Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №5 «Процедуры, функции, триггеры в PostgreSQL»

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Ковалев Г. П.

Факультет: ПИН

Группа: К3241

Преподаватель: Говорова М.М.

Санкт-Петербург 2025

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 триггеры для индивидуальной БД согласно варианту: 7 оригинальных триггеров - 7 баллов (max).

Вариант 1 БД «Отель»

Задание 4. Создайте хранимые процедуры:

- для увеличения цены всех номеров на 5 %, если в отеле нет свободных номеров;
- для получения информации о свободных одноместных номерах отеля на завтрашний день;
- бронирования двухместного номера в гостинице на заданную дату и количество дней проживания.

Процесс выполнения задания:

1) Процедуры

- для увеличения цены всех номеров на 5 %, если в отеле нет свободных номеров;

До:

```
newRandom=# SELECT * FROM "RoomInHotel";
 id_number | id_hotel | id_type | room_number | places_number |   status
-----------+----------+---------+-------------+---------------+------------
         1 |        4 |       2 |         101 |             2 | {Свободен}
         2 |        4 |       1 |         102 |             1 | {Занят}
         3 |        5 |       3 |         201 |             3 | {Свободен}
         4 |        5 |       2 |         202 |             2 | {Занят}
         5 |        6 |       1 |         301 |             1 | {Свободен}
         6 |        4 |       2 |         104 |             2 | {Свободен}
         7 |        6 |       1 |         302 |             1 | {Свободен}
         8 |        6 |       1 |         303 |             1 | {Свободен}
         9 |        5 |       1 |         203 |             1 | {Свободен}
        10 |        6 |       1 |         304 |             1 | {Свободен}
        11 |        5 |       1 |         204 |             1 | {Свободен}
        12 |        4 |       1 |         105 |             1 | {Свободен}
        13 |        6 |       1 |         305 |             1 | {Свободен}
        14 |        5 |       1 |         205 |             1 | {Свободен}
        15 |        7 |       1 |           1 |             1 | {Свободен}
        16 |        7 |       1 |           2 |             1 | {Свободен}
        17 |        7 |       1 |           3 |             1 | {Свободен}
        18 |        7 |       1 |           4 |             1 | {Свободен}
        19 |        4 |       1 |         106 |             1 | {Свободен}
        20 |        7 |       1 |           5 |             1 | {Свободен}
(20 rows)

newRandom=# SELECT * FROM "TypeOfRoom";
 id_type |     name      | num_of_rooms | num_of_places | conveniences  | base_price | num_of_free_rooms
---------+---------------+--------------+---------------+---------------+------------+-------------------
       1 | {Одноместный} |           20 |             1 | {1,2,3}       |       3000 |                10
       2 | {Двухместный} |           30 |             2 | {1,2,4,5}     |       5000 |                15
       3 | {Люкс}        |           10 |             3 | {1,2,3,4,5,6} |       9000 |                 4
(3 rows)
```

Процедура:

```
CREATE OR REPLACE PROCEDURE increase_prices_if_no_free_rooms()
LANGUAGE plpgsql
AS $$
BEGIN
   IF NOT EXISTS (
      SELECT 1
      FROM public."RoomInHotel"
      WHERE status[1] = 'Свободен'
   ) THEN
      UPDATE public."TypeOfRoom"
      SET base_price = ROUND(base_price * 1.05);
   END IF;
END;
$$;
```

После:

```
newRandom=# SELECT * FROM "TypeOfRoom";
 id_type |     name      | num_of_rooms | num_of_places | conveniences  | base_price | num_of_free_rooms
---------+---------------+--------------+---------------+---------------+------------+-------------------
       1 | {Одноместный} |           20 |             1 | {1,2,3}       |       3000 |                10
       2 | {Двухместный} |           30 |             2 | {1,2,4,5}     |       5000 |                15
       3 | {Люкс}        |           10 |             3 | {1,2,3,4,5,6} |       9000 |                 4
(3 rows)
```

```
 id_number | id_hotel | id_type | room_number | places_number | status
-----------+----------+---------+-------------+---------------+---------
         3 |        5 |       3 |         201 |             3 | {Занят}
         4 |        5 |       2 |         202 |             2 | {Занят}
         7 |        6 |       1 |         302 |             1 | {Занят}
         8 |        6 |       1 |         303 |             1 | {Занят}
         9 |        5 |       1 |         203 |             1 | {Занят}
        10 |        6 |       1 |         304 |             1 | {Занят}
        11 |        5 |       1 |         204 |             1 | {Занят}
        13 |        6 |       1 |         305 |             1 | {Занят}
        14 |        5 |       1 |         205 |             1 | {Занят}
        15 |        7 |       1 |           1 |             1 | {Занят}
        16 |        7 |       1 |           2 |             1 | {Занят}
        17 |        7 |       1 |           3 |             1 | {Занят}
        18 |        7 |       1 |           4 |             1 | {Занят}
        20 |        7 |       1 |           5 |             1 | {Занят}
         5 |        6 |       1 |         301 |             1 | {Занят}
         2 |        4 |       1 |         102 |             1 | {Занят}
         6 |        4 |       2 |         104 |             2 | {Занят}
        12 |        4 |       1 |         105 |             1 | {Занят}
        19 |        4 |       1 |         106 |             1 | {Занят}
         1 |        4 |       2 |         101 |             2 | {Занят}
(20 rows)

newRandom=# CALL increase_prices_if_no_free_rooms();
CALL
newRandom=# SELECT * FROM "TypeOfRoom";
 id_type |     name      | num_of_rooms | num_of_places | conveniences  | base_price | num_of_free_rooms
---------+---------------+--------------+---------------+---------------+------------+-------------------
       1 | {Одноместный} |           20 |             1 | {1,2,3}       |       3150 |                10
       2 | {Двухместный} |           30 |             2 | {1,2,4,5}     |       5250 |                15
       3 | {Люкс}        |           10 |             3 | {1,2,3,4,5,6} |       9450 |                 4
(3 rows)
```

