Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное
учреждение высшего образования
“Национальный исследовательский университет ИТМО”

Факультет инфокоммуникационных технологий

# ЛАБОРАТОРНАЯ РАБОТА №3

## Процедуры, функции, триггеры в PostgreSQL
## по дисциплине:
## «Проектирование и реализация баз данных»

**Выполнил студент:**
Тюмин Никита Сергеевич
Группа №K32402
**Преподаватель:**
Говорова Марина Михайловна

Санкт-Петербург
2023

## Цель работы:

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

## Программное обеспечение:

СУБД PostgreSQL 14, pgAdmin 4.

## Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию
2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

## Ход работы:

Работа проводилась в созданной ранее базе данных, ER диаграмма представлена на Рисунке 1.

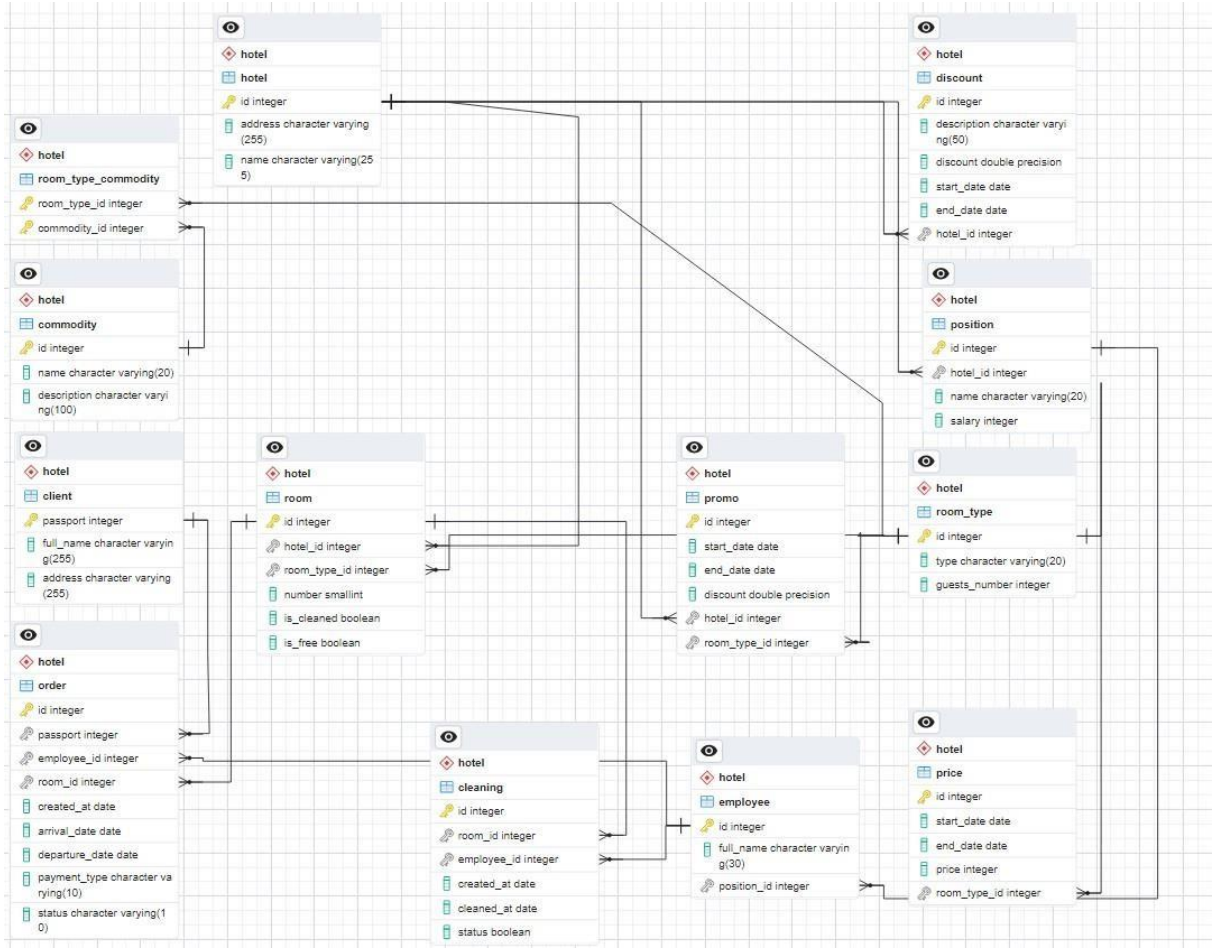
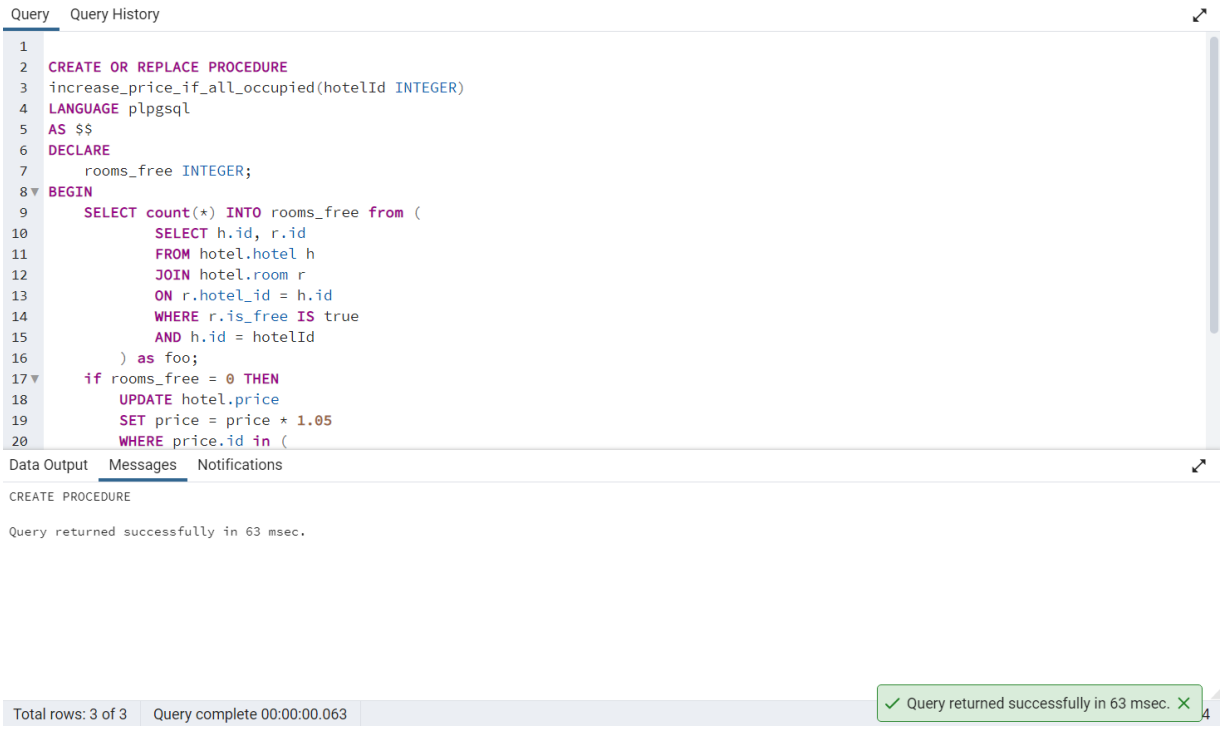



