Выполнил	_____ дата, подпись студента
Проверил	_____ дата, подпись руководителя

Омск 2025

Глава 1. Разработка базы данных для салона красоты

1.1 Краткое описание предметной области

Предметной областью данного проекта является салон красоты, который предоставляет различные косметические услуги клиентам. Салон имеет штат мастеров с определенной квалификацией, которые выполняют различные услуги.

1.1.1 Бизнес-правила

- Салон предлагает различные услуги (стрижки, маникюр, окрашивание и т.д.)
- Клиенты бронируют запись на услуги на определенное время
- Каждая услуга выполняется мастером салона
- Мастера имеют разную квалификацию и специализацию
- Услуги имеют определенную продолжительность и стоимость
- Клиенты могут оставлять отзывы после получения услуг

1.1.2 Основные сущности и атрибуты

- **Клиенты**: идентификатор, имя, фамилия, телефон, email, дата регистрации
- **Сотрудники**: идентификатор, имя, фамилия, должность, телефон, дата приема на работу
- **Услуги**: идентификатор, название, описание, продолжительность, стоимость
- **Записи**: идентификатор, идентификатор клиента, идентификатор сотрудника, идентификатор услуги, дата, время, статус

– **Отзывы**: идентификатор, идентификатор записи, оценка, комментарий, дата

1.2 Концептуальное проектирование базы данных

На этапе концептуального проектирования была разработана схема базы данных в нотации IDEF1X (Рис. 1). Данная схема отражает основные сущности и связи между ними:

1.3 Реализация базы данных в PostgreSQL

На основе разработанной концептуальной модели была реализована физическая модель базы данных в системе управления базами данных PostgreSQL. Были созданы следующие таблицы:

1.3.1 Таблица клиентов (clients)

Листинг 1.1: Создание таблицы клиентов

```
CREATE TABLE clients (
    client_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    email VARCHAR(100),
    registration_date DATE NOT NULL DEFAULT CURRENT_DATE
);
```

Комментарий: Таблица хранит основную информацию о клиентах салона. Поле `client_id` является первичным ключом с автоинкрементом. Обязательными полями являются имя, фамилия и телефон. Дата регистрации устанавливается автоматически.

1.3.2 Таблица сотрудников (employees)

Листинг 1.2: Создание таблицы сотрудников

```
CREATE TABLE employees (
    employee_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    position VARCHAR(50) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    hire_date DATE NOT NULL
);
```

Комментарий: Таблица содержит информацию о сотрудниках салона. Поле `position` используется для указания должности/специализации мастера.

1.3.3 Таблица услуг (services)

Листинг 1.3: Создание таблицы услуг

```
CREATE TABLE services (
    service_id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    description TEXT,
    duration INT NOT NULL, -- продолжительность в минутах
    price DECIMAL(10, 2) NOT NULL
);
```

Комментарий: Таблица хранит каталог услуг салона. Длительность услуги указывается в минутах, а цена - в десятичном формате с точностью до копеек.

1.3.4 Таблица записей (appointments)

Листинг 1.4: Создание таблицы записей

```
CREATE TABLE appointments (
    appointment_id SERIAL PRIMARY KEY,
    client_id INT NOT NULL,
    employee_id INT NOT NULL,
    service_id INT NOT NULL,
    appointment_date DATE NOT NULL,
    start_time TIME NOT NULL,
    status VARCHAR(20) NOT NULL DEFAULT 'scheduled', --
   scheduled, completed, cancelled
    CONSTRAINT fk_client FOREIGN KEY (client_id) REFERENCES
   clients(client_id),
    CONSTRAINT fk_employee FOREIGN KEY (employee_id) REFERENCES
   employees(employee_id),
    CONSTRAINT fk_service FOREIGN KEY (service_id) REFERENCES
   services(service_id)
);
```

Комментарий: Центральная таблица, связывающая клиентов, сотрудников и услуги. Содержит три внешних ключа и хранит информацию о дате, времени и статусе записи.

