

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: Пузенко А.А.

Факультет: ИКТ

Группа: К3240

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

| | |
|----------------------|----|
| Цель работы | 3 |
| Практическое задание | 3 |
| Выполнение: | 3 |
| Вывод: | 11 |

Цель работы

Овладеть практическими создания и использования процедур, функций и триггеров в базе данных PostgreSQL.

Практическое задание

Вариант 1 (max - 6 баллов)

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.

Вариант 2 (max - 8 баллов)

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

2.1. Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу).

2.2. Создать авторский триггер по варианту индивидуального задания.

Указание. Работа выполняется в консоли SQL Shell (psql).

Выполнение:

Вариант 20 “Автозаправки”

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

- Вывести сведения обо всех покупках одного из клиентов за заданную дату (данные клиента, дата, объем топлива, уплаченная сумма).

```
1 CREATE OR REPLACE FUNCTION get_client_purchases(  
2     client_id_param INTEGER,  
3     purchase_date_param DATE  
4 )  
5 RETURNS TABLE (  
6     client_full_name CHARACTER VARYING(500),  
7     purchase_date DATE,  
8     fuel_quantity DOUBLE PRECISION,  
9     paid_amount DOUBLE PRECISION  
10 ) AS $$  
11 BEGIN  
12     RETURN QUERY  
13     SELECT c.full_name AS client_full_name,  
14            DATE(p.purchase_date) AS purchase_date,  
15            p.amount_fuel AS fuel_quantity,  
16            p.amount_fuel * fs.price AS paid_amount  
17     FROM clients c  
18     JOIN cards crd ON c.id_client = crd.id_client  
19     JOIN purchases p ON crd.id_card = p.id_card  
20     JOIN fuels_sold fs ON p.id_fuel_sold = fs.id_fuel_sold  
21     WHERE c.id_client = client_id_param  
22            AND DATE(p.purchase_date) = purchase_date_param;  
23 END;  
24 $$ LANGUAGE plpgsql;
```

Query

Query History

1

SELECT * FROM get_client_purchases(96, '2024-01-19')

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

| | client_full_name character varying 🔒 | purchase_date date 🔒 | fuel_quantity double precision 🔒 | paid_amount double precision 🔒 |
|---|---|-------------------------|-------------------------------------|-----------------------------------|
| 1 | Sergey Chernov | 2024-01-19 | 14 | 792.4 |
| 2 | Sergey Chernov | 2024-01-19 | 15 | 849 |

- Количество видов топлива, поставляемых каждой фирмой-поставщиком.

| Query | Query History |
|-------|--|
| 1 | CREATE OR REPLACE FUNCTION count_fuels_per_supplier() |
| 2 | RETURNS TABLE (supplier_name TEXT, number_of_fuels INT) |
| 3 | AS \$\$ |
| 4 | BEGIN |
| 5 | RETURN QUERY |
| 6 | SELECT |
| 7 | sf.name AS supplier_name, |
| 8 | COUNT (DISTINCT f.mark) AS number_of_fuels |
| 9 | FROM |
| 10 | supplier_firms sf |
| 11 | JOIN |
| 12 | gas_stations gs ON sf.id_firm = gs.id_firm |
| 13 | JOIN |
| 14 | fuels_sold fs ON gs.id_station = fs.id_station |
| 15 | JOIN |
| 16 | fuels f ON fs.id_fuel = f.id_fuel |
| 17 | GROUP BY |
| 18 | sf.name; |
| 19 | END; |
| 20 | \$\$ LANGUAGE plpgsql; |

Query

Query History

1

SELECT * FROM count_fuels_per_supplier()

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑

🗄

⬇

📈

| | supplier_name character varying 🔒 | number_of_fuels bigint 🔒 |
|---|--------------------------------------|-----------------------------|
| 1 | Газпром нефть | 4 |
| 2 | ЛУКОЙЛ | 3 |
| 3 | РОСНЕФТЬ | 5 |

- Самый непопулярный вид топлива за прошедшую неделю.

