

**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»

Автор: Павлишина Ирина Романовна

Факультет: ИКТ

Группа: K32391

Преподаватель: Говорова М. М.

Дата: 17.05.2023

ИТМО

Санкт-Петербург 2023

Функции

Для поиска билетов в заданный пункт назначения.

```
CREATE OR REPLACE FUNCTION all_flights.get_ticket_to (airport_name character varying)
RETURNS TABLE(ticket_num character varying)
AS
$$
    SELECT ticket_num
    FROM all_flights.ticket t
        JOIN all_flights.flight f ON f.id = t.id_flight
        JOIN all_flights.schedule s ON s.id = f.id_schedule
        JOIN all_flights.airport a ON a.id = s.id_airport
    WHERE a.name = airport_name;
$$
language sql;
```

```
select all_flights.get_ticket_to('airport_100');
```

	ticket_num character varying (20) 🔒
1	ticket_num_580
2	ticket_num_306
3	ticket_num_995
4	ticket_num_373
5	ticket_num_988

Создания новой кассы продажи билетов.

Запрос:

```
SELECT * FROM all_flights.cash_register
ORDER BY id DESC
LIMIT 2;
```

	id [PK] integer	address character varying (100)	city character varying (30)
1	400	address_399	city_80
2	399	address_398	city_49

Функция:

```
CREATE OR REPLACE FUNCTION all_flights.create_cash_register (_address character
varying, _city character varying)
```

```
RETURNS void
```

```
AS
```

```
$$
```

```
INSERT INTO all_flights.cash_register (address, city)
```

```
VALUES(_address, _city);
```

```
$$
```

```
language sql;
```

```
select all_flights.create_cash_register('test_address_func', 'test_city_func');
```

Data Output			
	id [PK] integer	address character varying (100)	city character varying (30)
1	404	test_address_func	test_city_func
2	400	address_399	city_80

Определить расход топлива по всем маршрутам за истекший месяц.

```
WHERE
```

```
s.arrive_date < '2024-05-01' AND s.departure_date > '2024-04-01';
```

Функция:

```
CREATE OR REPLACE FUNCTION all_flights.count_fuel_consumption(start_date character
varying, end_date character varying)
```

```
RETURNS TABLE(fuel_consumption_throught_the_distance integer)
```

```
AS
```

```
$$
```

```

SELECT b.fuel_consumption * s.distance
FROM all_flights.flight f
      JOIN all_flights.schedule s ON f.id_schedule = s.id
      JOIN all_flights.plane p ON f.id_plane = p.id
      JOIN all_flights.brand b ON p.id_brand = b.id
WHERE
      s.arrive_date < CAST(end_date AS DATE) AND s.departure_date >
CAST(start_date AS DATE);
$$
language sql;

```

```
select all_flights.count_fuel_consumption('2024-04-01', '2024-05-01');
```

Data Output	
	fuel_consumption integer
1	271244
2	295450
3	206492
4	107502
5	165452
6	367612

Триггер

Добавим проверку для билетов: невозможно добавить билет к уже прошедшему или уже начавшемуся полету; невозможно добавить билет, если полет к которому привязано место, и полет, к которому привязан билет, не совпадают.

```

CREATE OR REPLACE FUNCTION ticket_insert()
RETURNS TRIGGER AS
$$
BEGIN
      IF NOT EXISTS(
      SELECT 1
      FROM all_flights.flight f

```

```

        WHERE f.id = NEW.id_flight
        AND (date_departure < '2024-05-12')
    )THEN
        RAISE EXCEPTION 'This flight had already happened';
    END IF;

    IF NOT EXISTS(
        SELECT 1
        FROM all_flights.flight f
        JOIN all_flights.place p ON p.id = NEW.id_place
        WHERE f.id = NEW.id_flight
        AND p.id_flight = f.id
    )THEN
        RAISE EXCEPTION 'This place does not exist fot this flight';
    END IF;

    RETURN NEW;
END;
$$
LANGUAGE plpgsql;

```

```

CREATE TRIGGER ticket_insert
BEFORE INSERT ON all_flights.ticket
FOR EACH ROW
EXECUTE FUNCTION ticket_insert();

```

Проверка:

[Query](#)
[Query History](#)
[Messages](#)
[Notifications](#)

ERROR: ОШИБКА: This flight had already happened
 CONTEXT: функция PL/pgSQL ticket_insert(), строка 9, оператор RAISE

SQL state: P0001

```
ERROR: ОШИБКА: This place does not exist fot this flight
CONTEXT:  функция PL/pgSQL ticket_insert(), строка 19, оператор RAISE
```

```
INSERT 0 1
```

```
Query returned successfully in 98 msec.
```

Выводы

В результате выполнения лабораторной работы были достигнуты следующие цели:

- Овладение практическими навыками создания процедур, функций и триггеров в базе данных PostgreSQL
- Создание процедур и функций в соответствии с индивидуальным заданием, что позволило эффективно реализовать определенную логику обработки данных
- Создание триггеров для логирования событий вставки, что позволило улучшить контроль за изменениями в базе данных и обеспечить более точную отчетность.

Таким образом, выполнение лабораторной работы позволило успешно освоить необходимые навыки работы с базой данных PostgreSQL и использовать их для решения задач по обработке, хранению и анализу данных.