

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИТМО»**

Факультет инфокоммуникационных технологий

Дисциплина:

«Проектирование и реализация баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«Процедуры, функции, триггеры в PostgreSQL»

Выполнил:

студент группы К32392

Бочкарь Артём Артёмович

(подпись)

Проверил(а):

Говорова Марина Михайловна

(отметка о выполнении)

(подпись)

Санкт-Петербург
2023 г.

Цель работы: овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

Вариант 1

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
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.

Выполнение

Наименование БД: hotel

ERD диаграмма:

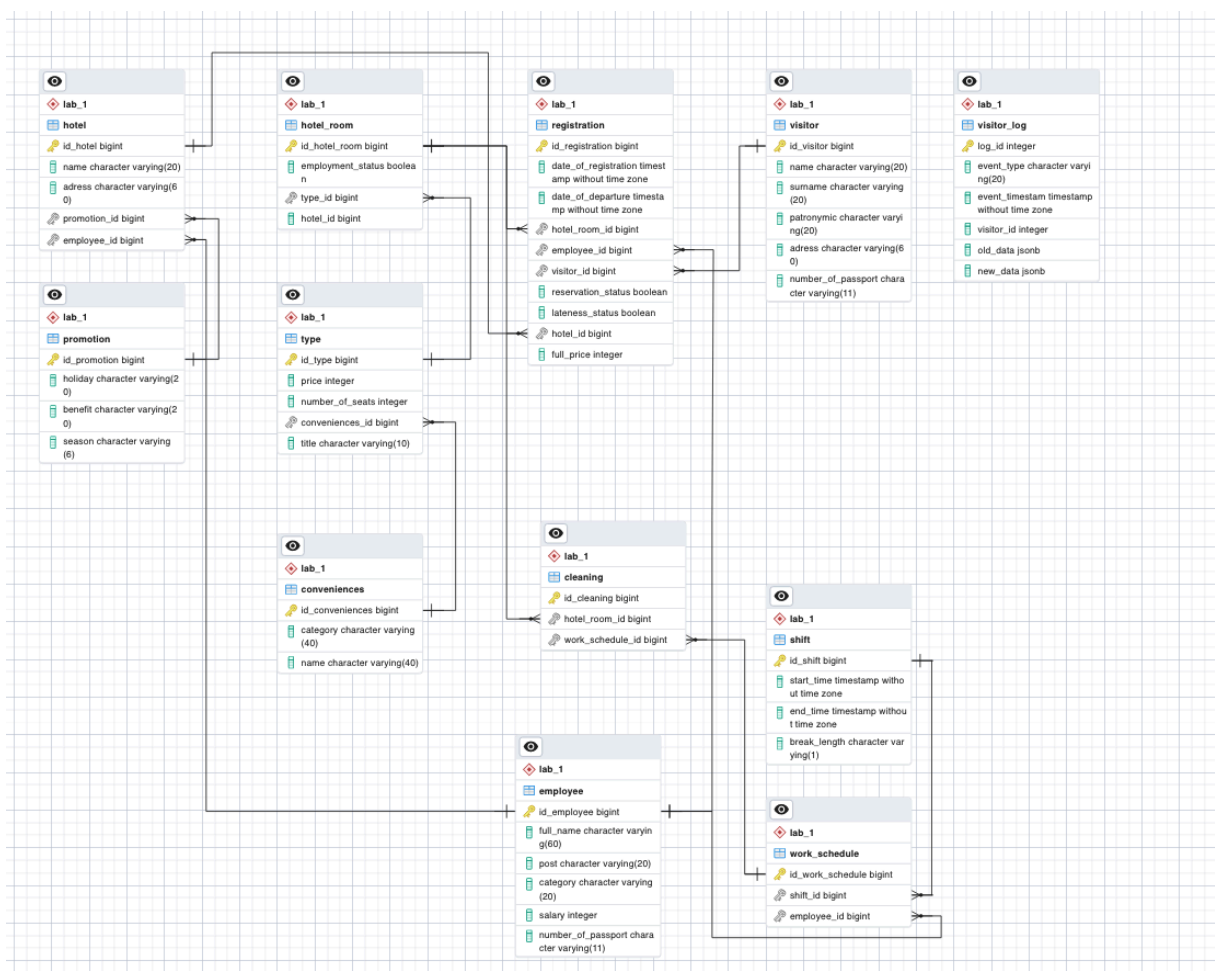


Рисунок 1 - ERD диаграмма

Задание_1: Создание хранимых процедур

1. Хранимая процедура для увеличения цены всех номеров на 5 %, если в отеле нет свободных номеров

```
CREATE OR REPLACE FUNCTION price_increase_if_no_available_rooms() RETURNS void AS $$
DECLARE there_are_available_rooms BOOLEAN;
BEGIN
    SELECT EXISTS (
        SELECT 1
        FROM lab_1.hotel_room
        WHERE employment_status IS FALSE
    )
    INTO there_are_available_rooms = TRUE;
    IF there_are_available_rooms IS FALSE THEN
        UPDATE lab_1.type
        SET price = price * 1.05;
    END IF;
END;
$$
LANGUAGE plpgsql;
```

Результат вызова функции, если свободные номера есть:

```