| Query | Query History |
|--|---------------|
| <pre>1 CREATE OR REPLACE FUNCTION least_popular_fuel_last_week() 2 RETURNS TABLE (3 unpopular_fuel VARCHAR(50), 4 sales_count DOUBLE PRECISION 5) AS \$\$ 6 BEGIN 7 RETURN QUERY 8 SELECT f.type AS fuel_type, 9 COALESCE(SUM(p.amount_fuel), 0.0) AS total_sales 10 FROM fuels f 11 LEFT JOIN fuels_sold fs ON f.id_fuel = fs.id_fuel 12 LEFT JOIN purchases p ON fs.id_fuel_sold = p.id_fuel_sold 13 AND p.purchase_date >= (CURRENT_DATE - INTERVAL '1 week') 14 GROUP BY f.type 15 HAVING COALESCE(SUM(p.amount_fuel), 0.0) = (16 SELECT MIN(total_sales) 17 FROM (18 SELECT COALESCE(SUM(p.amount_fuel), 0.0) AS total_sales 19 FROM fuels f 20 LEFT JOIN fuels_sold fs ON f.id_fuel = fs.id_fuel 21 LEFT JOIN purchases p ON fs.id_fuel_sold = p.id_fuel_sold 22 AND p.purchase_date >= (CURRENT_DATE - INTERVAL '1 week') 23 GROUP BY f.type 24) AS min_sales 25); 26 END; 27 \$\$ LANGUAGE plpgsql;</pre> | |

Query

Query History

1

SELECT * FROM least_popular_fuel_last_week();

Data Output

Messages

Notifications

≡+

📄

▼

📋

▼

🗑️

🗄️

⬇️

📈

| | unpopular_fuel character varying | sales_count double precision |
|---|-------------------------------------|---------------------------------|
| 1 | дизель | 0 |
| 2 | газ | 0 |
| 3 | бензин | 0 |
| 4 | мазут | 0 |

- Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL

```
1 CREATE TABLE public.data_changes_log (  
2     id_event SERIAL PRIMARY KEY,  
3     event_type VARCHAR(50),  
4     event_time TIMESTAMP DEFAULT CURRENT_TIMESTAMP,  
5     table_name VARCHAR(100),  
6     key_column_value TEXT,  
7     user_name VARCHAR(100)  
8 );  
9  
10 CREATE OR REPLACE FUNCTION log_data_changes()  
11 RETURNS TRIGGER AS $$  
12 BEGIN  
13     IF TG_OP = 'INSERT' THEN  
14         INSERT INTO public.data_changes_log (event_type, table_name, key_column_value, user_name)  
15         VALUES ('INSERT', TG_TABLE_NAME, NEW.*::TEXT, current_user);  
16     ELSIF TG_OP = 'UPDATE' THEN  
17         INSERT INTO public.data_changes_log (event_type, table_name, key_column_value, user_name)  
18         VALUES ('UPDATE', TG_TABLE_NAME, OLD.*::TEXT || ' --> ' || NEW.*::TEXT, current_user);  
19     ELSIF TG_OP = 'DELETE' THEN  
20         INSERT INTO public.data_changes_log (event_type, table_name, key_column_value, user_name)  
21         VALUES ('DELETE', TG_TABLE_NAME, OLD.*::TEXT, current_user);  
22     END IF;  
23     RETURN NULL;  
24 END;  
25 $$ LANGUAGE plpgsql;  
26
```



```

27 CREATE TRIGGER trg_log_clients
28 AFTER INSERT OR UPDATE OR DELETE ON public.clients
29 FOR EACH ROW EXECUTE FUNCTION log_data_changes();
30
31 CREATE TRIGGER trg_log_positions
32 AFTER INSERT OR UPDATE OR DELETE ON public.positions
33 FOR EACH ROW EXECUTE FUNCTION log_data_changes();
34
35 CREATE TRIGGER trg_log_fuels
36 AFTER INSERT OR UPDATE OR DELETE ON public.fuels
37 FOR EACH ROW EXECUTE FUNCTION log_data_changes();
38
39 CREATE TRIGGER trg_log_supplier_firms
40 AFTER INSERT OR UPDATE OR DELETE ON public.supplier_firms
41 FOR EACH ROW EXECUTE FUNCTION log_data_changes();
42
43 CREATE TRIGGER trg_log_cards
44 AFTER INSERT OR UPDATE OR DELETE ON public.cards
45 FOR EACH ROW EXECUTE FUNCTION log_data_changes();
46
47 CREATE TRIGGER trg_log_gas_stations
48 AFTER INSERT OR UPDATE OR DELETE ON public.gas_stations
49 FOR EACH ROW EXECUTE FUNCTION log_data_changes();
50
51 CREATE TRIGGER trg_log_employees
52 AFTER INSERT OR UPDATE OR DELETE ON public.employees
53 FOR EACH ROW EXECUTE FUNCTION log_data_changes();
54
55 CREATE TRIGGER trg_log_fuels_sold
56 AFTER INSERT OR UPDATE OR DELETE ON public.fuels_sold
57 FOR EACH ROW EXECUTE FUNCTION log_data_changes();
58
59 CREATE TRIGGER trg_log_purchases
60 AFTER INSERT OR UPDATE OR DELETE ON public.purchases
61 FOR EACH ROW EXECUTE FUNCTION log_data_changes();

