

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

**Отчет**  
по лабораторной работе №5 «Запросы на выборку и модификацию данных.  
Представления. Работа с индексами»  
по дисциплине **«Проектирование и реализация баз данных»**

Автор: Петухов С.А.

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова М.М.

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

## Практическое задание:

Вариант 1 (max - 6 баллов)

Создать процедуры/функции согласно индивидуальному заданию (часть 4).

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.

## Выполнение:

Процедуры/Функции

1) Поиск билетов в заданный пункт назначения.

```
CREATE OR REPLACE FUNCTION public.GetTicketsByDestination(  
    destination_city character varying(50)  
)  
RETURNS TABLE (  
    ticket_number integer,  
    flight_id integer,  
    schedule_id integer,  
    paper numeric(11,0),  
    service_number integer,  
    registration_status character varying(20),  
    seat character varying(4),  
    price numeric,  
    seat_type character varying(10),  
    purchase_type character varying(10)  
)  
LANGUAGE plpgsql  
AS $$  
BEGIN  
    RETURN QUERY  
    SELECT  
        t.ticket_number,  
        t.flight_id,  
        t.schedule_id,  
        t.paper,  
        t.service_number,  
        t.registration_status,  
        t.seat,  
        t.price,  
        t.seat_type,  
        t.purchase_type  
    FROM  
        public."Ticket" t  
    JOIN
```

```

        public."Flight" f ON t.flight_id = f.flight_id
    JOIN
        public."Schedule" s ON f.schedule_id = s.schedule_number
    JOIN
        public."Airport" a ON s.arrival_airport_number = a.airport_number
    WHERE
        a.address ~* destination_city;
END;
$$;

SELECT * FROM public.GetTicketsByDestination('Москва');

```

```

Airport=# SELECT * FROM public.GetTicketsByDestination('Москва');

```

ticket_number	flight_id	schedule_id	paper	service_number	registration_status	seat	price	seat_type	purchase_type
13	9	55	6168487494	31	Зарегистрирован	6	24173	эконом	Онлайн
14	9	5	2240009013	32	В обработке	9	47788	эконом	Онлайн
24	46	67	5643865474	52	В обработке	3	7886	люкс	Офлайн
33	41	65	9890189749	41	Зарегистрирован	7	28698	бизнес	Онлайн
47	41	12	9774019249	40	Зарегистрирован	3	45735	люкс	Онлайн
49	24	41	9471767762	33	Зарегистрирован	14	16674	бизнес	Онлайн
53	24	65	6777023813	5	Зарегистрирован	8	5712	бизнес	Офлайн
57	46	40	2543915182	55	Зарегистрирован	14	18701	эконом	Онлайн
58	46	39	5700065600	4	Зарегистрирован	15	30078	люкс	Онлайн

(9 строк)

## 2) Создания новой кассы продажи билетов.

```

CREATE OR REPLACE PROCEDURE public.CreateSalesCash(
    IN address_input character varying(50)
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO public."Sales_cash" ( address)
    VALUES (address_input);

    RAISE NOTICE 'Создана новая касса продажи билетов с адресом: %',
address_input;
END;
$$;

CALL public.CreateSalesCash('Г.Санкт-Петербург ул.Ломоносова 9');

```

```

Airport=# CALL public.CreateSalesCash('Г.Санкт-Петербург ул.Ломоносова 9');
ЗАМЕЧАНИЕ: Создана новая касса продажи билетов с адресом: Г.Санкт-Петербург ул.Ломоносова 9
CALL
Airport=#

```

## 3) Определить расход топлива по всем маршрутам за истекший месяц.

```

CREATE OR REPLACE FUNCTION
public.CalculateFuelConsumptionForPreviousMonth()
RETURNS TABLE (
    flight_id integer,
    on_board_number character varying(20),
    schedule_id integer,
    fuel_consumption_on_flight integer
)
LANGUAGE plpgsql
AS $$
BEGIN
    RETURN QUERY

```

```

SELECT
    f.flight_id,
    f."on-board_number",
    f.schedule_id,
    a.fuel_consumption * s.in_flight_time AS
fuel_consumption_on_flight
FROM
    public."Flight" f
JOIN
    public."Plane" p ON f."on-board_number" = p."on-board_number"
JOIN
    public."Aircraft_model" a ON p.model_number = a.model_number
JOIN
    public."Schedule" s ON f.schedule_id = s.schedule_number
WHERE
    f.departure_time BETWEEN DATE_TRUNC('MONTH', CURRENT_DATE -
INTERVAL '1 month') AND DATE_TRUNC('MONTH', CURRENT_DATE);
END;
$$;

SELECT * FROM public.CalculateFuelConsumptionForPreviousMonth();

```

```

Airport=# SELECT * FROM public.CalculateFuelConsumptionForPreviousMonth();
 flight_id | on_board_number | schedule_id | fuel_consumption_on_flight
-----+-----+-----+-----
      59 | ПУ-205         |          7 |              60
      87 | ПУ-205         |          4 |              75
      94 | HE-1488        |         58 |             336
     174 | HE-1488        |          6 |             147
     175 | HE-1488        |         73 |             399
     176 | ДК-715         |          6 |              56
     177 | ДК-715         |         73 |             152
     178 | ПУ-1933        |          6 |             140
     179 | ПУ-1933        |         73 |             380
     180 | ПУ-1944        |         73 |             380
     181 | ПУ-1944        |          6 |             140
     182 | ПУ-2349        |          6 |             140
     183 | ПУ-2349        |         73 |             380
     184 | ПУ-2490        |         73 |             380
     185 | ПУ-2490        |          6 |             140
     186 | ПУ-2663        |          6 |             140
     187 | ПУ-2663        |         73 |             380
     188 | ПУ-2856        |         73 |             380
     189 | ПУ-2856        |          6 |             140
     190 | ПУ-3538        |          6 |             140
     191 | ПУ-3538        |         73 |             380
     192 | ПУ-3894        |         73 |             380
     193 | ПУ-3894        |          6 |             140
     194 | ПУ-4052        |          6 |             140
     195 | ПУ-4052        |         73 |             380
     196 | ПУ-410         |         73 |             380
     197 | ПУ-410         |          6 |             140
     198 | ПУ-843         |          6 |             140
     199 | ПУ-843         |         73 |             380
(29 строк)