Рисунок 1 – ER диаграмма базы данных

## Создание процедур/функций:

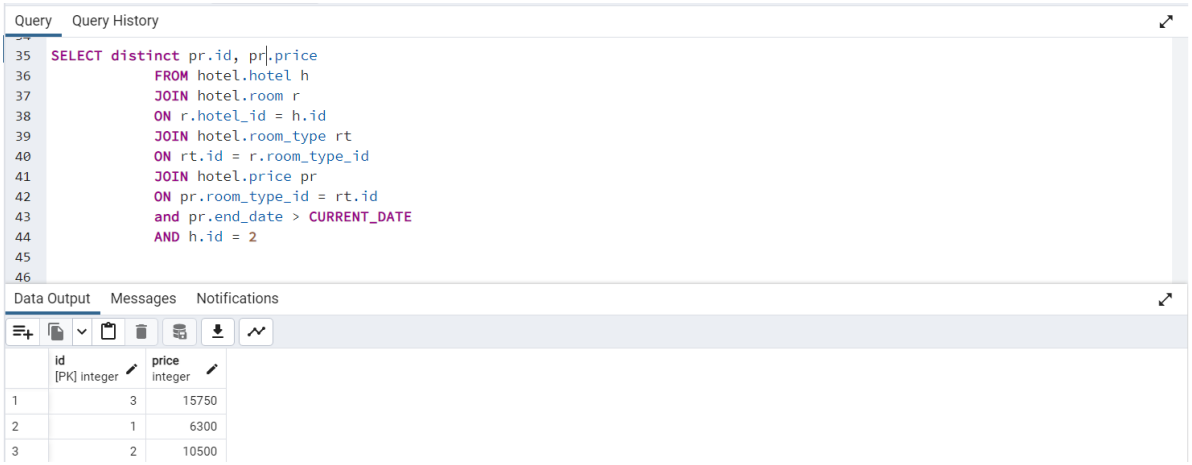
Увеличение цены всех номеров на 5 %, если в отеле нет свободных номеров



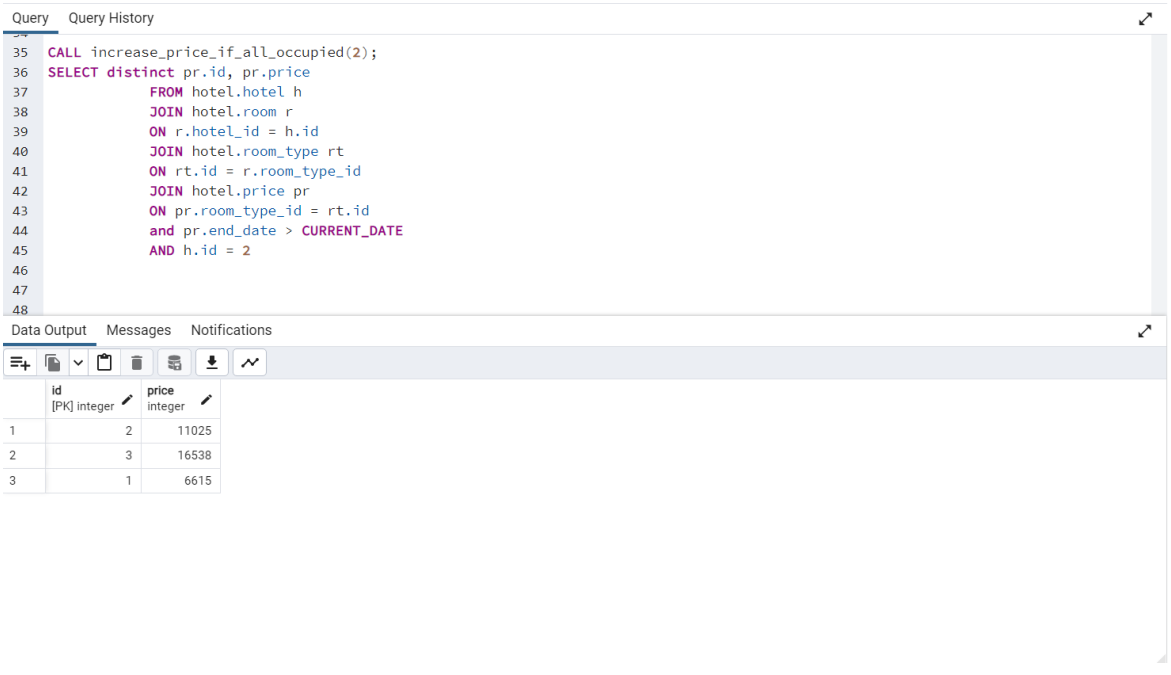

```
CREATE OR REPLACE PROCEDURE
increase_price_if_all_occupied(hotelId INTEGER)
LANGUAGE plpgsql
AS $$
DECLARE
      rooms_free INTEGER;
BEGIN
      SELECT count(*) INTO rooms_free from (
                  SELECT h.id, r.id
                  FROM hotel.hotel h
                  JOIN hotel.room r
                  ON r.hotel_id = h.id
                  WHERE r.is_free IS true
                  AND h.id = hotelId
            ) as foo;
      if rooms_free = 0 THEN
            UPDATE hotel.price
            SET price = price * 1.05
            WHERE price.id in (
                  SELECT distinct pr.id
                  FROM hotel.hotel h
                  JOIN hotel.room r
                  ON r.hotel_id = h.id
                  JOIN hotel.room_type rt
                  ON rt.id = r.room_type_id
                  JOIN hotel.price pr
```

```
                ON pr.room_type_id = rt.id
                and pr.end_date > CURRENT_DATE
                AND h.id = hotelId
        );
    END IF;
end;
$$;
```