```

Query

Query History

Scratch Pad

×

1 SELECT * FROM public.data_changes_log

2 ORDER BY id_event ASC

Data Output

Messages

Notifications

≡

📄

▼

📋

▼

🗑

🗄

⬇

📈

| | id_event [PK] integer | event_type character vary | event_time timestamp without time z | table_name character varying (100) | key_column_value text | user_name character varying (100) |
|---|--------------------------|------------------------------|--|---------------------------------------|--|--------------------------------------|
| 1 | 1 | DELETE | 2024-03-04 11:21:46... | clients | (101,"Ivan Testovich",89990001122,Moscow) | postgres |
| 2 | 2 | INSERT | 2024-03-04 11:22:17... | clients | (102,"Ivan Testovich",89998887766,Moscow) | postgres |
| 3 | 3 | UPDATE | 2024-03-04 11:22:25... | clients | (102,"Ivan Testovich",89998887766,Moscow) --> (102,"Ivan Testovich",899988866... | postgres |
| 4 | 4 | INSERT | 2024-03-04 11:25:45... | fuels | (21,АИ-93,бензин,литр) | postgres |
| 5 | 5 | DELETE | 2024-03-04 11:25:49... | fuels | (21,АИ-93,бензин,литр) | postgres |

- Создать авторский триггер по варианту индивидуального задания.

Триггер обновляющий баланс на карте при добавлении новой покупки в таблицу

| | |
|--|---------------|
| Query | Query History |
| <pre> 1 CREATE OR REPLACE FUNCTION update_card_balance() 2 RETURNS TRIGGER AS \$\$ 3 DECLARE 4 fuel_price NUMERIC; 5 BEGIN 6 SELECT fs.price 7 INTO fuel_price 8 FROM fuels_sold fs 9 WHERE fs.id_fuel_sold = NEW.id_fuel_sold; 10 11 UPDATE cards 12 SET balance = balance - NEW.amount_fuel * fuel_price 13 WHERE id_card = NEW.id_card; 14 15 RETURN NEW; 16 END; 17 \$\$ LANGUAGE plpgsql; 18 19 CREATE TRIGGER update_card_balance_trigger 20 AFTER INSERT ON purchases 21 FOR EACH ROW 22 EXECUTE FUNCTION update_card_balance(); </pre> | |

Результат работы

| | | | |
|----|----|-----|------|
| 15 | 24 | 102 | 7327 |
|----|----|-----|------|

Query

Query History

1

INSERT INTO purchases(id_card, id_employee, id_fuel_sold, amount_fuel, purchase_date)

2

VALUES (24, 23, 102, 11, '2024-01-05')

Data Output

Messages

Notifications

INSERT 0 1

y

Query History

SELECT * FROM cards WHERE id_card = 24;

Output

Messages

Notifications

| id_card [PK] bigint | id_client bigint | balance double precision |
|------------------------|---------------------|-----------------------------|
| 24 | 102 | 6581.2 |

Вывод:

В данной лабораторной работе я освоил навыки создания и использования процедур, функций и триггеров в PostgreSQL.