- для получения информации о свободных одноместных номерах отеля на завтрашний день

Процедура:

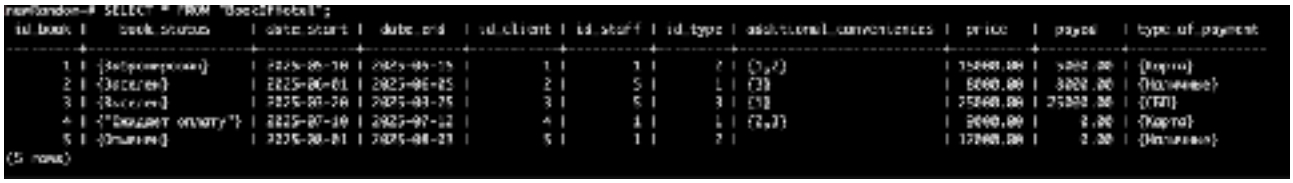
```
CREATE OR REPLACE PROCEDURE get_free_single_rooms_tomorrow(
   IN input_hotel_id INTEGER,
   INOUT result REFCURSOR
)
LANGUAGE plpgsql
AS $$
BEGIN
   OPEN result FOR
   SELECT room.id_number, room.room_number, room.places_number, room.status[1] AS status
   FROM public."RoomInHotel" room
   WHERE room.id_hotel = input_hotel_id
    AND room.places_number = 1
    AND room.id_number NOT IN (
       SELECT checkIN.id_room
       FROM public."CheckIn" checkIN
       WHERE CURRENT_DATE + 1 BETWEEN checkIN.date_check_in AND
checkIN.date_check_out
    )
          AND room.id_type NOT IN (
       SELECT book.id_type
       FROM public."BookOfHotel" book
       WHERE CURRENT_DATE + 1 BETWEEN book.date_start AND book.date_end
    );
END;
$$;
```

Результат + вызов:

```
newRandom=# BEGIN;
CALL get_free_single_rooms_tomorrow(4, 'rooms_tomorrow');
FETCH ALL FROM rooms_tomorrow;
COMMIT;
BEGIN
     result
---------------
 rooms_tomorrow
(1 row)

 id_number | room_number | places_number |  status
-----------+-------------+---------------+----------
         2 |         102 |             1 | Занят
        12 |         105 |             1 | Свободен
        19 |         106 |             1 | Свободен
(3 rows)

COMMIT
```

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

До:



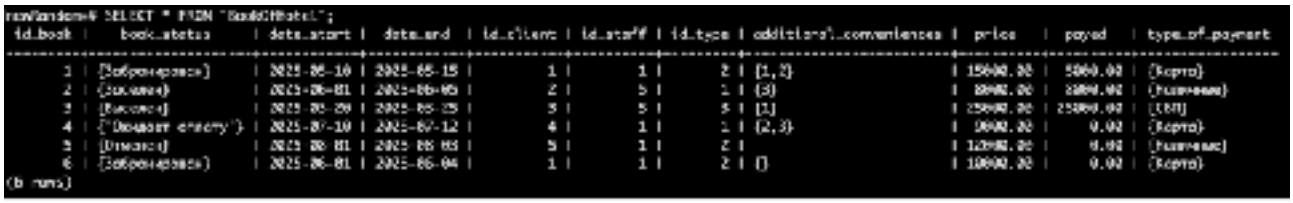
Процедура:

```
CREATE OR REPLACE PROCEDURE book_double_room(
   IN input_hotel_id INTEGER,
   IN input_client_id INTEGER,
   IN input_date_start DATE,
   IN input_days INTEGER,
   OUT success_code INTEGER
)
LANGUAGE plpgsql
AS $$
```

```
DECLARE
   booking_end DATE;
   room_id INTEGER;
BEGIN
   booking_end := input_date_start + input_days;

   SELECT room.id_number
         INTO room_id
         FROM public."RoomInHotel" room
         WHERE room.id_hotel = input_hotel_id
         AND room.places_number = 2
         AND NOT EXISTS (
     SELECT 1 FROM public."CheckIn" checkIn
     WHERE checkIn.id_room = room.id_number
      AND (input_date_start, booking_end) OVERLAPS (checkIn.date_check_in,
checkIn.date_check_out)
         )
         AND (
     SELECT COUNT(*)
     FROM public."BookOfHotel" book
     WHERE book.id_type = room.id_type
      AND (input_date_start, booking_end) OVERLAPS (book.date_start, book.date_end)
         ) < (
     SELECT COUNT(*)
     FROM public."RoomInHotel"
     WHERE id_hotel = input_hotel_id AND id_type = room.id_type
         )
         LIMIT 1;

   IF room_id IS NULL THEN
      success_code := 0;
      RETURN;
   END IF;

   INSERT INTO public."BookOfHotel" (
      id_book,
      book_status,
      date_start,
      date_end,
      id_client,
      id_staff,
      id_type,
      additional_conveniences,
      price,
      payed,
      type_of_payment
   ) VALUES (
      (SELECT COALESCE(MAX(id_book), 0) + 1 FROM public."BookOfHotel"),
```

```
    ARRAY['Забронирован'],
    input_date_start,
    booking_end,
    input_client_id,
    1,
    2,
    ARRAY[]::INTEGER[],
    10000,
    0,
    ARRAY['Карта']
  );

  success_code := 1;
END;
$$;
```

После:



2) Триггеры

- Создать триггер, который будет увеличивать количество номеров в таблице «Hotel», после того, как мы добавили номер в «RoomsInHotel»

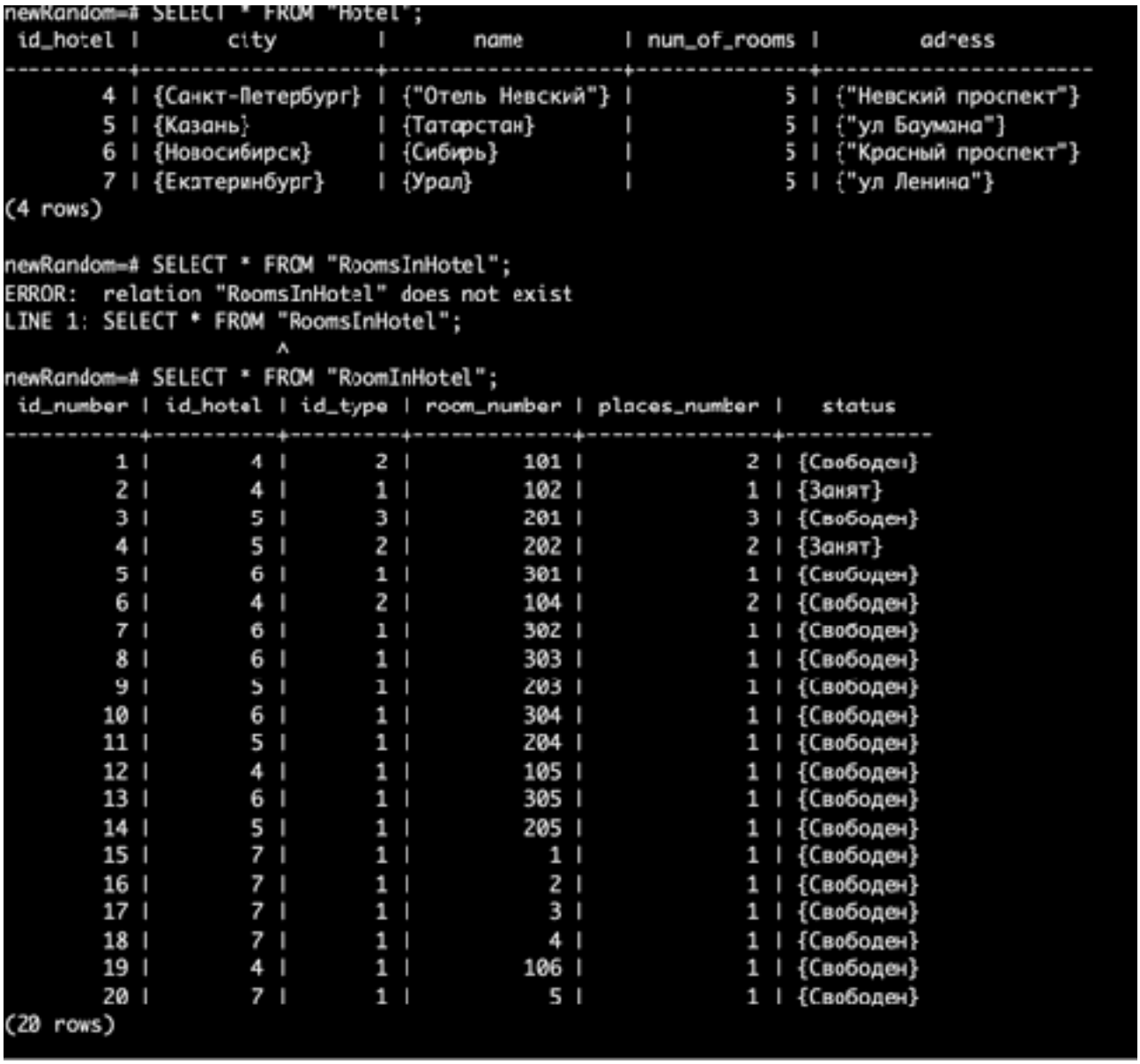
Функция:

```
CREATE OR REPLACE FUNCTION update_num_of_rooms_after_insert()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
  UPDATE public."Hotel"
  SET num_of_rooms = num_of_rooms + 1
  WHERE id_hotel = NEW.id_hotel;

  RETURN NEW;
END;
$$;
```