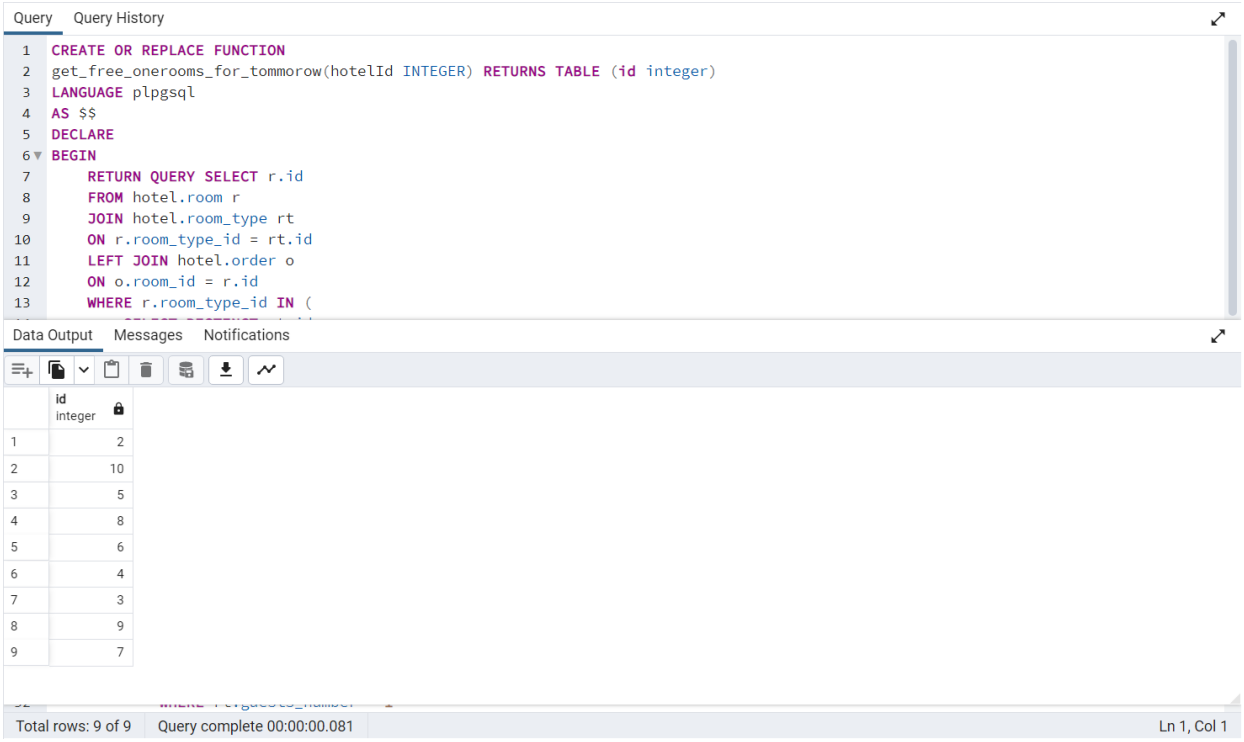
До запроса:




После:




Получение информации о свободных одноместных номерах отеля на завтрашний день;

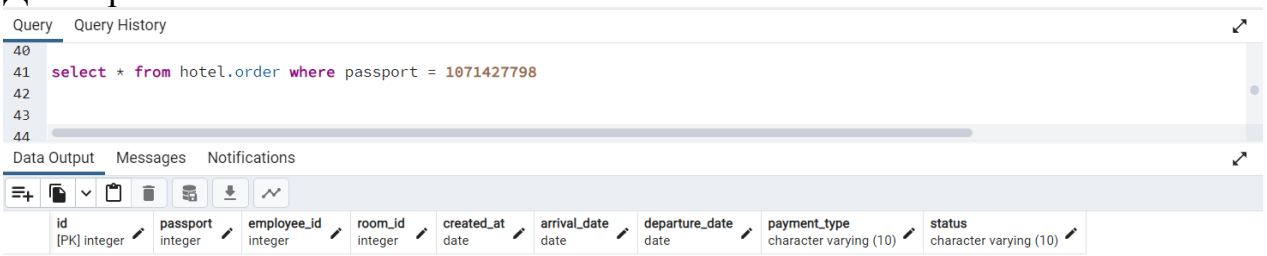



```
CREATE OR REPLACE FUNCTION
get_free_onerooms_for_tommorow(hotelId INTEGER) RETURNS TABLE (id integer)
LANGUAGE plpgsql
AS $$
DECLARE
BEGIN
	RETURN QUERY SELECT r.id
	FROM hotel.room r
	JOIN hotel.room_type rt
	ON r.room_type_id = rt.id
	LEFT JOIN hotel.order o
	ON o.room_id = r.id
	WHERE r.room_type_id IN (
		SELECT DISTINCT rt.id
		FROM hotel.room r
		JOIN hotel.room_type rt
		ON r.room_type_id = rt.id
		WHERE rt.guests_number = 1
	)
	AND r.id NOT IN (
		SELECT r.id
		FROM hotel.room r
		JOIN hotel.room_type rt
		ON r.room_type_id = rt.id
		LEFT JOIN hotel.order o
		ON o.room_id = r.id
		WHERE r.room_type_id IN (
			SELECT DISTINCT rt.id
```

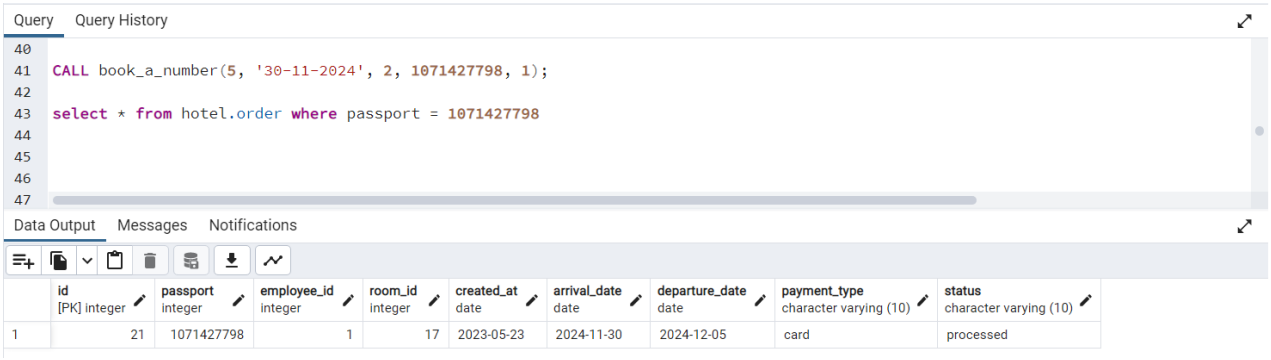
```
                FROM hotel.room r
                JOIN hotel.room_type rt
                ON r.room_type_id = rt.id
                WHERE rt.guests_number = 1
            )
            AND r.hotel_id = hotelId
            AND CURRENT_DATE + interval '1 day' >= o.arrival_date
            AND CURRENT_DATE + interval '1 day' <= o.departure_date
      )
      AND r.hotel_id = hotelId;
END;
$$;

select * from get_free_onerooms_for_tommorow(2);
```

