

**Министерство науки и высшего образования Российской Федерации**  
федеральное государственное автономное образовательное учреждение высшего  
образования  
**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

## **Отчет**

по лабораторной работе №3 «Процедуры, функции, триггеры в PostgreSQL»  
по дисциплине «Базы данных»

Выполнил: Холодов-Воронцов А. А.

Факультет: Инфокоммуникационных технологий

Группа: K3240

Проверила: Говорова М. М.



**УНИВЕРСИТЕТ ИТМО**

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, 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.

### БД Railroad

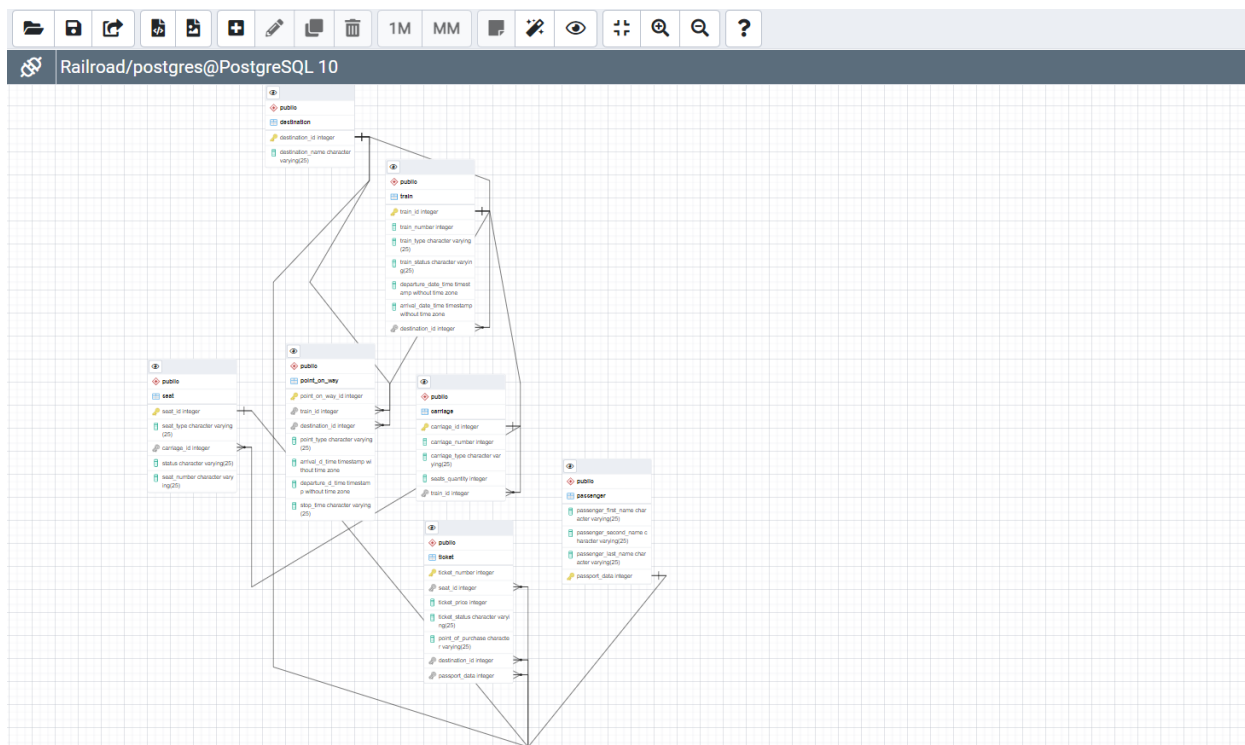


Рис.1 – ERD-схема логической модели БД

**Задание 1. Создать хранимые процедуры**

**1. Для повышения цен в пригородные поезда на 20%.**

```
create or replace function price_edit_20_percent() returns integer
as
$$
begin
update ticket set ticket_price = ticket_price * 1.2;

raise notice 'PRICE UPDATE!';

return 1;

end;
$$ language plpgsql volatile;
```

Query Editor   Query History

1   `select ticket_number, seat_id, ticket_price from ticket`

Data Output   Explain   Messages   Notifications

	<code>ticket_number</code> [PK] integer	<code>seat_id</code> integer	<code>ticket_price</code> integer
1	3	3	8500
2	6	1	16000
3	8	20	8500
4	9	21	8500
5	10	22	8500
6	11	23	8500
7	13	27	8500
8	15	29	8500
9	16	30	8500
10	17	35	8500
...	...	...	...