1.3.5 Таблица отзывов (feedback)

Листинг 1.5: Создание таблицы отзывов

```
CREATE TABLE feedback (
    feedback_id SERIAL PRIMARY KEY,
    appointment_id INT NOT NULL UNIQUE,
    rating INT NOT NULL CHECK (rating BETWEEN 1 AND 5),
    comment TEXT,
    feedback_date DATE NOT NULL DEFAULT CURRENT_DATE,
    CONSTRAINT fk_appointment FOREIGN KEY (appointment_id)
   REFERENCES appointments(appointment_id)
);
```

Комментарий: Таблица для хранения отзывов клиентов. Связана с таблицей записей отношением "один-к-одному"(один отзыв на одну запись). Оценка ограничена диапазоном от 1 до 5.

1.3.6 Диаграмма базы данных

На рисунке 2 представлена физическая диаграмма базы данных, отражающая таблицы, их поля и связи между ними:

1.4 Разработка запросов к базе данных

1.4.1 Запросы на выборку данных

Запрос 1: Все записи на определенную дату

Листинг 1.6: Запрос записей на определенную дату

```
SELECT
    a.appointment_id,
    CONCAT(c.first_name, ' ', c.last_name) AS client_name,
    CONCAT(e.first_name, ' ', e.last_name) AS employee_name,
    s.name AS service_name,
    a.start_time,
    s.duration,
    s.price,
    a.status
FROM
    appointments a
JOIN
    clients c ON a.client_id = c.client_id
JOIN
    employees e ON a.employee_id = e.employee_id
JOIN
    services s ON a.service_id = s.service_id
WHERE
    a.appointment_date = '2024-11-21'
ORDER BY
    a.start_time;
```

