Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждениевысшего образования
«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

# ЛАБОРАТОРНАЯ РАБОТА №3

по теме: процедуры, функции, триггеры в PostgreSQL по дисциплине:
Проектирование и реализация баз данных

**Выполнил**:студент 3 курса ИКТ
группы К33401 Ф.И.О.: **Мамин И. И.**

**Проверила**: **Говорова Марина Михайловна**

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.

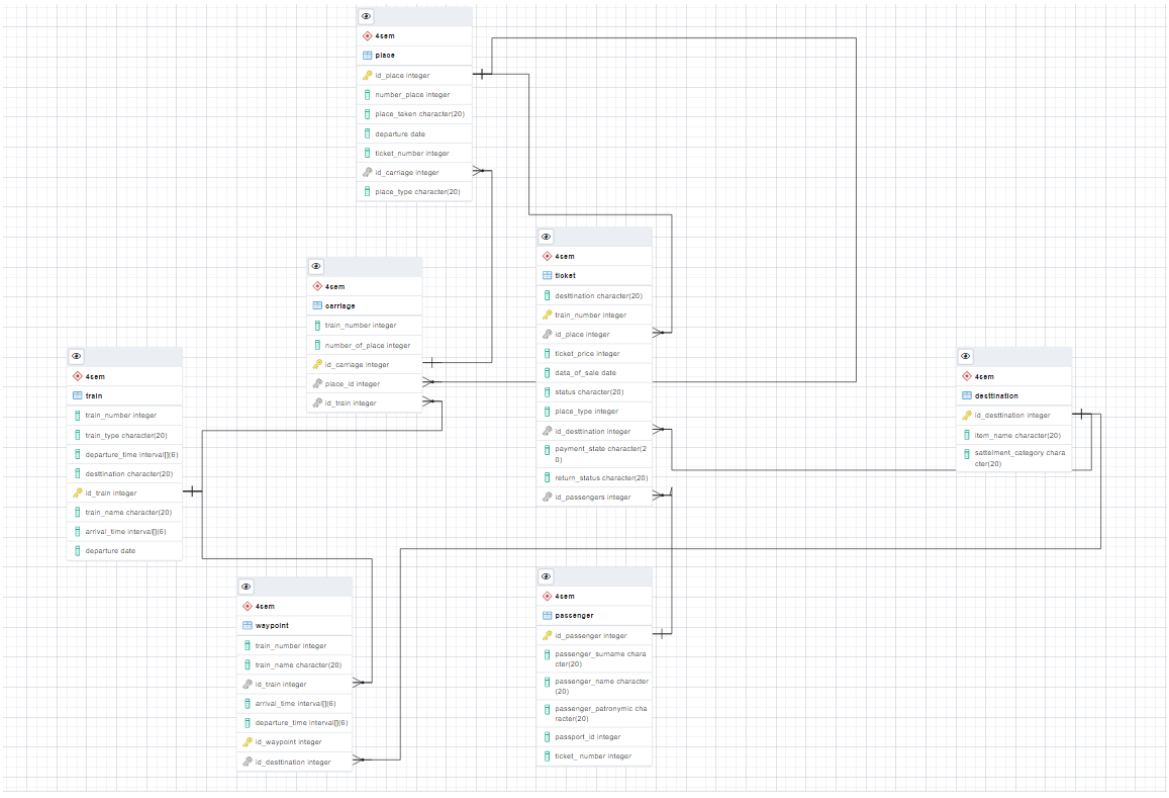
**Технология выполнения работы:**

**Вариант 6. БД «Пассажир»**

Описание предметной области: Информационная система служит для продажи железнодорожных билетов . Билеты могут продаваться на текущие сутки или предварительно (не более чем за 45 суток). Цена билета при предварительной продаже снижается на 5 %.

БД должна содержать следующий минимальный набор сведений: Номер поезда. Название поезда. Тип поезда. Пункт назначения. Пункт назначения для проданного билета. Номер вагона. Тип вагона. Количество мест в вагоне. Цена билета. Дата отправления. Дата прибытия. Дата прибытия для пункта назначения проданного билета. Время отправления. Номер вагона в поезде. Номер билета. Место. Тип места. Фамилия пассажира. Имя пассажира. Отчество пассажира. Паспортные данные.

