Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский университет ИТМО»
(Университет ИТМО)

Факультет прикладной информатики
Образовательная программа Мобильные и сетевые технологии
Направление подготовки 09.03.03 Мобильные и сетевые технологии

ОТЧЕТ по Лабораторной работе № 5
по дисциплине «Проектирование и реализация баз данных»

Обучающийся: Шукалов Андрей Денисович
Преподаватель: Говорова Марина Михайловна

Санкт-Петербург, 2025

Описание задания

Цель работы

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание

1. Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)

2. Создать 7 оригинальных триггеров.

Схема базы данных

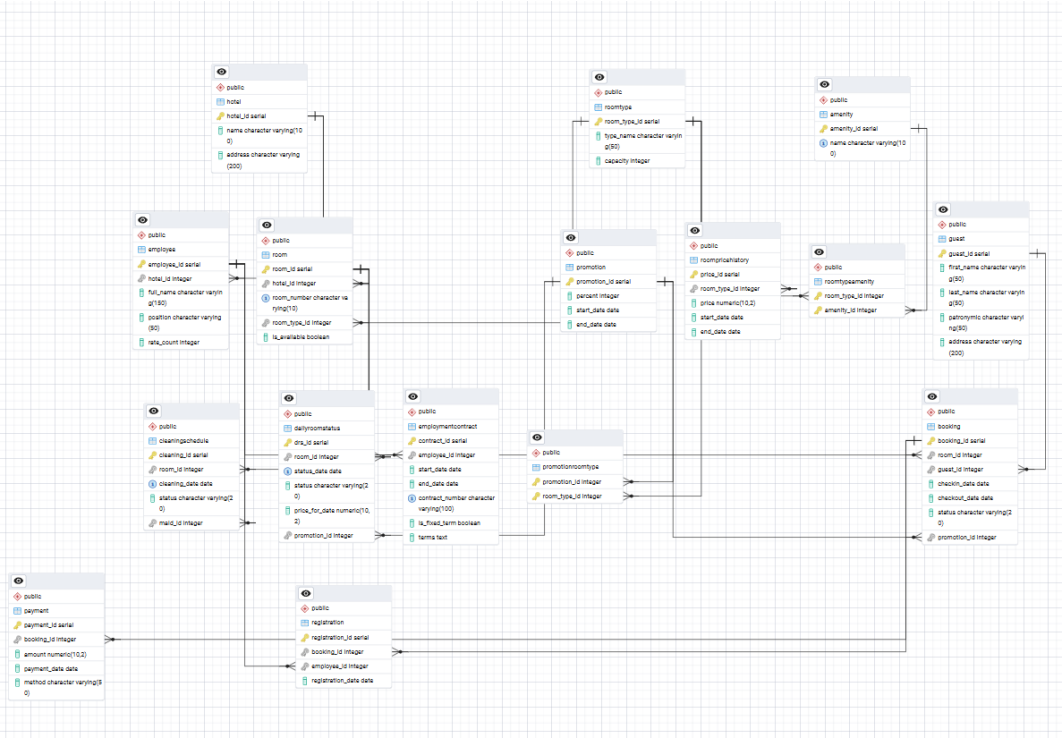

Рис. 1: ER диаграмма

Ход работы

1 Создание процедур

Процедура 1

Для увеличения цены всех номеров на 5 %, если в отеле нет свободных номеров.

```
CREATE OR REPLACE PROCEDURE increase_price(p_hotel_id INT) -- входной параметр
процедуры
LANGUAGE plpgsql
AS $$
DECLARE
    free_count INT; -- количество свободных номеров
BEGIN
    -- считаем количество свободных номеров в данном отеле
    SELECT COUNT(*) INTO free_count
    FROM Room r
    LEFT JOIN DailyRoomStatus drs
      ON r.room_id = drs.room_id AND drs.status_date = current_date
    WHERE r.hotel_id = p_hotel_id
      AND (drs.status IS NULL OR drs.status = 'free');

    IF free_count = 0 THEN
        -- В истории цен обновляем последние записи, чтобы увеличить цены на эти
номера
        UPDATE RoomPriceHistory
        SET price = ROUND(price * 1.05, 2)
        WHERE room_type_id IN (
            SELECT DISTINCT room_type_id
            FROM Room
            WHERE hotel_id = p_hotel_id
        )
        AND (end_date IS NULL OR end_date >= current_date - INTERVAL '1 day')
        AND start_date <= current_date;
    END IF;
END;
$$;
```

```
hotel_db=# SELECT * FROM RoomPriceHistory;
 price_id | room_type_id |  price   | start_date |  end_date
----------+--------------+----------+------------+------------
        1 |            1 |  5250.00 | 2025-01-01 | 2025-10-01
        2 |            2 | 10500.00 | 2025-01-01 | 2025-10-01
        3 |            3 | 21000.00 | 2025-01-01 | 2025-10-01
        4 |            4 | 26250.00 | 2025-01-01 | 2025-10-01
(4 rows)


hotel_db=# call increase_price(1);
CALL
hotel_db=# SELECT * FROM RoomPriceHistory;
 price_id | room_type_id |  price   | start_date |  end_date
----------+--------------+----------+------------+------------
        1 |            1 |  5512.50 | 2025-01-01 | 2025-10-01
        2 |            2 | 11025.00 | 2025-01-01 | 2025-10-01
        3 |            3 | 22050.00 | 2025-01-01 | 2025-10-01
        4 |            4 | 27562.50 | 2025-01-01 | 2025-10-01
(4 rows)
```

