**Министерство науки и высшего образования Российской Федерации** федеральное государственное автономное образовательное учреждение высшего образования

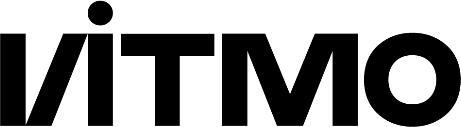
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Отчет**

по лабораторной работе №5 «Процедуры, функции, триггеры в PostgreSQL**»** по дисциплине **«Проектирование и реализация баз данных»**

Автор: Русинов В.А. Факультет: ИКТ Группа: К3240

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

**Оглавление**

1. [Создать процедуры согласно индивидуальному заданию 3](#_bookmark0)
2. [Модифицировать триггер 5](#_bookmark1)
3. [Создать необходимый триггер 7](#_bookmark2)

[Вывод 8](#_bookmark3)

**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:**

* 1. Создать процедуры/функции согласно индивидуальному заданию (часть 4).
  2. Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу).
  3. Создать авторский триггер по варианту индивидуального задания.

# Создать процедуры согласно индивидуальному заданию

Создать хранимые процедуры:

* + Для поиска билетов в заданный пункт назначения.

CREATE OR REPLACE FUNCTION find\_tickets\_to\_destination(destination\_city VARCHAR(20))

RETURNS TABLE (

id\_ticket INT, passenger\_name TEXT, date\_departure DATE, date\_arrival DATE

)

LANGUAGE plpgsql AS $$ BEGIN

RETURN QUERY

SELECT t.id\_ticket, CONCAT(p.name, ' ', p.surname) AS passenger\_name, r.date\_departure, r.date\_arrival

FROM schema.ticket t

JOIN schema.passenger p ON t.id\_passenger = p.id\_passenger JOIN schema.route r ON t.id\_route = r.id\_route

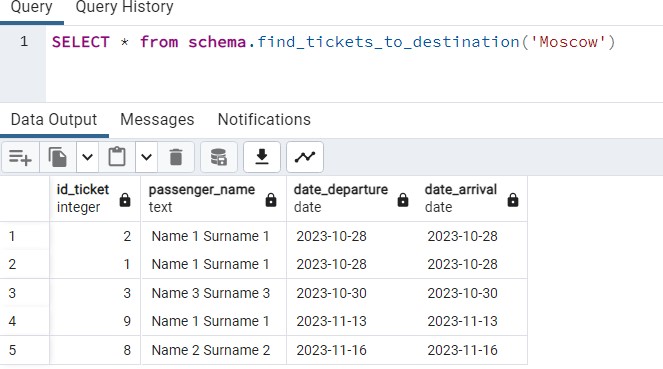
JOIN schema.schedule s ON r.id\_schedule = s.id\_schedule

JOIN schema.airport a\_arr ON s.id\_airport\_arrival = a\_arr.id\_airport WHERE a\_arr.city = destination\_city

ORDER BY r.date\_departure;

END;

$$;



* Создания новой кассы продажи билетов.

CREATE OR REPLACE PROCEDURE create\_ticket\_office( city VARCHAR(20),

address VARCHAR(50), country VARCHAR(20)

)

LANGUAGE SQL

AS $$

INSERT INTO schema.ticket\_office (city, address, country) VALUES (city, address, country);

$$;



* Определить расход топлива по всем маршрутам за истекший месяц.

CREATE OR REPLACE FUNCTION display\_fuel\_consumption\_table() RETURNS TABLE (

id\_route INT, fuel\_consumption NUMERIC

)

LANGUAGE plpgsql AS $$ BEGIN

RETURN QUERY

SELECT r.id\_route,

ROUND((EXTRACT(EPOCH FROM s.time\_arrival) - EXTRACT(EPOCH FROM

s.time\_departure))/3600 \* p.fuel\_rate, 0) AS fuel\_consumption FROM schema.route r

JOIN schema.schedule s ON r.id\_schedule = s.id\_schedule JOIN schema.plane pl ON r.id\_plane = pl.id\_plane

JOIN schema.model p ON pl.id\_model = p.id\_model;