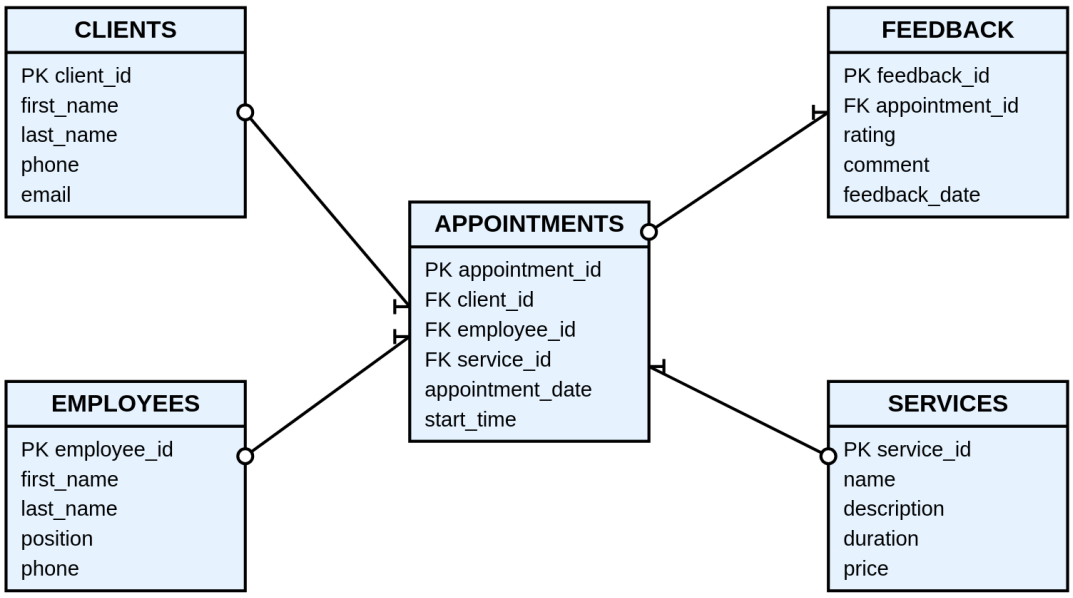

Рисунок 1 — Схема базы данных салона красоты в нотации IDEF1X

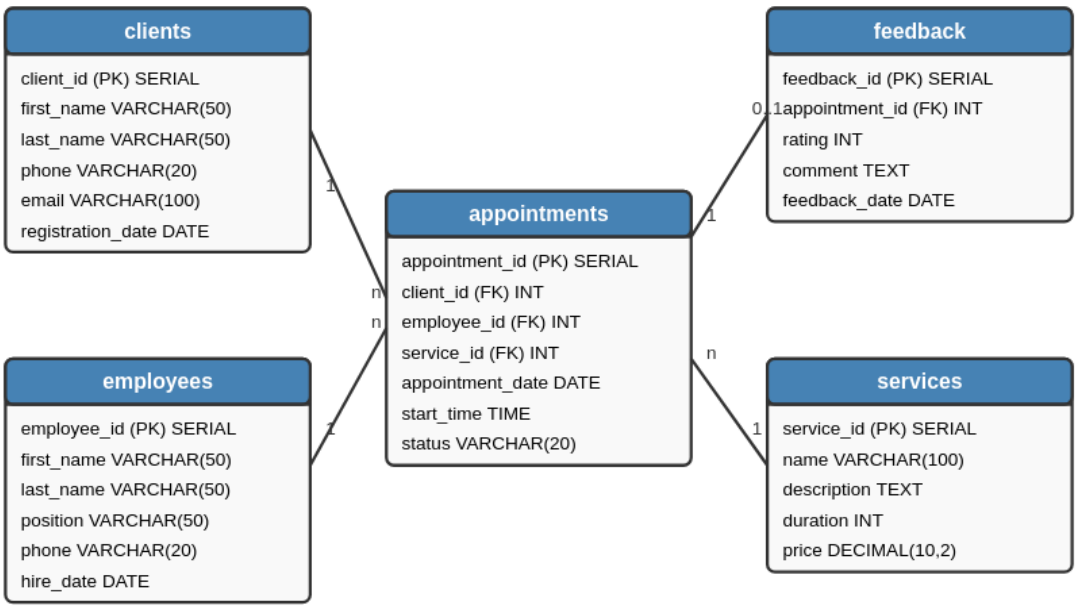

Рисунок 2 — Физическая диаграмма базы данных салона красоты

Комментарий: Запрос выбирает все записи на конкретную дату с информацией о клиентах, мастерах и услугах, отсортированные по времени начала.

Запрос 2: Рейтинг мастеров

Листинг 1.7: Запрос рейтинга мастеров по отзывам

```
SELECT
    CONCAT(e.first_name, ' ', e.last_name) AS employee_name,
    e.position,
    COUNT(f.feedback_id) AS feedback_count,
    ROUND(AVG(f.rating), 2) AS average_rating
FROM
    employees e
LEFT JOIN
    appointments a ON e.employee_id = a.employee_id
LEFT JOIN
    feedback f ON a.appointment_id = f.appointment_id
GROUP BY
    e.employee_id, e.first_name, e.last_name, e.position
HAVING
    COUNT(f.feedback_id) > 0
ORDER BY
    average_rating DESC;
```

Комментарий: Запрос вычисляет средний рейтинг мастеров на основе отзывов клиентов, включая только мастеров, получивших хотя бы один отзыв.

Запрос 3: Анализ популярности услуг

Листинг 1.8: Запрос анализа популярности услуг

```
SELECT
    s.name AS service_name,
    COUNT(a.appointment_id) AS appointment_count,
    SUM(s.price) AS total_revenue,
    ROUND(AVG(f.rating), 2) AS average_rating
```

```
FROM
    services s
LEFT JOIN
    appointments a ON s.service_id = a.service_id
LEFT JOIN
    feedback f ON a.appointment_id = f.appointment_id
WHERE
    a.status = 'completed'
    AND a.appointment_date BETWEEN '2025-01-01' AND '2025-05-01'
GROUP BY
    s.service_id, s.name
ORDER BY
    appointment_count DESC, total_revenue DESC;
```

Комментарий: Запрос анализирует популярность услуг за определенный период, включая количество записей, общую выручку и средний рейтинг.

Запрос 4: Клиенты, не посещавшие салон более 6 месяцев

Листинг 1.9: Запрос клиентов

```
SELECT
    c.client_id,
    CONCAT(c.first_name, ' ', c.last_name) AS client_name,
    c.phone,
    c.email,
    MAX(a.appointment_date) AS last_visit_date,
    CURRENT_DATE - MAX(a.appointment_date) AS
   days_since_last_visit
FROM
    clients c
LEFT JOIN
    appointments a ON c.client_id = a.client_id
WHERE
    a.status = 'completed'
GROUP BY
    c.client_id, c.first_name, c.last_name, c.phone, c.email
HAVING
    MAX(a.appointment_date) < CURRENT_DATE - INTERVAL '6 months'
ORDER BY
```

```
    last_visit_date ASC;
```

Комментарий: Запрос находит клиентов, не посещавших салон более 6 месяцев, для возможного маркетинга и возврата клиентов.