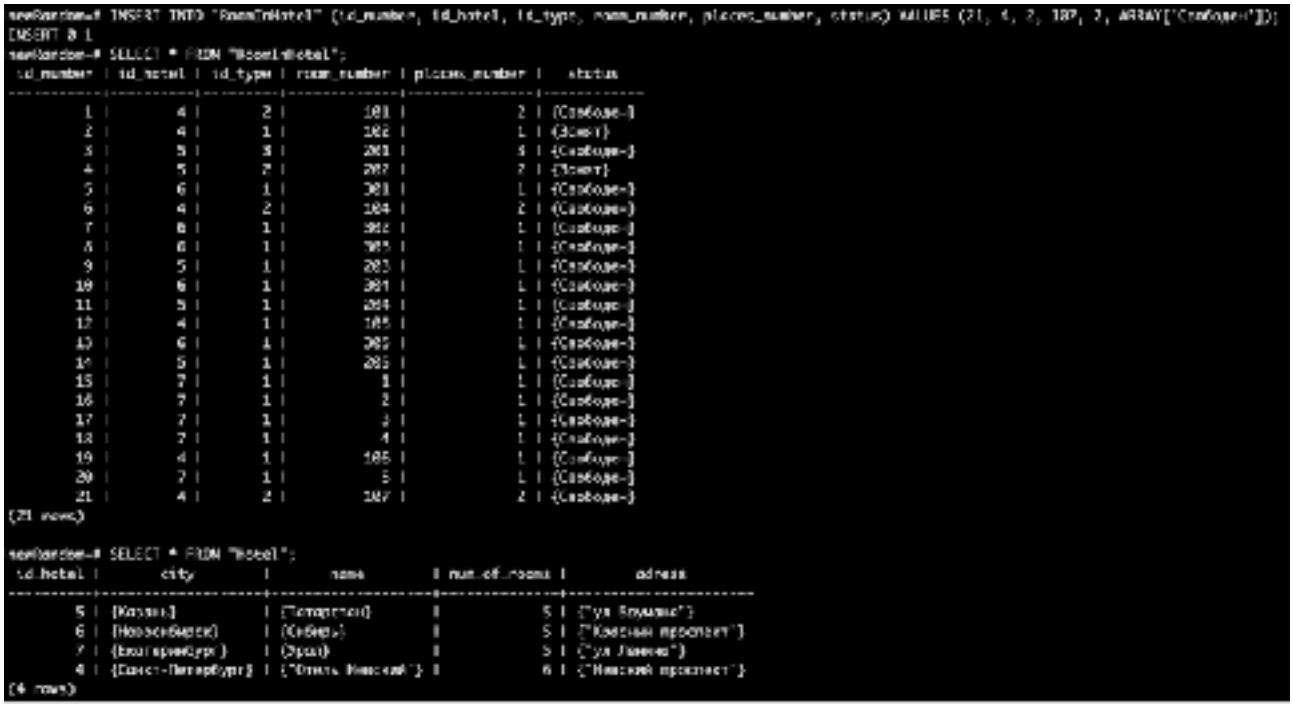
Триггер:

```
CREATE TRIGGER trg_increment_room_count
```