SELECT * FROM price_increase_if_no_available_rooms();
```

	id_type [PK] bigint	price integer	number_of_seats integer	conveniences_id bigint	title character varying (10)
1	6	15000	8	1	Президент
2	7	2500	2	2	Классик
3	8	4000	3	3	Классик
4	9	8000	4	4	Люкс
5	10	10000	6	5	Люкс
6	11	50000	2	2	Президент
7	12	250000	10	1	Бог
8	13	300000	2	5	Илон Маск

Результат вызова функции, если свободных номеров нет:


	id_type [PK] bigint	price integer	number_of_seats integer	conveniences_id bigint	title character varying (10)
1	6	15750	8	1	Президент
2	7	2625	2	2	Классик
3	8	4200	3	3	Классик
4	9	8400	4	4	Люкс
5	10	10500	6	5	Люкс
6	11	52500	2	2	Президент
7	12	262500	10	1	Бог
8	13	315000	2	5	Илон Маск

2. Хранимая процедура для получения информации о свободных одноместных номерах отеля на завтрашний день

```
CREATE OR REPLACE FUNCTION search_available_rooms_tomorrow()
RETURNS TABLE (
    available_rooms bigint
) AS $$
BEGIN
    RETURN QUERY
    SELECT id_hotel_room
    FROM lab_1.hotel_room
    JOIN lab_1.type ON hotel_room.type_id = type.id_type
    WHERE number_of_seats = 1
    AND employment_status = FALSE
    AND hotel_room.id_hotel_room NOT IN (
        SELECT registration.hotel_room_id
        FROM lab_1.registration
        WHERE date_of_registration = CURRENT_DATE + INTERVAL '1 day'
    );
END;
$$
LANGUAGE plpgsql
```

Вызов функции для просмотра результата:

```
SELECT * FROM search_available_rooms_tomorrow();
```