```

## Триггеры

```

CREATE OR REPLACE FUNCTION log_schedule_changes()

```

```

RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO public."Log_schedule" (
            "Event_Type",
            schedule_number,
            in_flight_time,
            departure_time,
            arrival_time,
            departure_airport_number,
            arrival_airport_number,
            "Edite_time"
        ) VALUES (
            'INSERT',
            NEW.schedule_number,
            NEW.in_flight_time,
            NEW.departure_time,
            NEW.arrival_time,
            NEW.departure_airport_number,
            NEW.arrival_airport_number,
            NOW()
        );
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO public."Log_schedule" (
            "Event_Type",
            schedule_number,
            in_flight_time,
            departure_time,
            arrival_time,
            departure_airport_number,
            arrival_airport_number,
            "Edite_time"
        ) VALUES (
            'UPDATE',
            NEW.schedule_number,
            NEW.in_flight_time,
            NEW.departure_time,
            NEW.arrival_time,
            NEW.departure_airport_number,
            NEW.arrival_airport_number,
            NOW()
        );
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO public."Log_schedule" (
            "Event_Type",
            schedule_number,
            in_flight_time,
            departure_time,
            arrival_time,
            departure_airport_number,
            arrival_airport_number,
            "Edite_time"
        ) VALUES (
            'DELETE',
            OLD.schedule_number,
            OLD.in_flight_time,
            OLD.departure_time,
            OLD.arrival_time,
            OLD.departure_airport_number,
            OLD.arrival_airport_number,
            NOW()
        );
    END IF;
    RETURN NULL;

```

```

END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER schedule_changes_trigger
AFTER INSERT OR UPDATE OR DELETE
ON public."Schedule"
FOR EACH ROW
EXECUTE FUNCTION log_schedule_changes();

```

```

Airport=# Select * from Public."Log_schedule";

```

Log_id	Event_Type	schedule_number	in_flight_time	departure_time	arrival_time	departure_airport_number	arrival_airport_number	Edit_time
6	INSERT	94	4	12:00:00+03	16:00:00+03	1	57	2023-12-07 14:38:36.701081
7	UPDATE	93	6	13:00:00+03	19:00:00+03	6	8	2023-12-07 14:38:49.242078
8	DELETE	94	4	12:00:00+03	16:00:00+03	1	57	2023-12-07 14:38:53.57128

(3 строки)

```

CREATE OR REPLACE FUNCTION log_flight_changes()

```

```

RETURNS TRIGGER AS $$

```

```

BEGIN

```

```

IF TG_OP = 'INSERT' THEN
    INSERT INTO public."Log_flight" (
        "Event_type",
        flight_id,
        "on-board_number",
        schedule_id,
        departure_time,
        arrival_time,
        status,
        "Edit_time"
    ) VALUES (
        'INSERT',
        NEW.flight_id,
        NEW."on-board_number",
        NEW.schedule_id,
        NEW.departure_time,
        NEW.arrival_time,
        NEW.status,
        NOW()
    );
ELSIF TG_OP = 'UPDATE' THEN
    INSERT INTO public."Log_flight" (
        "Event_type",
        flight_id,
        "on-board_number",
        schedule_id,
        departure_time,
        arrival_time,
        status,
        "Edit_time"
    ) VALUES (
        'UPDATE',
        NEW.flight_id,
        NEW."on-board_number",
        NEW.schedule_id,
        NEW.departure_time,
        NEW.arrival_time,
        NEW.status,
        NOW()
    );
ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO public."Log_flight" (
        "Event_type",
        flight_id,
        "on-board_number",
        schedule_id,
        departure_time,
        arrival_time,
        status,

```

```

        "Edit_time"
    ) VALUES (
        'DELETE',
        OLD.flight_id,
        OLD."on-board_number",
        OLD.schedule_id,
        OLD.departure_time,
        OLD.arrival_time,
        OLD.status,
        NOW()
    );
END IF;
RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER flight_changes_trigger
AFTER INSERT OR UPDATE OR DELETE
ON public."Flight"
FOR EACH ROW
EXECUTE FUNCTION log_flight_changes();

```

```

Airport=# Select * from Public."Log_flight";
 log_id | Event_type | flight_id | on-board_number | schedule_id | departure_time | arrival_time | status | Edit_time
-----+-----+-----+-----+-----+-----+-----+-----+-----
      3 | UPDATE    |      26 | ПУ-843          |          84 | 2020-09-08    | 2023-12-27 | Регистрация | 2023-12-07 15:11:34.572855
      4 | INSERT    |      203 | ПУ-1667         |          40 | 2022-06-01    | 2022-06-01 | Завершён   | 2023-12-07 15:11:34.572855
(2 строки)

```

Выводы:

В данной работе были рассмотрены способы создание функций и процедур. Были созданы триггеры для логирования изменений в таблице schedule.