Рис. 2: Результат работы процедуры

Процедура 2

Для получения информации о свободных одноместных номерах отеля на завтрашний день.

```
CREATE OR REPLACE FUNCTION get_free_rooms(
    p_hotel_id INT DEFAULT NULL
)
RETURNS TABLE (
    room_id INT,
    room_number TEXT,
    hotel_id INT,
    hotel_name TEXT,
    room_type_id INT,
    type_name TEXT,
    price_for_date NUMERIC
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY
    SELECT
        r.room_id,
        r.room_number::TEXT,
        r.hotel_id,
        h.name::TEXT,
        rt.room_type_id,
        rt.type_name::TEXT,
        drs.price_for_date
    FROM Room r -- среди всех комнат
    JOIN RoomType rt ON r.room_type_id = rt.room_type_id -- объединяем с типами
по совпадению room_type_id
```

```
    JOIN Hotel h ON r.hotel_id = h.hotel_id -- объединяем с отелями по hotel_id
    LEFT JOIN DailyRoomStatus drs -- left join чтобы не пропустить комнаты, о которых
нет записей
      ON drs.room_id = r.room_id
     AND drs.status_date = current_date + INTERVAL '1 day' -- свободны на завтра
    WHERE rt.capacity = 1 -- размер 1
      AND (drs.status = 'free' OR drs.status IS NULL)
      AND (p_hotel_id IS NULL OR r.hotel_id = p_hotel_id);
END;
$$;
```

```
 room_id | room_number | hotel_id |  hotel_name  | room_type_id | type_name | price_for_date
---------+-------------+----------+--------------+--------------+-----------+----------------
       1 | 101         |        1 | ╬Єхы№ ╠юёътр |            1 | ёЄрэфрЁЄ   |
(1 row)
```

Рис. 3: Результат работы процедуры

Процедура 3

Бронирование двухместного номера в гостинице на заданную дату и количество дней проживания.

```
CREATE OR REPLACE PROCEDURE book_double_room(
    p_hotel_id INT,
    p_guest_id INT,
    p_start_date DATE,
    p_days INT,
    OUT p_booking_id INT
)
LANGUAGE plpgsql
AS $$
DECLARE
    p_end_date DATE; -- Дата окончания брони
    v_room_id INT;
BEGIN
    p_booking_id := NULL;
    p_end_date := p_start_date + (p_days - 1);
    SELECT r.room_id
    FROM Room r
    JOIN RoomType rt ON r.room_type_id = rt.room_type_id
    WHERE r.hotel_id = p_hotel_id
      AND rt.capacity = 2
      --Проверяем наличие бронирование на этот номер, даты которых пересекаются
с нашими
      AND NOT EXISTS (
          SELECT 1 FROM Booking b
          WHERE b.room_id = r.room_id
            AND b.status IN ('booked','checked_in')
            AND NOT (b.checkout_date < p_start_date OR b.checkin_date > p_end_date)
      )
    LIMIT 1
    INTO v_room_id;

    IF v_room_id IS NULL THEN
        RETURN; -- Если не нашлось пустых 2х местных номеров, то выходим
    END IF;

    --Вставляем новое бронирование
    INSERT INTO Booking (room_id, guest_id, checkin_date, checkout_date, status)
    VALUES (v_room_id, p_guest_id, p_start_date, p_end_date, 'booked');
END;
$$;
```

```
hotel_db=# \e
CREATE PROCEDURE
hotel_db=# SELECT * FROM Booking;
 booking_id | room_id | guest_id | checkin_date | checkout_date |   status    | promotion_id
------------+---------+----------+--------------+---------------+-------------+--------------
          1 |       2 |        1 | 2025-09-10   | 2025-09-15    | checked_in  |            1
          2 |       3 |        2 | 2025-09-05   | 2025-09-10    | checked_out |            1
          3 |       5 |        3 | 2025-09-12   | 2025-09-14    | booked      |
          4 |       2 |        1 | 2025-09-25   | 2025-09-27    | checked_out |
          5 |       8 |        3 | 2025-09-28   | 2025-09-30    | checked_out |
          6 |       5 |        2 | 2025-09-26   | 2025-09-29    | checked_out |
          7 |       1 |        1 | 2025-09-30   | 2025-10-01    | booked      |
          8 |       1 |        1 | 2025-10-01   | 2025-10-03    | checked_in  |
(8 rows)


hotel_db=# CALL book_double_room(1, 2, '2025-10-10'::date, 3, NULL);
 p_booking_id
--------------

(1 row)


hotel_db=# SELECT * FROM Booking;
 booking_id | room_id | guest_id | checkin_date | checkout_date |   status    | promotion_id
------------+---------+----------+--------------+---------------+-------------+--------------
          1 |       2 |        1 | 2025-09-10   | 2025-09-15    | checked_in  |            1
          2 |       3 |        2 | 2025-09-05   | 2025-09-10    | checked_out |            1
          3 |       5 |        3 | 2025-09-12   | 2025-09-14    | booked      |
          4 |       2 |        1 | 2025-09-25   | 2025-09-27    | checked_out |
          5 |       8 |        3 | 2025-09-28   | 2025-09-30    | checked_out |
          6 |       5 |        2 | 2025-09-26   | 2025-09-29    | checked_out |
          7 |       1 |        1 | 2025-09-30   | 2025-10-01    | booked      |
          8 |       1 |        1 | 2025-10-01   | 2025-10-03    | checked_in  |
          9 |       2 |        2 | 2025-10-10   | 2025-10-12    | booked      |
(9 rows)
```