Запрос 5: Эффективность услуг

Листинг 1.10: Запрос анализа эффективности услуг

```
SELECT
    service_id,
    name AS service_name,
    price,
    duration,
    ROUND(price / NULLIF(duration, 0), 2) AS price_per_minute,
    ROUND(price / (NULLIF(duration, 0) / 60.0), 2) AS
   price_per_hour,
    RANK() OVER (ORDER BY price / NULLIF(duration, 0) DESC) AS
   profitability_rank
FROM
    services
WHERE
    duration > 0
ORDER BY
    price_per_minute DESC,
    price DESC;
```

Комментарий: Запрос анализирует эффективность услуг, рассчитывая стоимость минуты и часа услуги, а также ранжируя услуги по прибыльности.

1.4.2 Запрос на обновление данных

Листинг 1.11: Запрос на обновление цен услуг

```
UPDATE services
SET price = price * 1.1
WHERE service_id IN (1, 2, 3);
```

Комментарий: Запрос увеличивает цены на выбранные услуги на 10% для учета инфляции.

1.4.3 Запрос на вставку данных

Листинг 1.12: Запрос на добавление новой записи

```
INSERT INTO appointments (client_id, employee_id, service_id,
   appointment_date, start_time, status)
VALUES (1, 2, 3, '2025-05-15', '14:30:00', 'scheduled');
```

Комментарий: Запрос добавляет новую запись клиента на определенную услугу, дату и время.

1.4.4 Запрос на удаление данных

Листинг 1.13: Запрос на удаление старых отмененных записей

```
DELETE FROM appointments
WHERE status = 'cancelled'
AND appointment_date < CURRENT_DATE - INTERVAL '3 months';
```

Комментарий: Запрос удаляет отмененные записи старше 3 месяцев для очистки базы данных.

1.5 Разработка объектов базы данных

1.5.1 Хранимая процедура

Листинг 1.14: Хранимая процедура для расчета выручки салона

```
CREATE OR REPLACE PROCEDURE calculate_salon_revenue(
```

```
    start_date DATE,
    end_date DATE
)
LANGUAGE plpgsql
AS $$
DECLARE
    total_revenue DECIMAL(12,2) := 0;
    master_rec RECORD;
    service_rec RECORD;
BEGIN
    IF start_date > end_date THEN
        RAISE EXCEPTION 'Начальная дата не может быть позже коне
  чной даты';
    END IF;

    SELECT COALESCE(SUM(s.price), 0)
    INTO total_revenue
    FROM appointments a
    JOIN services s ON a.service_id = s.service_id
    WHERE a.appointment_date BETWEEN start_date AND end_date
    AND a.status = 'completed';

    RAISE NOTICE 'Период анализа: с % по %', start_date,
  end_date;
    RAISE NOTICE 'Общая выручка за период: % руб.',
  total_revenue;
    RAISE NOTICE
  '------------------------------------------------------';

    RAISE NOTICE 'ВЫРУЧКА ПО МАСТЕРАМ:';
    RAISE NOTICE
  '------------------------------------------------------';

    FOR master_rec IN
        SELECT
            e.employee_id,
            CONCAT(e.first_name, ' ', e.last_name) AS
  employee_name,
            COUNT(a.appointment_id) AS appointment_count,
            SUM(s.price) AS revenue,
            ROUND(AVG(COALESCE(f.rating, 0)), 1) AS avg_rating
        FROM employees e
```

```
        LEFT JOIN appointments a ON e.employee_id = a.
    employee_id
        LEFT JOIN services s ON a.service_id = s.service_id
        LEFT JOIN feedback f ON a.appointment_id = f.
    appointment_id
        WHERE (a.appointment_date BETWEEN start_date AND
    end_date OR a.appointment_date IS NULL)
        AND (a.status = 'completed' OR a.status IS NULL)
        GROUP BY e.employee_id, e.first_name, e.last_name
        ORDER BY SUM(COALESCE(s.price, 0)) DESC NULLS LAST
    LOOP
        RAISE NOTICE 'Мастер: %, Кол-во услуг: %, Выручка: % руб
    ., Ср. рейтинг: %',
            master_rec.employee_name,
            master_rec.appointment_count,
            COALESCE(master_rec.revenue, 0),
            master_rec.avg_rating;
    END LOOP;

    RAISE NOTICE
    '------------------------------------------------------';
    RAISE NOTICE 'ПОПУЛЯРНОСТЬ УСЛУГ:';
    RAISE NOTICE
    '------------------------------------------------------';

    FOR service_rec IN
        SELECT
            s.name AS service_name,
            COUNT(a.appointment_id) AS appointment_count,
            SUM(s.price) AS revenue,
            ROUND(AVG(COALESCE(f.rating, 0)), 1) AS avg_rating
        FROM services s
        LEFT JOIN appointments a ON s.service_id = a.service_id
        LEFT JOIN feedback f ON a.appointment_id = f.
    appointment_id
        WHERE (a.appointment_date BETWEEN start_date AND
    end_date OR a.appointment_date IS NULL)
        AND (a.status = 'completed' OR a.status IS NULL)
        GROUP BY s.service_id, s.name
        ORDER BY COUNT(a.appointment_id) DESC NULLS LAST
    LOOP
        RAISE NOTICE 'Услуга: %, Кол-во: %, Выручка: % руб., Ср.
     рейтинг: %',
```

```
            service_rec.service_name,
            service_rec.appointment_count,
            COALESCE(service_rec.revenue, 0),
            service_rec.avg_rating;
    END LOOP;

    RAISE NOTICE
   '------------------------------------------------------';
    RAISE NOTICE 'Анализ завершен.';
END;
$$;

-- Вызов процедуры
CALL calculate_salon_revenue('2025-01-01', '2025-05-01');
```