AFTER INSERT ON public."RoomInHotel"
FOR EACH ROW
EXECUTE FUNCTION update_num_of_rooms_after_insert();



```
newRandom=# SELECT * FROM "Hotel";
 id_hotel |       city        |       name        | num_of_rooms |        adress
----------+-------------------+-------------------+--------------+----------------------
        4 | {Санкт-Петербург} | {"Отель Невский"} |            5 | {"Невский проспект"}
        5 | {Казань}          | {Татарстан}       |            5 | {"ул Баумана"]
        6 | {Новосибирск}     | {Сибирь}          |            5 | {"Красный проспект"}
        7 | {Екатеринбург}    | {Урал}            |            5 | {"ул Ленина"}
(4 rows)

newRandom=# SELECT * FROM "RoomsInHotel";
ERROR:  relation "RoomsInHotel" does not exist
LINE 1: SELECT * FROM "RoomsInHotel";
                      ^
newRandom=# SELECT * FROM "RoomInHotel";
 id_number | id_hotel | id_type | room_number | places_number |   status
-----------+----------+---------+-------------+---------------+------------
         1 |        4 |       2 |         101 |             2 | {Свободен}
         2 |        4 |       1 |         102 |             1 | {Занят}
         3 |        5 |       3 |         201 |             3 | {Свободен}
         4 |        5 |       2 |         202 |             2 | {Занят}
         5 |        6 |       1 |         301 |             1 | {Свободен}
         6 |        4 |       2 |         104 |             2 | {Свободен}
         7 |        6 |       1 |         302 |             1 | {Свободен}
         8 |        6 |       1 |         303 |             1 | {Свободен}
         9 |        5 |       1 |         203 |             1 | {Свободен}
        10 |        6 |       1 |         304 |             1 | {Свободен}
        11 |        5 |       1 |         204 |             1 | {Свободен}
        12 |        4 |       1 |         105 |             1 | {Свободен}
        13 |        6 |       1 |         305 |             1 | {Свободен}
        14 |        5 |       1 |         205 |             1 | {Свободен}
        15 |        7 |       1 |           1 |             1 | {Свободен}
        16 |        7 |       1 |           2 |             1 | {Свободен}
        17 |        7 |       1 |           3 |             1 | {Свободен}
        18 |        7 |       1 |           4 |             1 | {Свободен}
        19 |        4 |       1 |         106 |             1 | {Свободен}
        20 |        7 |       1 |           5 |             1 | {Свободен}
(20 rows)
```



- Создать триггер, который будет уменьшать количество номеров в таблице «Hotel», после того, как мы удалили номер из «RoomsInHotel»

Функция:

```
CREATE OR REPLACE FUNCTION decrease_num_of_rooms_after_delete()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE public."Hotel"
    SET num_of_rooms = num_of_rooms - 1
    WHERE id_hotel = OLD.id_hotel;

    RETURN OLD;
END;
$$;
```

Триггер:

```
CREATE TRIGGER trg_decrement_room_count
AFTER DELETE ON public."RoomInHotel"
FOR EACH ROW
EXECUTE FUNCTION decrease_num_of_rooms_after_delete();
```

```
newRandom=# DELETE FROM public."RoomInHotel" WHERE id_number = 21;
DELETE 1
newRandom=# SELECT * FROM "RoomInHotel";
 id_number | id_hotel | id_type | roon_nunber | places_nunber |   status
-----------+----------+---------+-------------+---------------+------------
         1 |        4 |       2 |         101 |             2 | {Свободен}
         2 |        4 |       1 |         102 |             1 | {Занят}
         3 |        5 |       3 |         201 |             3 | {Свободен}
         4 |        5 |       2 |         202 |             2 | {Занят}
         5 |        6 |       1 |         301 |             1 | {Свободен}
         6 |        4 |       2 |         104 |             2 | {Свободен}
         7 |        6 |       1 |         302 |             1 | {Свободен}
         8 |        6 |       1 |         303 |             1 | {Свободен}
         9 |        5 |       1 |         203 |             1 | {Свободен}
        10 |        6 |       1 |         304 |             1 | {Свободен}
        11 |        5 |       1 |         204 |             1 | {Свободен}
        12 |        4 |       1 |         105 |             1 | {Свободен}
        13 |        6 |       1 |         305 |             1 | {Свободен}
        14 |        5 |       1 |         205 |             1 | {Свободен}
        15 |        7 |       1 |           1 |             1 | {Свободен}
        16 |        7 |       1 |           2 |             1 | {Свободен}
        17 |        7 |       1 |           3 |             1 | {Свободен}
        18 |        7 |       1 |           4 |             1 | {Свободен}
        19 |        4 |       1 |         106 |             1 | {Свободен}
        20 |        7 |       1 |           5 |             1 | {Свободен}
(20 rows)

newRandom=# SELECT * FROM "Hotel";
 id_hotel |       city        |       name        | num_of_rooms |        adress
----------+-------------------+-------------------+--------------+-----------------------
        5 | {Казань}          | {Татарстан}       |            5 | {"ул Баумана"}
        6 | {Новосибирск}     | {Сибирь}          |            5 | {"Красный проспект"}
        7 | {Екатеринбург}    | {Урал}            |            5 | {"ул Ленина"}
        4 | {Санкт-Петербург} | {"Отель Невский"} |            5 | {"Невский проспект"}
(4 rows)
```

- При добавлении данных в Check-In обновляется информация в «RoomInHotel» по занятости номера