Рис.2 – До

Query Editor Query History

```
1 create or replace function price_edit_20_percent() returns integer
2 as
3 $$
4 ▼ begin
5 update ticket set ticket_price = ticket_price * 1.2;
6 raise notice 'PRICE UPDATE!';
7 return 1;
8 end;
9 $$ language plpgsql volatile;
10
```

Data Output Explain Messages Notifications

CREATE FUNCTION

Query returned successfully in 90 msec.

Рис.3 – Создание функции

Query Editor Query History

```
1 select ticket_number, seat_id, ticket_price from ticket
```

Data Output Explain Messages Notifications





	 ticket_number [PK] integer 	seat_id integer 	ticket_price integer 	
1	3	3	10200	
2	6	1	19200	
3	8	20	10200	
4	9	21	10200	
5	10	22	10200	
6	11	23	10200	
7	13	27	10200	
8	15	29	10200	
9	16	30	10200	
10	17	35	10200	

Рис.4 – После

## 2. Для создания нового рейса на поезд

create or replace function add\_new\_train(inout train\_number integer, inout train\_type varchar(25), inout train\_status varchar(25), inout departure\_date\_time timestamp without time zone, inout arrival\_date\_time timestamp without time zone, inout destination\_id integer)

as

\$\$

begin

insert into train(train\_number, train\_type, train\_status,  
departure\_date\_time, arrival\_date\_time, destination\_id)

values(train\_number, train\_type, train\_status,  
departure\_date\_time, arrival\_date\_time, destination\_id);

raise notice 'DONE!';

end;

\$\$ language plpgsql volatile;

Query Editor

Query History

1

```
select * from train
```

Data Output

Explain

Messages

Notifications









	<div>train_id</div> <div>[PK] integer </div>	<div>train_number</div> <div>integer </div>	<div>train_type</div> <div>character varying (25) </div>	<div>train_status</div> <div>character varying (25) </div>	<div>departure_date_time</div> <div>timestamp without time zone </div>	<div>arrival_date_time</div> <div>timestamp without time zone </div>	<div>destination_id</div> <div>integer </div>
1	19	1	Скоростной	Прибыл	2022-06-15 22:35:00	2022-06-16 02:15:00	1
2	17	3	Скорый	Прибыл	2022-06-16 15:00:00	2022-06-16 23:00:00	1
3	18	5	Скоростной	Прибыл	2022-06-08 22:30:00	2022-06-09 03:15:00	1
4	21	6	Скоростной	Прибыл	2022-06-20 23:35:00	2022-06-21 03:15:00	1
5	22	7	Скорый	Отменён	2022-06-22 13:28:00	2022-06-22 18:15:00	3

Рис.5 – До

Query Editor	Query History
<pre>1 create or replace function add_new_train(inout train_number integer, inout train_type varchar(25), 2                                     inout train_status varchar(25), 3                                     inout departure_date_time timestamp without time zone, 4                                     inout arrival_date_time timestamp without time zone, 5                                     inout destination_id integer) 6 as 7 \$\$ 8 begin 9     insert into train(train_number, train_type, train_status, 10                     departure_date_time, arrival_date_time, destination_id) 11 values(train_number, train_type, train_status, 12         departure_date_time, arrival_date_time, destination_id); 13 raise notice 'DONE!'; 14 end; 15 \$\$ language plpgsql volatile;</pre>	
Data Output	Explain
Messages	Notifications
CREATE FUNCTION	