Рис. 4: Результат работы процедуры

2 Создание триггеров

2.1 Триггер 1

Обновление поля is_avaible у комнаты при бронировании:

```
CREATE OR REPLACE FUNCTION trigger_update_is_avaible()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.status IN ('booked','checked_in') THEN
        UPDATE Room SET is_available = FALSE WHERE room_id = NEW.room_id;
    ELSIF NEW.status IN ('checked_out','cancelled') THEN
        UPDATE Room SET is_available = TRUE WHERE room_id = NEW.room_id;
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE TRIGGER trigger_update_is_avaible
AFTER INSERT OR UPDATE OF status ON Booking
FOR EACH ROW
EXECUTE FUNCTION trigger_update_is_avaible();
```

```
hotel_db=# INSERT INTO Booking(room_id, guest_id, checkin_date, checkout_date, status)
hotel_db-# VALUES (1, 1, '2025-10-05', '2025-10-07', 'booked');
INSERT 0 1
hotel_db=# SELECT room_number, is_available FROM Room WHERE room_id = 1;
 room_number | is_available
-------------+--------------
 101         | f
(1 row)
```

Рис. 5: Результат работы триггера

Триггер 2

Вставка в таблицу DailyRoomStatus при бронировании.

```
CREATE OR REPLACE FUNCTION trigger_create_drs()
RETURNS TRIGGER AS $$
DECLARE
    d DATE;
BEGIN
    d := NEW.checkin_date;
    WHILE d <= NEW.checkout_date LOOP
        INSERT INTO DailyRoomStatus(room_id, status_date, status, price_for_date, promotion_id
        VALUES (NEW.room_id, d, 'booked', NULL, NEW.promotion_id)
        ON CONFLICT (room_id, status_date) DO NOTHING;
        d := d + INTERVAL '1 day';
    END LOOP;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_create_drs
AFTER INSERT ON Booking
FOR EACH ROW
EXECUTE FUNCTION trigger_create_drs();
```

```
hotel_db=# \e
 drs_id | room_id | status_date | status | price_for_date | promotion_id
--------+---------+-------------+--------+----------------+--------------
     20 |       1 | 2025-10-05  | booked |                |
     21 |       1 | 2025-10-06  | booked |                |
     22 |       1 | 2025-10-07  | booked |                |
(3 rows)
```

Рис. 6: Результат работы триггера

Триггер 3

Проверка доступности комнаты перед вставкой в расписание уборок:

```
CREATE OR REPLACE FUNCTION trigger_cleaning_date_check()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1 FROM DailyRoomStatus
        WHERE room_id = NEW.room_id
          AND status_date = NEW.cleaning_date
          AND status = 'out_of_service'
    ) THEN
        RAISE EXCEPTION 'Комната недоступна';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_cleaning_date_check
BEFORE INSERT ON CleaningSchedule
FOR EACH ROW
EXECUTE FUNCTION trigger_cleaning_date_check();
```

```
hotel_db=# \e
ОШИБКА:  новая строка в отношении "cleaningschedule" нарушает ограничение-проверку "cleaningschedule_status_check"
DETAIL:  Ошибочная строка содержит (4, 1, 2025-10-06, C?P+C?P°P?, 3).
hotel_db=#
```

