**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 PostgresSQL**»

**Выполнил:**

студент группы K32392

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:**

**Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание**

Рисунок 1 - ERD диаграмма

**Задание\_1: Создание хранимых процедур**

 1. Хранимая процедурамдля 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();

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание

Результат вызова функции, если свободных номеров нет:

Изображение выглядит как текст, число, Шрифт, снимок экрана

Автоматически созданное описание

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();

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

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

Изображение выглядит как текст, меню, чек

Автоматически созданное описание

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);

Изображение выглядит как текст, меню, снимок экрана, чек

Автоматически созданное описание

+ Новый созданный клиент

Изображение выглядит как текст, снимок экрана, Шрифт, чек

Автоматически созданное описание

**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

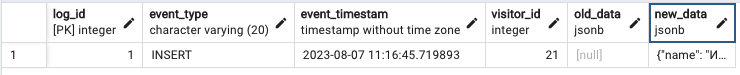
FOR EACH ROW

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')



UPDATE lab\_1.visitor

SET number\_of\_passport = '4012 345612'

WHERE name = 'Игорь' AND surname = 'Павлов' AND patronymic = 'Петрович'

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

DELETE FROM lab\_1.visitor

WHERE number\_of\_passport = '4012 345612'

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

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**).