	available_rooms_tomorrow 
	bigint
1	16
2	17

3. Хранимая процедура для бронирования двухместного номера в гостинице на заданную дату и количество дней проживания

```
CREATE OR REPLACE FUNCTION double_room_booking(v_name VARCHAR(20), v_surname VARCHAR(20), v_patronymic
VARCHAR(20),
    v_adress VARCHAR(60), v_number_of_passport VARCHAR(11), check_in DATE, duration INTEGER)
RETURNS void AS $$
BEGIN
    INSERT INTO lab_1.visitor (name, surname, patronymic, adress, number_of_passport)
    VALUES (v_name, v_surname, v_patronymic, v_adress, v_number_of_passport);
    IF EXISTS (
        SELECT 1
        FROM lab_1.hotel_room
        JOIN lab_1.type ON hotel_room.type_id = type.id_type
        WHERE number_of_seats = 2
        AND employment_status = FALSE
        AND hotel_room.id_hotel_room NOT IN (
            SELECT registration.hotel_room_id
```

```

        FROM lab_1.registration
        WHERE date_of_registration <= check_in + duration - 1
        AND date_of_departure >= check_in
    )
) THEN
INSERT INTO lab_1.registration (date_of_registration, date_of_departure, hotel_room_id, employee_id, visitor_id,
    reservation_status, lateness_status, hotel_id, full_price)
VALUES (
    check_in + INTERVAL '1 hour',
    check_in + duration - 1 + INTERVAL '1 hour',
    (
        SELECT id_hotel_room
        FROM lab_1.hotel_room
        JOIN lab_1.type ON hotel_room.type_id = type.id_type
        WHERE number_of_seats = 2
        AND employment_status = FALSE
        AND hotel_room.id_hotel_room NOT IN (
            SELECT registration.hotel_room_id
            FROM lab_1.registration
            WHERE date_of_registration <= check_in + duration - 1
            AND date_of_departure >= check_in
        )
        LIMIT 1
    ),
    (
        SELECT id_employee
        FROM lab_1.employee
        WHERE post = 'Администратор'
        LIMIT 1
    ),
    (
        SELECT id_visitor
        FROM lab_1.visitor
        WHERE number_of_passport = v_number_of_passport
    ),
    TRUE,
    FALSE,
    (
        SELECT id_hotel
        FROM lab_1.hotel_room
        JOIN lab_1.type ON hotel_room.type_id = type.id_type
        JOIN lab_1.hotel ON hotel_room.hotel_id = hotel.id_hotel
        WHERE number_of_seats = 2
        AND employment_status = FALSE
        AND hotel_room.id_hotel_room NOT IN (
            SELECT registration.hotel_room_id
            FROM lab_1.registration
            WHERE date_of_registration <= check_in + duration - 1
            AND date_of_departure >= check_in
        )
        LIMIT 1
    ),
    (
        SELECT price
        FROM lab_1.hotel_room
        JOIN lab_1.type ON hotel_room.type_id = type.id_type
        WHERE number_of_seats = 2
        AND employment_status = FALSE
        AND hotel_room.id_hotel_room NOT IN (
            SELECT registration.hotel_room_id
            FROM lab_1.registration
            WHERE date_of_registration <= check_in + duration - 1
            AND date_of_departure >= check_in
        )
        LIMIT 1
    )
);
END IF;
END;
$$
LANGUAGE plpgsql;

```

Вывод таблицы с регистрацией до выполнения хранимой процедуры

	id_registration [PK] bigint	date_of_registration timestamp without time zone	date_of_departure timestamp without time zone	hotel_room_id bigint	employee_id bigint	visitor_id bigint	reservation_status boolean	lateness_status boolean	hotel_id bigint	full_price integer
1	6	2023-07-11 11:11:11	2023-07-17 22:22:22	6	1	1	true	false	1	1000
2	7	2023-07-15 13:34:11	[null]	7	2	3	true	false	2	1500
3	8	2023-07-07 18:17:45	2023-07-10 11:43:17	8	3	5	true	true	3	2000
4	9	2023-07-01 11:00:12	2023-07-05 10:16:23	9	4	4	false	false	4	2500
5	10	2023-07-12 14:15:16	2023-07-18 13:56:34	10	3	2	false	false	5	3000
6	11	2023-07-10 12:54:12	2023-07-14 21:16:34	8	4	5	true	false	1	3500
7	12	2023-07-01 13:46:54	2023-07-16 11:10:15	15	1	2	false	true	2	4000
8	13	2023-07-16 12:32:11	[null]	12	3	4	true	true	3	4500
9	14	2023-07-16 20:15:12	[null]	6	1	2	true	false	2	5000
10	15	2023-07-17 11:10:09	2023-07-18 10:32:11	8	3	5	false	false	1	5500
11	16	2023-04-11 13:13:56	2023-05-23 11:45:46	7	4	4	true	true	2	6000
12	17	2022-11-12 23:11:45	2022-12-30 10:11:47	8	2	3	true	false	1	6500
13	18	2023-07-23 16:13:31	[null]	10	3	5	false	true	5	7000
14	19	2020-07-17 20:32:30	[null]	6	3	4	true	true	2	7500
15	20	2003-10-15 08:07:02	2019-07-17 01:31:49	7	1	5	false	false	1	8000
16	21	2015-05-03 17:45:49	2016-01-13 02:14:53	8	2	1	false	true	5	8500
17	22	2021-12-20 01:44:29	[null]	9	4	2	true	true	4	9000
18	23	2008-02-28 17:07:40	2024-06-19 21:20:57	10	4	3	false	false	3	9500
19	24	2023-12-26 03:03:25	[null]	11	4	4	true	true	2	10000
20	25	2008-09-04 00:09:04	2008-10-01 21:40:59	12	1	5	false	false	1	10500
21	26	2000-12-17 03:21:43	2021-12-05 11:16:54	13	3	1	true	false	5	11000
22	27	2014-06-23 02:31:18	2019-12-12 16:03:27	14	2	2	true	false	4	11500
23	28	2022-12-10 14:11:56	2022-12-12 23:17:49	6	1	5	false	true	1	12000
24	29	[null]	[null]	17	2	3	true	false	2	12500