Комментарий: Хранимая процедура генерирует отчет о выручке салона за указанный период, включая данные по мастерам и услугам. Выполняет валидацию входных дат.

1.5.2 Триггер

Листинг 1.15: Триггер для отслеживания изменений в записях

```
-- Функция триггера для обновления даты последнего изменения
CREATE OR REPLACE FUNCTION update_last_modified()
RETURNS TRIGGER AS $$
BEGIN
    NEW.last_modified = CURRENT_TIMESTAMP;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Добавление колонки для отслеживания изменений
ALTER TABLE appointments
ADD COLUMN IF NOT EXISTS last_modified TIMESTAMP DEFAULT
   CURRENT_TIMESTAMP;

-- Создание триггера
CREATE TRIGGER update_appointment_timestamp
BEFORE UPDATE ON appointments
```

```
FOR EACH ROW
EXECUTE FUNCTION update_last_modified();
```

Комментарий: Триггер автоматически обновляет время последнего изменения записи при любом обновлении строки в таблице appointments, что позволяет отслеживать историю изменений.

1.5.3 Роль пользователя

Листинг 1.16: Создание роли пользователя с ограниченными правами

```
-- Создание роли для администратора салона
CREATE ROLE salon_admin;

-- Предоставление прав на таблицы
GRANT SELECT, INSERT, UPDATE ON
    clients, employees, services, appointments, feedback
TO salon_admin;

-- Ограничение прав на удаление
GRANT DELETE ON
    feedback
TO salon_admin;

-- Права на выполнение процедуры
GRANT EXECUTE ON PROCEDURE calculate_salon_revenue TO
   salon_admin;

-- Создание пользователя с этой ролью
CREATE USER manager_anna PASSWORD 'secure_password123';
GRANT salon_admin TO manager_anna;
```

Комментарий: Создается роль администратора салона с правами на просмотр, добавление и изменение данных, но с ограниченными правами на удаление (только для отзывов). Также предоставляется право на выполнение процедуры расчета выручки.

1.6 Заключение

В ходе выполнения расчетно-графической работы была разработана база данных для салона красоты, которая позволяет эффективно управлять информацией о клиентах, сотрудниках, услугах, записях и отзывах.

Были выполнены все этапы проектирования базы данных:

- Проведен анализ предметной области, выделены основные сущности и их атрибуты
- Разработана концептуальная модель в нотации IDEF1X
- Создана физическая модель базы данных в PostgreSQL
- Разработаны запросы для выборки, обновления, вставки и удаления данных
- Созданы хранимая процедура, триггер и роль пользователя