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

До запроса:




После запроса:




```
CREATE OR REPLACE PROCEDURE
book_a_number(daysNumber INTEGER, arrivalDate DATE, hotelId INTEGER, guestId
INTEGER, empId INTEGER)
LANGUAGE plpgsql
AS $$
DECLARE
      roomId INTEGER;
      departureDate DATE;
BEGIN
CREATE TEMP TABLE bookedRooms ON COMMIT DROP AS
      SELECT DISTINCT o.room_id
      FROM hotel.order o
```

```
     WHERE o.arrival_date BETWEEN arrivalDate AND (arrivalDate::date +
(daysNumber || ' day')::interval)
     OR o.departure_date BETWEEN arrivalDate AND (arrivalDate::date +
(daysNumber || ' day')::interval)
     OR (arrivalDate BETWEEN o.arrival_date AND o.departure_date AND
(arrivalDate::date + (daysNumber || ' day')::interval) BETWEEN o.arrival_date
AND o.departure_date);


     SELECT r.id into roomId
     FROM hotel.hotel h
     JOIN hotel.room r
     ON h.id = r.hotel_id
     JOIN hotel.room_type rt
     ON r.room_type_id = rt.id
     WHERE rt.guests_number = 2
     AND r.hotel_id = hotelId
     AND r.id NOT IN (SELECT * FROM bookedRooms)
     LIMIT 1;

     IF roomId IS NOT null THEN
          INSERT INTO hotel.order (passport, employee_id, room_id, created_at,
arrival_date, departure_date, payment_type, status)
          VALUES
          (guestId, empId, roomId, CURRENT_DATE, arrivalDate, arrivalDate +
(daysNumber::text || ' day')::INTERVAL, 'card', 'processed');
     END IF;

     END;
$$;
```

## Создание триггера для логирования вставки/изменения/удаления записей в таблице заказов.

Создание триггера:


```
CREATE TABLE IF NOT EXISTS order_logs (
      created_at DATE,
      "message" TEXT
);

CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
DECLARE
    msg varchar(254);
BEGIN
    IF    TG_OP = 'INSERT' THEN
           msg := 'New order (' || NEW.id || ') from ' || NEW.passport;
        INSERT INTO order_logs(created_at, "message") values (CURRENT_DATE,
msg);
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
           msg := 'Order (' || NEW.id || ') updated';
        INSERT INTO order_logs(created_at, "message") values (CURRENT_DATE,
msg);
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
           msg := 'Deleted order (' || NEW.id || ')';
        INSERT INTO order_logs(created_at, "message") values (CURRENT_DATE,
msg);
        RETURN OLD;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER t_order AFTER INSERT OR UPDATE OR DELETE ON hotel.order FOR EACH
ROW EXECUTE PROCEDURE add_to_log();
```

Таблица с логами после вставки:


```
INSERT INTO hotel.order (passport, employee_id, room_id, created_at,
arrival_date, departure_date, payment_type, status) VALUES
(1167417325, 2, 16, CURRENT_DATE, CURRENT_DATE, CURRENT_DATE + interval '7
day', 'cash', 'processed');

UPDATE hotel.order SET status = 'cancelled' WHERE id = 20 ;

DELETE FROM hotel.order WHERE id = 20;

SELECT * FROM public.order_logs
```

## Выводы:

1. Были созданы процедуры и функции согласно индивидуальному заданию
2. Был создан триггер на различные операции со строками.