Вывод таблицы с регистрацией после выполнения хранимой процедуры

SELECT * FROM double_room_booking('Артём', 'Бочкарь', 'Артёмович', 'г. СПб, ул. Марата, дом 6, кв. 1', '4016 236738', '2023-07-28', 7);

	id_registration [PK] bigint	date_of_registration timestamp without time zone	date_of_departure timestamp without time zone	hotel_room_id bigint	employee_id bigint	visitor_id bigint	reservation_status boolean	lateness_status boolean	hotel_id bigint	full_price integer
1	6	2023-07-11 11:11:11	2023-07-17 22:22:22	6	1	1	true	false	1	1000
2	7	2023-07-15 13:34:11	[null]	7	2	3	true	false	2	1500
3	8	2023-07-07 18:17:45	2023-07-10 11:43:17	8	3	5	true	true	3	2000
4	9	2023-07-01 11:00:12	2023-07-05 10:16:23	9	4	4	false	false	4	2500
5	10	2023-07-12 14:15:16	2023-07-18 13:56:34	10	3	2	false	false	5	3000
6	11	2023-07-10 12:54:12	2023-07-14 21:16:34	8	4	5	true	false	1	3500
7	12	2023-07-01 13:46:54	2023-07-16 11:10:15	15	1	2	false	true	2	4000
8	13	2023-07-16 12:32:11	[null]	12	3	4	true	true	3	4500
9	14	2023-07-16 20:15:12	[null]	6	1	2	true	false	2	5000
10	15	2023-07-17 11:10:09	2023-07-18 10:32:11	8	3	5	false	false	1	5500
11	16	2023-04-11 13:13:56	2023-05-23 11:45:46	7	4	4	true	true	2	6000
12	17	2022-11-12 23:11:45	2022-12-30 10:11:47	8	2	3	true	false	1	6500
13	18	2023-07-23 16:13:31	[null]	10	3	5	false	true	5	7000
14	19	2020-07-17 20:32:30	[null]	6	3	4	true	true	2	7500
15	20	2003-10-15 08:07:02	2019-07-17 01:31:49	7	1	5	false	false	1	8000
16	21	2015-05-03 17:45:49	2016-01-13 02:14:53	8	2	1	false	true	5	8500
17	22	2021-12-20 01:44:29	[null]	9	4	2	true	true	4	9000
18	23	2008-02-28 17:07:40	2024-06-19 21:20:57	10	4	3	false	false	3	9500
19	24	2023-12-26 03:03:25	[null]	11	4	4	true	true	2	10000
20	25	2008-09-04 00:09:04	2008-10-01 21:40:59	12	1	5	false	false	1	10500
21	26	2000-12-17 03:21:43	2021-12-05 11:16:54	13	3	1	true	false	5	11000
22	27	2014-06-23 02:31:18	2019-12-12 16:03:27	14	2	2	true	false	4	11500
23	28	2022-12-10 14:11:56	2022-12-12 23:17:49	6	1	5	false	true	1	12000
24	29	[null]	[null]	17	2	3	true	false	2	12500
25	31	2023-07-28 01:00:00	2023-08-03 01:00:00	18	1	12	true	false	5	20000

+ Новый созданный клиент

	id_visitor [PK] bigint	name character varying (20)	surname character varying (20)	patronymic character varying (20)	adress character varying (60)	number_of_passport character varying (11)
1	1	Артём	Бочкарь	Артёмович	Санкт-Петербург, улица Бронная, дом 15/4, кв. 1	4013 457890
2	2	Лобус	Виктория	Витальевна	Санкт-Петербург, улица Бронная, дом 15/4, кв. 1	4013 673412
3	3	Комаров	Алексей	Иванович	Мурманск, проспект Сергея Приминина, дом 37, кв. 16	4011 554421
4	4	Исаев	Батыр	Бактыбекович	Москва, улица Советников, дом 12, кв. 567	4010 425363
5	5	Кислый	Иван	Петрович	Пермь, улица Горская, дом 3, кв. 114	4006 571973
6	12	Артём	Бочкарь	Артёмович	г. СПб, ул. Марата, дом 6, кв. 1	4016 236738