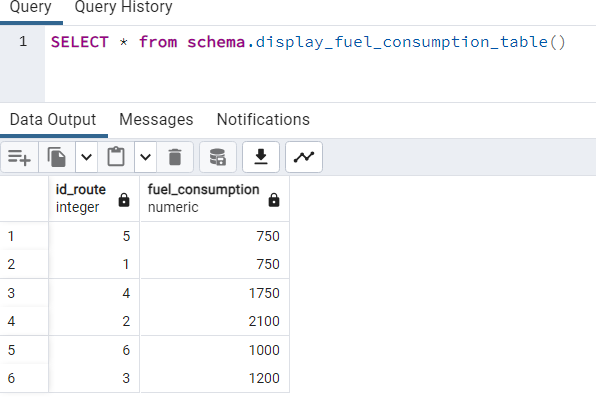
WHERE

s.time\_departure >= date\_trunc('month', CURRENT\_DATE) AND

s.time\_departure < date\_trunc('month', CURRENT\_DATE) + INTERVAL '1 month';

END;

$$;



# Модифицировать триггер

create or replace function fn\_check\_time\_punch() returns trigger as $psql$ begin if

new.is\_out\_punch = (select tps.is\_out\_punch from time\_punch tps

where tps.employee\_id = new.employee\_id order by tps.id desc limit 1 ) or

new.punch\_time>now() or

new.punch\_time <= (select tps.punch\_time from time\_punch tps

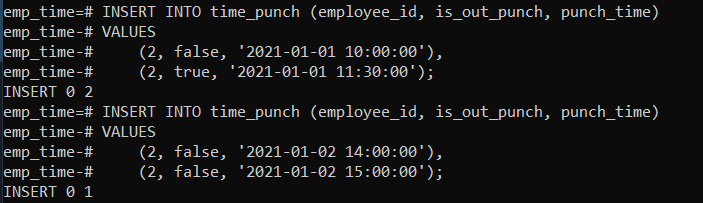
where tps.employee\_id = new.employee\_id order by tps.id desc limit 1 )

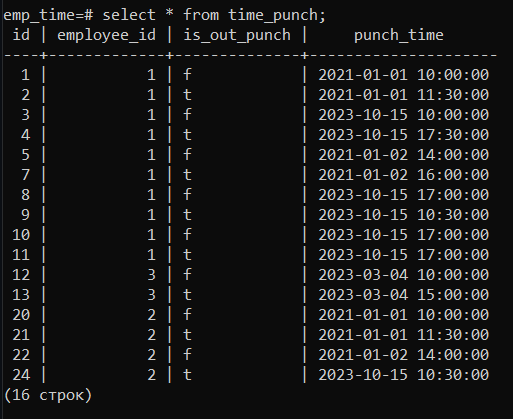
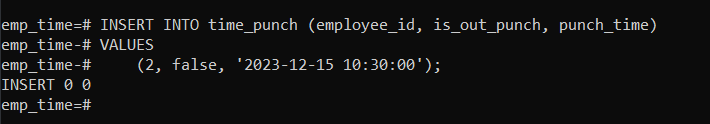
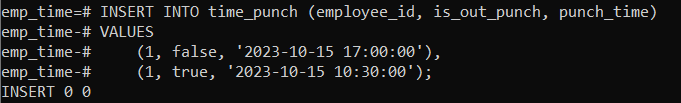
then return null; end if; return new; end;

$psql$ language plpgsql;

drop trigger if exists check\_time\_punch on time\_punch; create trigger check\_time\_punch

before insert on time\_punch for each row execute procedure fn\_check\_time\_punch();





# Создать необходимый триггер

CREATE OR REPLACE FUNCTION update\_ticket\_status() RETURNS TRIGGER AS $$

BEGIN

IF NEW.payment\_status = 'Paid' AND OLD.payment\_status <> 'Paid' THEN UPDATE schema.ticket

SET status = 'Purchased'

WHERE id\_ticket = NEW.id\_ticket; END IF;

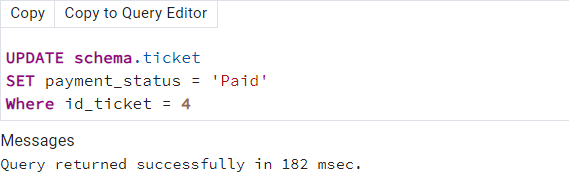
RETURN NEW; END;

$$ LANGUAGE plpgsql;

CREATE TRIGGER ticket\_payment\_status\_trigger AFTER UPDATE ON schema.ticket

FOR EACH ROW

EXECUTE FUNCTION update\_ticket\_status();





# Вывод

В ходе лабораторной работы были успешно созданы и протестированы процедуры и триггеры в системе управления базами данных PostgreSQL. Работа позволила овладеть практическими навыками программирования в SQL и пониманием механизма триггеров.