

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

«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

**Лабораторная работа №3**  
**«Процедуры, функции, триггеры в PostgreSQL»**  
**по дисциплине «Проектирование и**  
**реализация баз данных»**

Выполнили  
: студент II курса ИКТ  
группы К3241  
Траоре Мамуду.

Проверила:  
Говорова М.М

Санкт-Петербург 2022

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

**Оборудование:** компьютерный класс.

**Программное обеспечение:** СУБД PostgreSQL.

### Практическое задание: Вариант 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.

### Выполнение:

#### 1. Создадим базу данных



```
Query Editor  Query History
1  create database emp_time;

Data Output  Messages  Notifications
CREATE DATABASE

Query returned successfully in 933 msec.
```

#### 2. Создадим таблицу работников:



```
Query Editor  Query History
1  create table employee
2  (
3      id serial primary key,
4      username varchar(500)
5  );
6

Data Output  Messages  Notifications
CREATE TABLE

Query returned successfully in 306 msec.
```

### 3. Создадим таблицу отметок:

```
7 create table time_punch
8 (
9     id serial primary key,
10    employee_id int references employee(id),
11    is_out_punch boolean default false, --false - вход
12    punch_time timestamp default now()
13 );
14 |
```

Data Output Messages Notifications

CREATE TABLE

Query returned successfully in 75 msec.

### 4. Создаём триггерную функцию:

Query Editor Query History

```
1
2 create or replace function fn_check_time_punch() returns trigger
3 as
4 $$
5 declare
6     punch_id int; --идентификатор предыдущей отметки
7     last_time timestamp; --предыдущее время отметки
8     punch_type boolean; --тип предыдущего события
9 begin
10    --получаем данные по предыдущей отметке
11    select id, punch_time, is_out_punch
12    into punch_id, last_time, punch_type
13    from time_punch
14    where employee_id = new.employee_id
15    and id = (select max(id)
16              from time_punch
17              where employee_id = new.employee_id);
18
19    --raise notice '%, %, %', punch_id, last_time, punch_type;
20    --предыдущих отметок не было
21    if(punch_id is null)then
22        --новая отметка на вход- все ок
23        if(new.is_out_punch = false)then
24            return new;
```

Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 294 msec.

```

16      from time_punch
17      where employee_id = new.employee_id);
18
19      --raise notice '%, %, %', punch_id, last_time, punch_type;
20      --предыдущих отметок не было
21  if(punch_id is null) then
22      --новая отметка на вход- все ок
23  if(new.is_out_punch = false) then
24      return new;
25  else
26      --нельзя сразу выйти, первым должен быть вход
27      return null;
28  end if;
29  end if;
30  --предыдущая отметка Должна быть другого типа
31  if(punch_type = new.is_out_punch) then
32      raise notice 'После отметки на вход, должна идти отметка на выход или наоборот';
33      return null;
34  end if;
35  --время предыдущей отметки больше или равно времени текущей отметки
36  if(new.punch_time <= last_time) then
37      raise notice 'Неверное время отметки';
38      return null;
39  end if;
40  return new;
41  exception when others then
42      return null;
43  end;
44  $$
45  language plpgsql;

```

## 5. Создаем триггер

```

1  create trigger check_time_punch_trg
2  before insert
3  on time_punch
4  for each row
5  execute procedure fn_check_time_punch();
6

```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 81 msec.

## 6. Заполняем таблицу сотрудников:

```

1 insert into employee (username)
2 values
3 ('Иванов'),
4 ('Петров'),
5 ('Сидоров');
6

```

Data Output Messages Notifications

INSERT 0 3

Query returned successfully in 119 msec.

Query Editor Query History

```

1 select * from employee;
2
3

```

Data Output Messages Notifications

	id [PK] integer	username character varying (500)
1	1	Иванов
2	2	Петров
3	3	Сидоров

7. Пробуем заполнять таблицу с отметками:

-Первое событие выхода, а не входа

```

1 insert into time_punch (employee_id, is_out_punch, punch_time)
2 values
3 (2, true, '2022-06-02 09:59:10.000000'::timestamp);
4

```

Data Output Messages Notifications

INSERT 0 0

Query returned successfully in 131 msec.

-Событие выхода, но с временной отметкой раньше входа:

```

1 insert into time_punch (employee_id, is_out_punch, punch_time)
2 values
3 (1, true, '2022-06-01 09:59:10.000000'::timestamp);
4

```

Data Output Messages Notifications

NOTICE: Неверное время отметки

INSERT 0 0

Query returned successfully in 133 msec.

```

19 insert into time_punch (employee_id, is_out_punch, punch_time)
20 values
21 (1, false, '2022-06-01 11:00:00.000000'::timestamp);
22

```

Data Output Messages Notifications

NOTICE: После отметки на вход, должна идти отметка на выход или наоборот  
 INSERT 0 0

Query returned successfully in 99 msec.

## Исходный код:

```
create database emp_time;
```

```
create table employee
```

```
(
id serial primary key,
username varchar(500)
);
```

```
create table time_punch
```

```
(
id serial primary key,
employee_id int references employee(id),
is_out_punch boolean default false, --false - вход
punch_time timestamp default now()
);
```

```
create or replace function fn_check_time_punch() returns trigger
```

```
as
```

```
$$
```

```
declare
```

```
    punch_id int; --идентификатор предыдущей отметки
```

```
    last_time timestamp; --предыдущее время отметки
```

```
    punch_type boolean; --тип предыдущего события
```

```
begin
```

```
    --получаем данные по предыдущей отметке
```

```
    select id, punch_time, is_out_punch
```

```
    into punch_id, last_time, punch_type
```

```
    from time_punch
```

```
    where employee_id = new.employee_id
```

```
    and id = (select max(id)
```

```
                from time_punch
```

```
                where employee_id = new.employee_id);
```

```
--raise notice '%, %, %', punch_id, last_time, punch_type;
```

```
--предыдущих отметок не было
```

```
if(punch_id is null)then
```

```
    --новая отметка на вход- все ок
```

```

if(new.is_out_punch = false)then
    return new;
    else
        --нельзя сразу выйти, первым должен быть вход
        return null;
    end if;
end if;
--предыдущая отметка Должна быть другого типа
if(punch_type = new.is_out_punch)then
    raise notice 'После отметки на вход, должна идти отметка на выход или наоборот';
    return null;
end if;
--время предыдущей отметки больше или равно времени текущей отметки
if(new.punch_time <= last_time)then
    raise notice 'Неверное время отметки';
    return null;
end if;
    return new;
exception when others then
    return null;
end;
$$
language plpgsql;

create trigger check_time_punch_trg
    before insert
on time_punch
for each row
execute procedure fn_check_time_punch();

insert into employee (username)
values
('Иванов'),
('Петров'),
('Сидоров');

select * from employee;

insert into time_punch (employee_id, is_out_punch, punch_time)
values
(1, false, '2022-06-01 10:00:00.000000'::timestamp);

insert into time_punch (employee_id, is_out_punch, punch_time)
values
(1, false, '2022-06-01 11:00:00.000000'::timestamp);

insert into time_punch (employee_id, is_out_punch, punch_time)

```

```
values  
(1, true, '2022-06-01 09:59:10.000000'::timestamp);  
  
insert into time_punch (employee_id, is_out_punch, punch_time)  
values  
(2, true, '2022-06-02 09:59:10.000000'::timestamp);
```

**Вывод:** Была модифицирована и протестирована триггерная функция для проверки корректности входа и выхода сотрудника с учетом “узких” мест.