Задание_2: Создание триггеров для логирования событий вставки, удаления и обновления данных в таблице

Создание таблицы логирования

```
CREATE TABLE lab_1.visior_log (  
    log_id SERIAL PRIMARY KEY,  
    event_type VARCHAR(20) NOT NULL,  
    event_timestam TIMESTAMP DEFAULT current_timestamp,  
    client_id INTEGER,  
    old_data JSONB,  
    new_data JSONB  
);
```

Создани триггера для логирования взаимодействий с таблицей visitor

```
CREATE OR REPLACE FUNCTION lab_1.visitor_log_trigger_function()  
RETURNS TRIGGER AS  
$$  
BEGIN  
    IF (TG_OP = 'INSERT') THEN  
        INSERT INTO lab_1.visitor_log (event_type, visitor_id, new_data)  
        VALUES ('INSERT', NEW.id_visitor, to_jsonb(NEW));  
    ELSIF (TG_OP = 'UPDATE') THEN  
        INSERT INTO lab_1.visitor_log (event_type, visitor_id, old_data, new_data)  
        VALUES ('UPDATE', NEW.id_visitor, to_jsonb(OLD), to_jsonb(NEW));  
    ELSIF (TG_OP = 'DELETE') THEN  
        INSERT INTO lab_1.visitor_log (event_type, visitor_id, old_data)  
        VALUES ('DELETE', OLD.id_visitor, to_jsonb(OLD));  
    END IF;  
    RETURN NEW;  
END;  
$$  
LANGUAGE plpgsql;
```

Создание триггера

```
CREATE TRIGGER visitor_log_trigger  
AFTER INSERT OR UPDATE OR DELETE ON lab_1.visitor  
FOR EACH ROW
```

```
EXECUTE FUNCTION lab_1.visitor_log_trigger_function();
```

Проверка работы триггера

```
INSERT INTO lab_1.visitor (name, surname, patronymic, adress, number_of_passport)
VALUES ('Игорь', 'Павлов', 'Петрович', 'г.Москва, ул.Пушкина, дом.Колотушкина, кв.1', '4023
541278')
```

	log_id [PK] integer	event_type character varying (20)	event_timestam timestamp without time zone	visitor_id integer	old_data jsonb	new_data jsonb
1	1	INSERT	2023-08-07 11:16:45.719893	21	[null]	{"name": "И...

```
UPDATE lab_1.visitor
SET number_of_passport = '4012 345612'
WHERE name = 'Игорь' AND surname = 'Павлов' AND patronymic = 'Петрович'
```

	log_id [PK] integer	event_type character varying (20)	event_timestam timestamp without time zone	visitor_id integer	old_data jsonb	new_data jsonb
1	1	INSERT	2023-08-07 11:16:45.719893	21	[null]	{"name": "..."
2	2	UPDATE	2023-08-07 11:21:35.437288	21	{"name": "..."	{"name": "..."

```
DELETE FROM lab_1.visitor
WHERE number_of_passport = '4012 345612'
```

	log_id [PK] integer	event_type character varying (20)	event_timestam timestamp without time zone	visitor_id integer	old_data jsonb	new_data jsonb
1	1	INSERT	2023-08-07 11:16:45.719893	21	[null]	{"name": "..."
2	2	UPDATE	2023-08-07 11:21:35.437288	21	{"name": "..."	{"name": "..."
3	3	DELETE	2023-08-07 11:25:11.419421	21	{"name": "..."	[null]

Вывод

В ходе лабораторной работы получилось овладеть навыками создания хранимых процедур и триггеров в PSQL, были созданы функции и триггеры согласно индивидуальному заданию варианта 3.

Чтобы создать универсальный триггер для логирования данных можно воспользоваться переменными **old::text** и **new::text**, которые отдадут текстовую репрезентацию вставляемых/изменяемых/удаляемых данных. Также можно использовать переменную **TG_TABLE_NAME**, которая содержит в себе имя таблицы, которая участвует в операции. Таким образом, можно создать таблицу с логами, которая будет содержать в себе операцию (**TG_OP**), имя таблицы (**TG_TABLE_NAME**), старые данные (**old::text**) и новые данные (**new::text**).