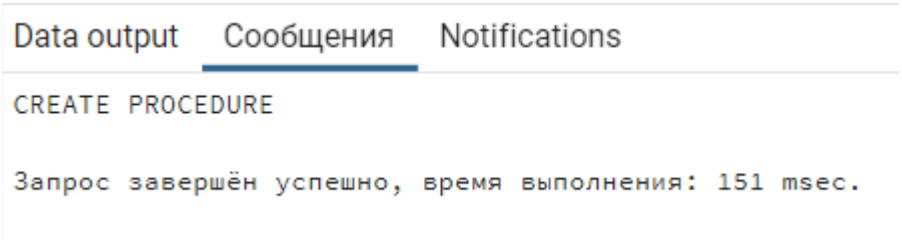
## Схема логической модели БД в нотации IDEF1X



Задание 4. Создать хранимые процедуры:

Для повышения цен в пригородные поезда на 20%.

```
create procedure "4sem".price_20()
language sql
as $$
update "4sem".ticket set ticket_price = ticket_price * 1.2
where ticket.destination = (select destination.item_name from "4sem".destination where destination.sattelment_category = 'Пригород');
$$;
```



Запрос История запросов

```
1  select destination, ticket_price from "4sem".ticket
2
```

Data output Сообщения Notifications

| | destination character (20) | ticket_price integer |
|---|---|---|
| 6 | Москва | 1500 |
| 7 | Новосибирск | 3000 |
| 8 | Москва | 12000 |
| 9 | Санкт-Петерб... | 1200 |
| 10 | Санкт-Петерб... | 11000 |
| 11 | Петергоф | 1000 |

```
1  call "4sem".price_20();
2  select destination, ticket_price from "4sem".ticket;
3
```

Data output | Сообщения | Notifications

| | destination character (20) | ticket_price integer |
|---|---|---|
| 6 | Москва | 1500 |
| 7 | Новосибирск | 3000 |
| 8 | Москва | 12000 |
| 9 | Санкт-Петерб... | 1200 |
| 10 | Санкт-Петерб... | 11000 |
| 11 | Петергоф | 1200 |

Для создания нового рейса на поезд.

```
create procedure "4sem".new_dest(id_destination integer,item_name character, sattelment_category character)
language sql
as $$
insert into "4sem".destination values (id_destination, item_name, sattelment_category);
$$;

```

Data output | Сообщения | Notifications

```
CREATE PROCEDURE

Запрос завершён успешно, время выполнения: 48 msec.
```

```
1  select * from "4sem".destination
```

Data output | Сообщения | Notifications

| | id_destination [PK] integer | item_name character (20) | sattelment_category character (20) |
|---|---|---|---|
| 1 | 1 | Москва | Город |
| 2 | 2 | Новосибирск | Город |
| 3 | 3 | Екатеринбург | Город |
| 4 | 4 | Санкт-Петерб… | Город |
| 5 | 5 | Петергоф | Пригород |
| 6 | 15 | Царское Село | Пригород |

```
1  call "4sem".new_dest('15','Царское Село', 'Пригород')
```

Data output | Сообщения | Notifications

```
CALL

Запрос завершён успешно, время выполнения: 45 msec.
```

Для формирования общей выручки по продаже билетов за сутки.

```
create or replace function
"4sem".price_sum()
returns table (count_t integer, sum_t integer)
language sql
as $$
select count(ticket_number), sum(ticket_price)
from "4sem".ticket, "4sem".place
where (ticket.data_of_sale = current_date - 1) and (ticket.id_place = place.id_place)
$$;
```

Data output | Сообщения | Notifications

```
CREATE FUNCTION

Запрос завершён успешно, время выполнения: 2 secs 706 msec.
```

```
1  select * from "4sem".price_sum()
```

Data output | Сообщения | Notifications

| | count_t integer | sum_t integer |
|---|---|---|
| 1 | 1 | 1200 |

Триггер

```
create or replace function "4sem".add_to_log() returns
trigger as $$
declare
mstr character(20);
astr character(100);
retstr character(254);
BEGIN
if TG_OP = 'INSERT' then
astr = new;
mstr := 'Add new';
retstr := mstr || astr;
insert into
"4sem".logs("info",log_time) values
(retstr, NOW());
return new;
elsif TG_OP = 'UPDATE' then
astr = new;
mstr := 'Update';
retstr := mstr || astr;
INSERT INTO
"4sem".logs("info", log_time) values
(retstr, NOW());
Return new;
elsif TG_OP = 'DELETE' then
astr = old;
```

```
astr = old;
mstr := 'Remove';
retstr := mstr || astr;
insert into
"4sem".logs("info", log_time) values
(retstr, NOW());
return old;
end if;
end;
$$ language plpgsql;
```