Рис. 7: Результат работы триггера

Триггер 4

Вывод данных об оплате в консоль.

```
CREATE OR REPLACE FUNCTION trigger_log_payment()
RETURNS TRIGGER AS $$
BEGIN
    RAISE NOTICE 'Оплата: Booking ID %, Сумма %, Метод %', NEW.booking_id, NEW.amount, NEW.met
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_log_payment
AFTER INSERT ON Payment
FOR EACH ROW
EXECUTE FUNCTION trigger_log_payment();
```

```
hotel_db=# \e
ЗАМЕЧАНИЕ:  P?PiP>P°C'P°: Booking ID 1, РЎС?Р?Р?Р° 5000.00, Р?РчС'Р?Р? card
INSERT 0 1
hotel_db=#
```

Рис. 8: Результат работы триггера

Триггер 5

Обновление предыдущей цены на комнату при вставке новой.

```
CREATE OR REPLACE FUNCTION trigger_update_previous_price()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE RoomPriceHistory
    SET end_date = NEW.start_date - INTERVAL '1 day'
    WHERE room_type_id = NEW.room_type_id
      AND (end_date IS NULL OR end_date >= NEW.start_date)
      AND price_id <> NEW.price_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_previous_price
AFTER INSERT ON RoomPriceHistory
FOR EACH ROW
EXECUTE FUNCTION trigger_update_previous_price();
```

```
hotel_db=# \e
INSERT 0 1
hotel_db=# INSERT INTO RoomPriceHistory(room_type_id, price, start_date)
hotel_db-# VALUES (1, 5500, '2025-10-01');
INSERT 0 1
hotel_db=# SELECT * FROM RoomPriceHistory WHERE room_type_id = 1;
 price_id | room_type_id |  price  | start_date |  end_date
----------+--------------+---------+------------+------------
        1 |            1 | 5512.50 | 2025-01-01 | 2025-09-30
        6 |            1 | 5500.00 | 2025-10-01 |
        5 |            1 | 5500.00 | 2025-10-01 | 2025-09-30
(3 rows)
```

Рис. 9: Результат работы триггера

Триггер 6

Отмена бронирования, если комната занята в текущие даты.

```
CREATE OR REPLACE FUNCTION trigger_check_room_availability()
RETURNS TRIGGER AS $$
BEGIN
    IF EXISTS (
        SELECT 1
        FROM Booking
        WHERE room_id = NEW.room_id
          AND status IN ('booked', 'checked_in')
          -- Проверяем пересечение дат
          AND (
              (NEW.checkin_date BETWEEN checkin_date AND checkout_date) OR
              (NEW.checkout_date BETWEEN checkin_date AND checkout_date) OR
              (NEW.checkin_date <= checkin_date AND NEW.checkout_date >= checkout_date)
          )
    ) THEN
        RAISE EXCEPTION 'Комната занята в данные даты';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_check_room_availability
BEFORE INSERT ON Booking
FOR EACH ROW
EXECUTE FUNCTION trigger_check_room_availability();
```

```
hotel_db=# \e
CREATE FUNCTION
CREATE TRIGGER
hotel_db=# \e
INSERT 0 1
hotel_db=# \e
ОШИБКА:  P?P?P?P?P°C'P° P·P°P?C?C'P° P? P?P°P?P?C<Pч P?P°C'C<
CONTEXT:  функция PL/pgSQL trigger_check_room_availability(), строка 15, оператор RAISE
```

Рис. 10: Результат работы триггера

Триггер 7

Удаление ссылок из зависимых таблиц при удалении акции.

```
CREATE OR REPLACE FUNCTION trigger_update_references()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE Booking
    SET promotion_id = NULL
    WHERE promotion_id = OLD.promotion_id;

    UPDATE DailyRoomStatus
    SET promotion_id = NULL
    WHERE promotion_id = OLD.promotion_id;

    RETURN OLD;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trigger_update_ferences
BEFORE DELETE ON Promotion
FOR EACH ROW
EXECUTE FUNCTION trigger_update_ferences();
```

```
hotel_db=# DELETE FROM Promotion WHERE promotion_id = 1;
ОШИБКА:  UPDATE или DELETE в таблице "promotion" нарушает ограничение внешнего ключа "dailyroomstatus_promotion_id_fkey" таблицы "dail
yroomstatus"
DETAIL:  На ключ (promotion_id)=(1) всё ещё есть ссылки в таблице "dailyroomstatus".
hotel_db=# \e
CREATE FUNCTION
CREATE TRIGGER
hotel_db=# DELETE FROM Promotion WHERE promotion_id = 1;
DELETE 1
```

Рис. 11: Результат работы триггера

Выводы

В ходе лабораторной работы были реализованы три процедуры в соответствии с вариантом 1 БД "Отель а также придуманы и реализованы семь триггеров, позволяющих контролировать работу базы банных.