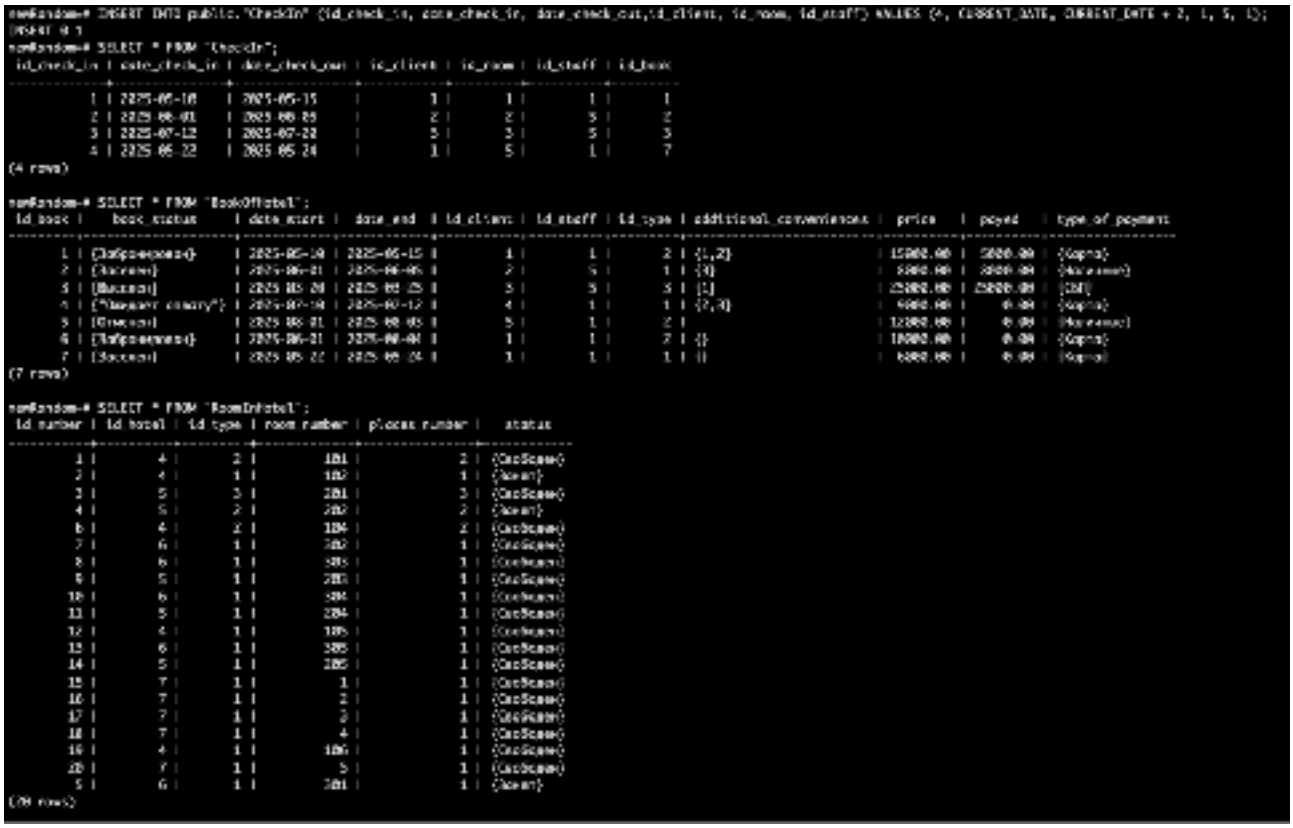
Функция:

```
CREATE OR REPLACE FUNCTION set_room_occupied_if_today_checkin()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
   IF CURRENT_DATE BETWEEN NEW.date_check_in AND NEW.date_check_out THEN
      UPDATE public."RoomInHotel"
      SET status = ARRAY['Занят']
      WHERE id_number = NEW.id_room;
   END IF;
```

```
    RETURN NEW;
END;
$$;
```

Триггер:

```
CREATE TRIGGER trg_set_room_occupied_if_today
AFTER INSERT ON public."CheckIn"
FOR EACH ROW
EXECUTE FUNCTION set_room_occupied_if_today_checkin();
```



- Если мы создаем CheckIn с id_book null, то мы автоматически создаем еще и book

Функция:

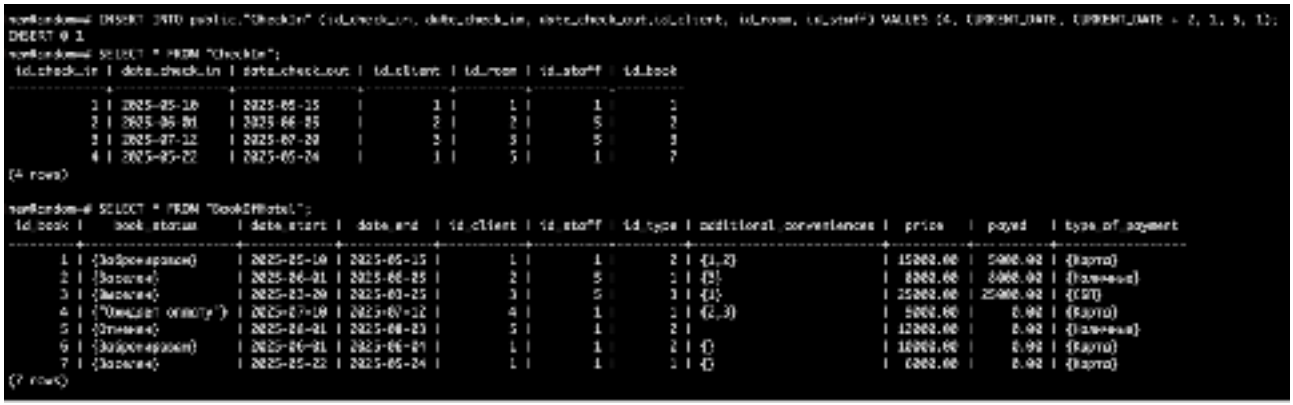
```
CREATE OR REPLACE FUNCTION auto_create_book_if_null()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
DECLARE
    new_book_id INTEGER;
    room_type_id INTEGER;
    base_price NUMERIC;
```

```
    days_count INTEGER;
    total_price NUMERIC;
BEGIN
    SELECT room.id_type INTO room_type_id
    FROM public."RoomInHotel" room
    WHERE room.id_number = NEW.id_room;

    SELECT typeRoom.base_price INTO base_price
    FROM public."TypeOfRoom" typeRoom
    WHERE typeRoom.id_type = room_type_id;

    days_count := GREATEST((NEW.date_check_out - NEW.date_check_in), 1);
    total_price := ROUND(base_price * days_count, 2);

    IF NEW.id_book IS NULL THEN
        SELECT COALESCE(MAX(id_book), 0) + 1 INTO new_book_id FROM
public."BookOfHotel";

        INSERT INTO public."BookOfHotel" (
            id_book,
            book_status,
            date_start,
            date_end,
            id_client,
            id_staff,
            id_type,
            additional_conveniences,
            price,
            payed,
            type_of_payment
        ) VALUES (
            new_book_id,
            ARRAY['Заселен'],
            NEW.date_check_in,
            NEW.date_check_out,
            NEW.id_client,
            NEW.id_staff,
            room_type_id,
            ARRAY[]::INTEGER[],
            total_price,
            0,
            ARRAY['Карта']
        );

        NEW.id_book := new_book_id;
    END IF;

    RETURN NEW;
```

END;
$$;

Триггер:

CREATE TRIGGER trg_auto_create_book
BEFORE INSERT ON public."CheckIn"
FOR EACH ROW
EXECUTE FUNCTION auto_create_book_if_null();



- Если CheckIn обновляется и новая дата выселения <= текущей даты, мы обновляем статус в «RoomInHotel»

Функция:

CREATE OR REPLACE FUNCTION free_room_after_checkout()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
  IF NEW.date_check_out <= CURRENT_DATE THEN
    UPDATE public."RoomInHotel"
    SET status = ARRAY['Свободен']
    WHERE id_number = NEW.id_room;
  END IF;

  RETURN NEW;
END;
$$;

Триггер:

CREATE TRIGGER trg_free_room_after_checkout
AFTER UPDATE ON public."CheckIn"
FOR EACH ROW

WHEN (NEW.date_check_out IS DISTINCT FROM OLD.date_check_out)
EXECUTE FUNCTION free_room_after_checkout();

До:

```
newRandom=# SELECT * FROM "RoomInHotel";
 id_number | id_hotel | id_type | room_number | places_number |   status
-----------+----------+---------+-------------+---------------+------------
         1 |        4 |       2 |         101 |             2 | {Свободен}
         2 |        4 |       1 |         102 |             1 | {Занят}
         3 |        5 |       3 |         201 |             3 | {Свободен}
         4 |        5 |       2 |         202 |             2 | {Занят}
         6 |        4 |       2 |         104 |             2 | {Свободен}
         7 |        6 |       1 |         302 |             1 | {Свободен}
         8 |        6 |       1 |         303 |             1 | {Свободен}
         9 |        5 |       1 |         203 |             1 | {Свободен}
        10 |        6 |       1 |         304 |             1 | {Свободен}
        11 |        5 |       1 |         204 |             1 | {Свободен}
        12 |        4 |       1 |         105 |             1 | {Свободен}
        13 |        6 |       1 |         305 |             1 | {Свободен}
        14 |        5 |       1 |         205 |             1 | {Свободен}
        15 |        7 |       1 |           1 |             1 | {Свободен}
        16 |        7 |       1 |           2 |             1 | {Свободен}
        17 |        7 |       1 |           3 |             1 | {Свободен}
        18 |        7 |       1 |           4 |             1 | {Свободен}
        19 |        4 |       1 |         106 |             1 | {Свободен}
        20 |        7 |       1 |           5 |             1 | {Свободен}
         5 |        6 |       1 |         301 |             1 | {Занят}
(20 rows)
```

После:

```
newRandom=# UPDATE public."CheckIn"
SET date_check_out = CURRENT_DATE
WHERE id_room = 5;
UPDATE 1
newRandom=# SELECT id_number, status FROM public."RoomInHotel" WHERE id_number = 5;
 id_number |   status
-----------+------------
         5 | {Свободен}
(1 row)
```

- Предотвращает удаление номера в «RoomInHotel», если он уже использовался в истории заселений — чтобы не нарушить данные о проживании.