Data output | Сообщения | Notifications

```
CREATE FUNCTION

Запрос завершён успешно, время выполнения: 72 msec.
```

```
1  create trigger t_destination after insert or update
2  or delete on "4sem".destination
3  for each row execute procedure "4sem".add_to_log();
```

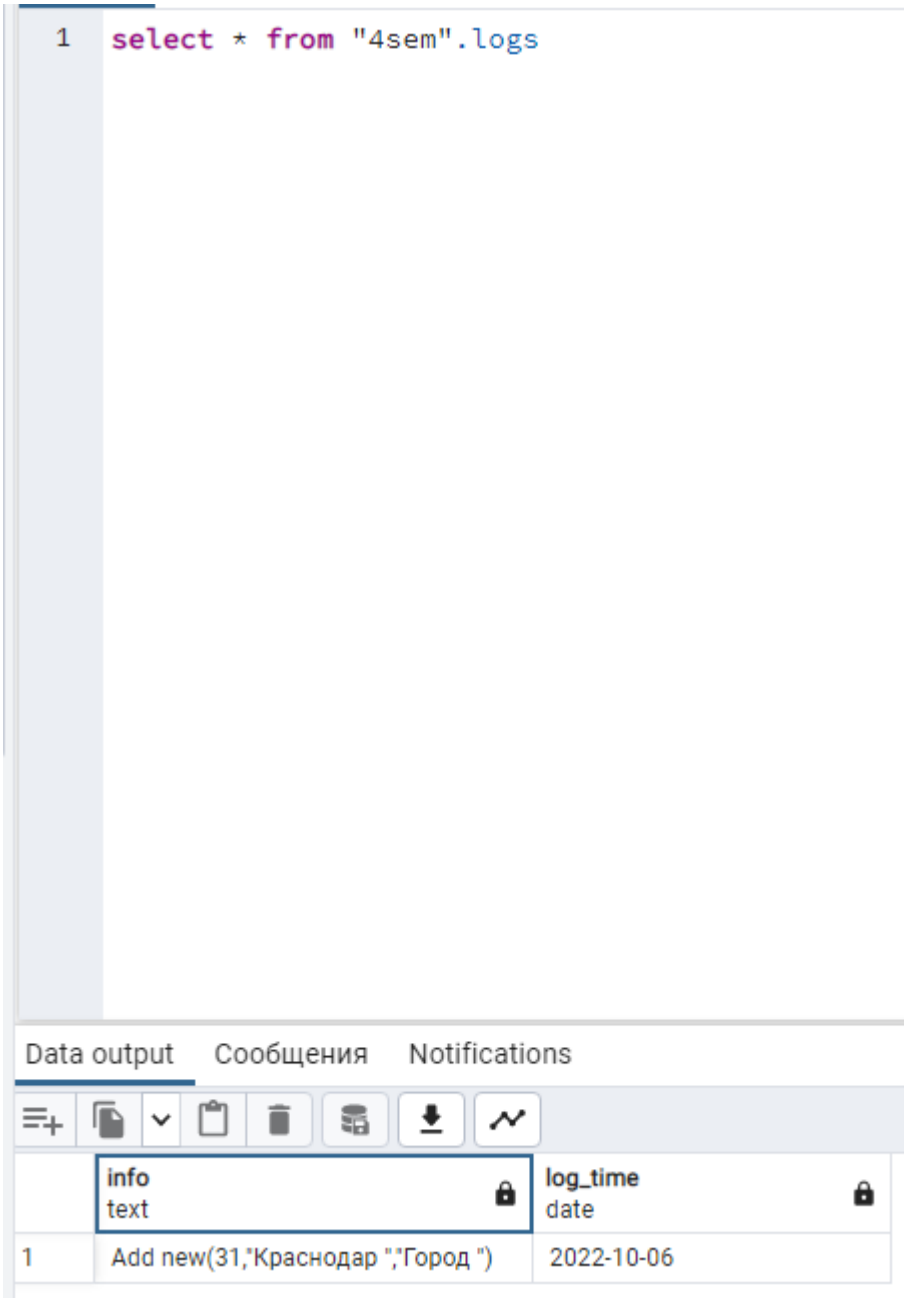
Data output | Сообщения | Notifications

```
CREATE TRIGGER

Запрос завершён успешно, время выполнения: 48 msec.
```

```sql
insert into "4sem".destination (id_destination, item_name, sattelment_category)
values (31, 'Краснодар', 'Город')
```

Data output | Сообщения | Notifications

```
INSERT 0 1

Запрос завершён успешно, время выполнения: 47 msec.
```




Вывод: я овладел практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.