# Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение высшего образования «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

### ОТЧЕТ

## ПО ЛАБОРАТОРНОЙ РАЬОТЕ №3

«Процедуры, функции, триггеры в PostgresSQL»

Автор: Павлишина Ирина Романовна

Факультет: ИКТ

Группа: К32391

Преподаватель: Говорова М. М.

Дата: 17.05.2023



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;
```



#### Функция:

CREATE OR REPLACE FUNCTION all\_flights.create\_cash\_register (\_address character varying, city character varying)

**RETURNS** void

AS

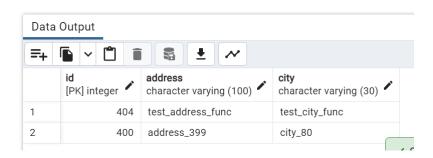
\$\$

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');



#### 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)

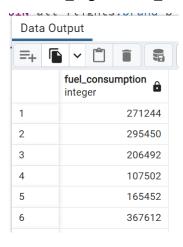
RETURNS TABLE(fuel consumption throught the distance integer)

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);
```

select all flights.count fuel consumption('2024-04-01', '2024-05-01');



# Триггер

**BEGIN** 

\$\$

language sql;

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 \$\$

> 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
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.