Функция:

```
CREATE OR REPLACE FUNCTION prevent_room_delete_if_checked_in()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
   IF EXISTS (
      SELECT 1 FROM public."CheckIn"
      WHERE id_room = OLD.id_number AND date_check_out > CURRENT_DATE
   ) THEN
      RAISE EXCEPTION 'You cant delete number % — its using in CheckIn.', OLD.id_number;
   END IF;

   RETURN OLD;
END;
$$;
```

Триггер:

```
CREATE TRIGGER trg_prevent_used_room_delete
BEFORE DELETE ON public."RoomInHotel"
FOR EACH ROW
EXECUTE FUNCTION prevent_room_delete_if_checked_in();
```

```
newRandon=# SELECT * FROM public."CheckIn" WHERE id_room = 2;
 id_check_in | date_check_in | date_check_out | id_client | id_room | id_staff | id_book
-------------+---------------+----------------+-----------+---------+----------+---------
           2 | 2025-06-01    | 2025-06-05     |         2 |       2 |        5 |       2
(1 row)

newRandon=# DELETE FROM public."RoomInHotel" WHERE id_number = 2;
ERROR:  You cant delete number 2 - its using in CheckIn.
CONTEXT:  PL/pgSQL function prevent_room_delete_if_checked_in() line 7 at RAISE
newRandon=#
```

- Если клиент заселяется, то статус бронирования становится «Заселен»

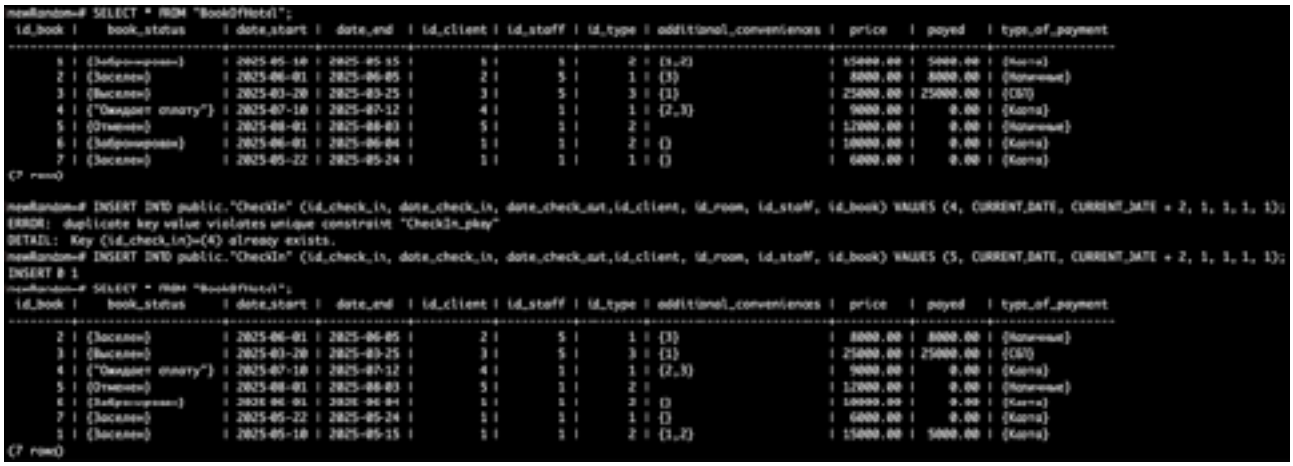
Функция:

```
CREATE OR REPLACE FUNCTION mark_booking_as_checked_in()
RETURNS TRIGGER
LANGUAGE plpgsql
```

```
AS $$
BEGIN
    IF NEW.id_book IS NOT NULL THEN
        UPDATE public."BookOfHotel"
        SET book_status = ARRAY['Заселен']
        WHERE id_book = NEW.id_book;
    END IF;

    RETURN NEW;
END;
$$;
```

Триггер:

```
CREATE TRIGGER trg_mark_booking_as_checked_in
AFTER INSERT ON public."CheckIn"
FOR EACH ROW
EXECUTE FUNCTION mark_booking_as_checked_in();
```



Выводы:

В ходе выполнения лабораторной работы были разработаны и протестированы процедуры, функции и триггеры для автоматизации логики работы гостиничной базы данных. Реализована проверка бизнес-правил, автоматическое обновление связанных данных и защита от некорректных операций. Полученные навыки позволяют эффективно управлять целостностью и поведением данных в PostgreSQL.