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

Отчет

по лабораторной работе №5 «Процедуры, функции, триггеры в PostgreSQL»

по дисциплине **«Проектирование и реализация баз данных»**

Автор: Хурс П.И

Факультет: ИКТ

Группа: К3241

Преподаватель: Говорова М.М.

Санкт-Петербург 2023

Оглавление

1. Создать процедуры согласно индивидуальному заданию 3
2. Создать необходимый триггер 6
Вывод 6

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:

1. Создать процедуры/функции 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.

1. Создать процедуры согласно индивидуальному заданию

1. Для создания нового рейса на поезд.

```
CREATE OR REPLACE PROCEDURE create_new_train_trip(
    _train_id integer,
    _schedule_id integer,
    _departure_date date,
    _arrival_date date,
    _train_status varchar)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO public.train(train_id, shedule_id, departure_date,
arrival_date, train_status)
    VALUES (_train_id, _schedule_id, _departure_date, _arrival_date,
_train_status);
END;
$$;
```

До:

```
pashahurs=# SELECT COUNT(*) FROM public.train;
 count
-------
    17
(1 row)
```

После:

```
pashahurs=# call create_new_train_trip(123, 2, '2023-01-01', '2023-01-02', 'Active');
CALL
pashahurs=# SELECT COUNT(*) FROM public.train;
 count
-------
    18
(1 row)
```

2. Для формирования общей выручки по продаже билетов за сутки.

```
CREATE OR REPLACE FUNCTION
total_ticket_sales_revenue_for_day(sale_date_param date)
```

```
RETURNS bigint
LANGUAGE plpgsql AS
$$
DECLARE
    total_revenue bigint;
BEGIN
    SELECT SUM(t.price) INTO total_revenue
    FROM public.ticket AS t
    JOIN public.seat AS s ON t.seat_id = s.seat_id
    JOIN public.carriage AS c ON s.carriage_id = c.carriage_id
    JOIN public.train AS tr ON c.train_id = tr.train_id
    WHERE tr.departure_date = sale_date_param;

    RETURN total_revenue;
END;
$$;
```

```
pashahurs=# SELECT total_ticket_sales_revenue_for_day('2023-12-07');
 total_ticket_sales_revenue_for_day
------------------------------------
                                260
```

3. Для вычисления количества автомобилей заданной марки.

```
CREATE OR REPLACE PROCEDURE increase_commuter_train_prices()
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE public.ticket
    SET price = price * 1.2
    FROM public.seat, public.carriage, public.train, public.schedule
    WHERE
        public.ticket.seat_id = public.seat.seat_id AND
        public.seat.carriage_id = public.carriage.carriage_id AND
        public.carriage.train_id = public.train.train_id AND
        public.train.shedule_id = public.schedule.schedule_id AND
        public.schedule.train_type = 'Commuter';
END;
$$;
```

```
pashahurs-# WHERE s.carriage_id = 1;WHERE s.carriage_id = 1;
 ticket_id | price
-----------+-------
         1 |   100
       101 |   100
         5 |   100
       102 |   100
       117 |     2
       118 |     3
       119 |     4
       120 |     5
       121 |     6
       122 |     7
       123 |     8
       124 |     9
       125 |    10
       126 |    11
       127 |    12
       128 |    13
       129 |    14
       130 |    15
       131 |    16
       132 |    17
       133 |    18
       134 |    19
       135 |    20
       136 |    21
       137 |    22
       138 |    23
       139 |    24
       140 |    25
       141 |    26
       142 |    27
       143 |    28
       144 |    29
       145 |    30
       146 |    31
       147 |    32
       148 |    33
       149 |    34
       150 |    35
       151 |    36
       152 |    37
       153 |    38
       154 |    39
       155 |    40
       156 |    41
```

после

```
pashahurs=# SELECT t.ticket_id, t.price
FROM public.ticket t
JOIN public.seat s ON t.seat_id = s.seat_id
WHERE s.carriage_id = 1;
 ticket_id | price
-----------+-------
       101 |   120
         1 |   120
         5 |   120
       102 |   120
       117 |     2
       118 |     4
       119 |     5
       120 |     6
       121 |     7
       122 |     8
       123 |    10
       124 |    11
       125 |    12
       126 |    13
       127 |    14
       128 |    16
       129 |    17
       130 |    18
       131 |    19
       132 |    20
       133 |    22
       134 |    23
       135 |    24
       136 |    25
       137 |    26
       138 |    28
       139 |    29
       140 |    30
       141 |    31
       142 |    32
       143 |    34
       144 |    35
       145 |    36
       146 |    37
       147 |    38
       148 |    40
       149 |    41
       150 |    42
       151 |    43
       152 |    44
       153 |    46
```

2. Создать необходимый триггер

Триггер будет автоматически обновлять статус автомобиля на "недоступен" после того, как контракт на его аренду будет создан. Это логичное действие, поскольку автомобиль, который уже арендован, не должен быть доступен для новых аренд до возврата.

```
CREATE TABLE IF NOT EXISTS audit_log (
    log_id SERIAL PRIMARY KEY,
    table_name VARCHAR(255),
    action VARCHAR(50),
    original_data TEXT,
    new_data TEXT,
    log_timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

CREATE OR REPLACE FUNCTION universal_audit_trigger()
RETURNS TRIGGER AS $$
BEGIN
    IF TG_OP = 'INSERT' THEN
        INSERT INTO audit_log (table_name, action, new_data)
        VALUES (TG_TABLE_NAME, 'INSERT', row_to_json(NEW));
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        INSERT INTO audit_log (table_name, action, original_data)
        VALUES (TG_TABLE_NAME, 'DELETE', row_to_json(OLD));
        RETURN OLD;
    ELSIF TG_OP = 'UPDATE' THEN
        INSERT INTO audit_log (table_name, action, original_data,
new_data)
        VALUES (TG_TABLE_NAME, 'UPDATE', row_to_json(OLD),
row_to_json(NEW));
        RETURN NEW;
    END IF;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER carriage_audit_trigger
AFTER INSERT OR DELETE OR UPDATE ON carriage
FOR EACH ROW EXECUTE FUNCTION universal_audit_trigger();
```

```
pashahurs=# SELECT * FROM audit_log;
 log_id | table_name | action | original_data | new_data | log_timestamp
--------+------------+--------+---------------+----------+---------------
(0 rows)
```

```
pashahurs=# SELECT * FROM audit_log;
 log_id | table_name | action |                                original_data                                  |
                        new_data                                        |       log_timestamp
--------+------------+--------+-------------------------------------------------------------------------------+------
------------------------------------------------------------------------+----------------------------
      1 | carriage   | INSERT |                                                                               | {"car
riage_id":49,"train_id":1,"carriage_type":"Test Type","carriage_number":29}         | 2023-12-17 22:05:27.331668
      2 | carriage   | UPDATE | {"carriage_id":49,"train_id":1,"carriage_type":"Test Type","carriage_number":29}         | {"car
riage_id":49,"train_id":1,"carriage_type":"Updated Test Type","carriage_number":29} | 2023-12-17 22:07:48.122829
      3 | carriage   | DELETE | {"carriage_id":49,"train_id":1,"carriage_type":"Updated Test Type","carriage_number":29} |
                                                                                    | 2023-12-17 22:08:04.213972
(3 rows)
```

Вывод

В ходе лабораторной работы были успешно созданы и протестированы процедуры и триггеры в системе управления базами данных PostgreSQL. Работа позволила овладеть практическими навыками программирования в SQL и пониманием механизма триггеров.