Рис.6 – Создание функции

```
1 select * from train
2
```

Data Output Explain Messages Notifications

	train_id [PK] integer	train_number integer	train_type character varying (25)	train_status character varying (25)	departure_date_time timestamp without time zone	arrival_date_time timestamp without time zone	destination_id integer
1	19	1	Скоростной	Прибыл	2022-06-15 22:35:00	2022-06-16 02:15:00	1
2	17	3	Скорый	Прибыл	2022-06-16 15:00:00	2022-06-16 23:00:00	1
3	18	5	Скоростной	Прибыл	2022-06-08 22:30:00	2022-06-09 03:15:00	1
4	21	6	Скоростной	Прибыл	2022-06-20 23:35:00	2022-06-21 03:15:00	1
5	22	7	Скорый	Отменён	2022-06-22 13:28:00	2022-06-22 18:15:00	3
6	24	8	Скоростной	Ожидается отправл...	2022-06-28 11:07:00	2022-06-28 15:01:00	1

Рис. 7 – После

### 3. Для формирования общей выручки по продаже билетов за сутки.

create or replace function ticket\_profit\_3(d\_d\_time timestamp without time zone)

returns table(d\_1 timestamp without time zone, t\_1 bigint)

as

\$\$

begin

    return query

    (select ticket.departure\_date\_time, sum(ticket.ticket\_price)

    from ticket

    group by ticket.departure\_date\_time);

raise notice 'DONE!';

end;

\$\$ language plpgsql volatile;

select \* from ticket\_profit\_3('2022-06-27 23:35:00')

Query Editor

Query History

```

1 create or replace function ticket_profit_3(d_d_time timestamp without time zone)
2 returns table(d_1 timestamp without time zone, t_1 bigint)
3 as
4 $$
5 begin
6     return query
7     (select ticket.departure_date_time, sum(ticket.ticket_price)
8      from ticket
9      group by ticket.departure_date_time);
10 raise notice 'DONE!';
11
12 end;
13 $$ language plpgsql volatile;
14
15 select * from ticket_profit_3('2022-06-27 23:35:00')
16
17

```

Data Output

Explain

Messages

Notifications

	d_1 timestamp without time zone	t_1 bigint
1	2022-06-27 22:35:00	141600

Рис.8 – Создание функции и её вызов

## Задание 2. Создать триггер для логирования событий вставки, удаления, редактирования

Query Editor

Query History

```

1 create or replace function add_to_log() returns trigger as $$
2 declare
3     mstr varchar(40);
4     astr varchar(100);
5     retstr varchar(250);
6 begin
7     if tg_op = 'INSERT' then
8         astr = new;
9         mstr := 'add new data';
10        retstr := mstr||astr;
11        insert into logs(text, added, table_name) values (retstr, now(), TG_TABLE_NAME);
12        return new;
13    elsif tg_op = 'UPDATE' then
14        astr = new;
15        mstr := 'update data';
16        retstr := mstr||astr;
17        insert into logs(text, added, table_name) values (retstr, now(), TG_TABLE_NAME);
18    elsif tg_op = 'DELETE' then
19        astr = old;

```

Data Output

Explain

Messages

Notifications

CREATE FUNCTION

Рис.9 - Создание функции

Query Editor
Query History

```

1 trigger div_1 after insert or update or delete on public.passenger for each row execute procedure add_to_log()

```

Messages

CREATE TRIGGER

Query returned successfully in 233 msec.

Рис. 10 - Создание триггера

Query Editor
Query History

```

1 INSERT INTO public.passenger(
2     passenger_first_name, passenger_second_name, passenger_last_name, passport_data)
3     VALUES ('Ганс', 'Эрих', 'Куман', 229427);

```

Data Output
Explain
Messages
Notifications

INSERT 0 1

Рис.11 – INSERT

Query Editor
Query History

```

1 delete from passenger where passenger_second_name = 'Эрих'

```

Data Output
Explain
Messages
Notifications

DELETE 1

Рис.12 – DELETE

Query Editor
Query History

```

1 select * from logs

```

Data Output
Explain
Messages
Notifications

	text	table_name	added
	text	text	timestamp without time zone
1	Add new data(Ганс,Эрих,Куман,229427)	passenger	2022-06-28 12:20:49.601998
2	Remove data(Ганс,Эрих,Куман,229427)	passenger	2022-06-28 12:56:27.124722

Рис.13 - Проверка logs



## **Выводы**

В данной работе я изучил функции и процедуры, создал их для решения разных задач и условий. Также узнал про триггеры в БД и использовал их для хранения информации об изменениях данных в своей базе.