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

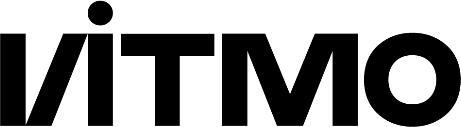
### «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

### Отчет

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

Автор: Даньшин С. А. Факультет: ИКТ Группа: К3240

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

**Оглавление**

[Вариант 7. БД «Курсы» 3](#_bookmark0)

[Ход работы 4](#_bookmark1)

[Вывод 12](#_bookmark2)

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

**Оборудование:** компьютерный класс.

**Программное обеспечение:** СУБД PostgreSQL, pgadmin 4.

### Практическое задание:

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

* 1. 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).
  2. Создать авторский триггер по варианту индивидуального задания.

**Вариант 10. БД «Автовокзал»**

**Задание 4.** Создать хранимые процедуры:

* Продажи билета.
* Возврата билета.
* Добавления нового рейса.

**Задание 5.** Создать необходимые триггеры.

# Ход работы:

Certainly! Below are the raw SQL code snippets for the stored procedures you've described:

1. Продажи билета

**CREATE** **or** **replace** **PROCEDURE** TicketSale (

**IN** p\_passenger\_id **INT**,

**IN** p\_trip\_id **INT**,

**IN** p\_seat\_number **INT**,

**IN** p\_start\_station\_id **INT**,

**IN** p\_end\_station\_id **INT**,

**IN** p\_is\_online\_sale **BOOLEAN**

)

**LANGUAGE** plpgsql

**AS** **$$**

**begin**

**RAISE** **NOTICE** 'Input Parameters: %, %, %, %, %, %', p\_passenger\_id, p\_trip\_id, p\_seat\_number, p\_start\_station\_id, p\_end\_station\_id, p\_is\_online\_sale;

**INSERT** **INTO** tickets (passenger\_id, trip\_id, seat\_number, status, start\_station\_id, end\_station\_id, is\_online\_sale, sold\_at)

**VALUES** (p\_passenger\_id, p\_trip\_id, p\_seat\_number, 'PAID', p\_start\_station\_id, p\_end\_station\_id, p\_is\_online\_sale, **NOW**());

**END**;

**$$**;

2. Возврата билета

**CREATE** **OR** **REPLACE** **PROCEDURE** RefundTicket (

**IN** p\_ticket\_id **INT**

)

**LANGUAGE** plpgsql

**AS** **$$**

**BEGIN**

**UPDATE** tickets

**SET**

status = 'CANCELED',

is\_online\_sale = **FALSE**

**WHERE**

id = p\_ticket\_id;

**END**;

**$$**;

1. Добавления нового рейса

**CREATE** **OR** **REPLACE** **PROCEDURE** AddNewTrip (

**IN** p\_route\_id **INT**,

**IN** p\_bus\_id **INT**,

**IN** p\_start\_time **TIMESTAMP**,

**IN** p\_end\_time **TIMESTAMP**,

**IN** p\_status **tripstatus**,

**IN** p\_price **INT**

)

**LANGUAGE** plpgsql

**AS** **$$**

**BEGIN**

**INSERT** **INTO** trips (route\_id, bus\_id, start\_time, end\_time, status, price)

**VALUES** (p\_route\_id, p\_bus\_id, p\_start\_time, p\_end\_time, p\_status, p\_price);

**END**;

**$$**;

1. Модифицировать триггер на проверку корректности входа и выхода сотрудника

(имеющиеся проблемы: может быть отрицательное время работы, человек зашел/вышел в будущем)

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();

1. Авторский триггер для проверки при вставке билета, что место свободно:

**REATE** **OR** **REPLACE** **FUNCTION** check\_seat\_availability()

**RETURNS** **TRIGGER** **AS** **$$**

**DECLARE**

v\_seat\_count **INT**;

**BEGIN**

-- Check if the seat is available for the specified trip

**SELECT** **COUNT**(\*)

**INTO** v\_seat\_count

**FROM** tickets

**WHERE** trip\_id = **NEW**.trip\_id

**AND** seat\_number = **NEW**.seat\_number

**AND** status = 'PAID'; -- Adjust if needed based on your business logic

**IF** v\_seat\_count > 0 **THEN**

-- Seat is already reserved or sold, raise an exception to prevent the insert

**RAISE** **EXCEPTION** 'Seat % for Trip % is not available.', **NEW**.seat\_number, **NEW**.trip\_id;

**END** **IF**;

-- Seat is available, proceed with the insert

**RETURN** **NEW**;

**END**;

**$$** **LANGUAGE** plpgsql;

**CREATE** **TRIGGER** check\_seat\_availability\_trigger

**BEFORE** **INSERT** **ON** tickets

**FOR** **EACH** **ROW** **EXECUTE** **FUNCTION** check\_seat\_availability();

# Вывод

## В ходе выполнения данной лабораторной работы научились создавать и работать с процедурами, функциями и триггерами в PostgreSQL.