

САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №3
по курсу «Проектирование и реализация баз данных»
Тема: Процедуры, функции, триггеры в PostgreSQL
Вариант: 1

Выполнил:
Седельников П.В.
К32401

Проверила:
Говорова М.М.

Санкт-Петербург
2023 г.

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

Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию и (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. Процедуры

1.1. Создайте хранимую процедуру для снижения цены на заданный процент для товаров, у которых срок пребывания на складе превысил заданный норматив.

CREATE PROCEDURE set_discount(days integer, percentages integer)

LANGUAGE SQL

AS \$\$

*UPDATE products SET cost = cost * (1 - CAST(percentages AS FLOAT) /*

CAST(100 AS FLOAT)) WHERE id IN (SELECT DISTINCT(product_id)

FROM delivery_items WHERE delivery_id IN (SELECT id FROM deliveries

WHERE create_date::date < CURRENT_DATE - days) AND remain != 0);

\$\$;

```
postgres=# CREATE PROCEDURE set_discount(days integer, percentages integer)
postgres=# LANGUAGE SQL
postgres=# AS $$
postgres$# UPDATE products SET cost = cost * (1 - CAST(percentages AS FLOAT) / CAST(100 AS FLOAT)) WHERE id IN (SELECT D
ISTINCT(product_id) FROM delivery_items WHERE delivery_id IN (SELECT id FROM deliveries WHERE create_date::date < CURREN
T_DATE - days AND status = 'Создан') AND remain != 0);
postgres$# $$;
CREATE PROCEDURE
postgres=# SELECT id, cost FROM products WHERE id IN (SELECT DISTINCT(product_id) FROM delivery_items WHERE delivery_id
IN (SELECT id FROM deliveries WHERE create_date::date < CURRENT_DATE - 1 AND status = 'Создан') AND remain != 0);
 id | cost
-----+-----
  64 | 34595
   91 | 14595
 524 | 16745
 671 |  4745
(4 строки)

postgres=# CALL set_discount(1, 50);
CALL
postgres=# SELECT id, cost FROM products WHERE id IN (SELECT DISTINCT(product_id) FROM delivery_items WHERE delivery_id
IN (SELECT id FROM deliveries WHERE create_date::date < CURRENT_DATE - 1 AND status = 'Создан') AND remain != 0);
 id | cost
-----+-----
  64 | 17297.5
   91 |  7297.5
 524 |  8372.5
 671 |  2372.5
(4 строки)

postgres=#
```

1.2. Создайте хранимую процедуру для расчета стоимости всех партий товаров, проданных за прошедшие сутки.

CREATE PROCEDURE get_amount_for_sold_deliveries(days integer)

LANGUAGE SQL

AS \$\$

```
SELECT SUM(delivery_total_amount) FROM (SELECT SUM(amount*cost)
AS delivery_total_amount FROM delivery_items WHERE delivery_id IN
(SELECT sales_table.delivery_id FROM (SELECT delivery_id,
SUM(total_amount) AS sales_in_delivery FROM (SELECT delivery_item_id,
SUM(amount) AS total_amount FROM order_items WHERE order_id IN
(SELECT id FROM orders WHERE create_date > CURRENT_DATE - days)
GROUP BY delivery_item_id) AS items_table JOIN delivery_items ON
delivery_items.id = delivery_item_id GROUP BY delivery_id) AS sales_table
JOIN (SELECT delivery_id, COUNT(*) AS amount_in_delivery FROM
delivery_items GROUP BY delivery_id) AS amount_table ON
amount_table.delivery_id = sales_table.delivery_id WHERE
sales_in_delivery = amount_in_delivery)) AS result_table;
$$;
```

// нахожу все товары проданные за день -> нахожу количество
проданных товаров в каждой поставке за день -> нахожу количество
товаров в поставке -> если количество товаров в поставке и проданных
товаров в поставке за день одинаковое, считаю поставку проданной за
прошедший день -> нахожу сумму стоимости этих товаров

```
postgres=# CREATE PROCEDURE get_amount_for_sold_deliveries(days integer)
postgres=# LANGUAGE SQL
postgres=# AS $$
postgres$# SELECT SUM(delivery_total_amount) FROM (SELECT SUM(amount*cost) AS delivery_total_amount FROM delivery_items
WHERE delivery_id IN (SELECT sales_table.delivery_id FROM (SELECT delivery_id, SUM(total_amount) AS sales_in_delivery FR
OM (SELECT delivery_item_id, SUM(amount) AS total_amount FROM order_items WHERE order_id IN (SELECT id FROM orders WHERE
create_date > CURRENT_DATE - days) GROUP BY delivery_item_id) AS items_table JOIN delivery_items ON delivery_items.id =
delivery_item_id GROUP BY delivery_id) AS sales_table JOIN (SELECT delivery_id, COUNT(*) AS amount_in_delivery FROM del
ivery_items GROUP BY delivery_id) AS amount_table ON amount_table.delivery_id = sales_table.delivery_id WHERE sales_in_d
elivery = amount_in_delivery)) AS result_table;
postgres$# $$;
postgres=# CREATE PROCEDURE
postgres=# CALL get_amount_for_sold_deliveries(1)
postgres=# CALL get_amount_for_sold_deliveries(1);
ОШИБКА: ошибка синтаксиса (примерное положение: "CALL")
СТРОКА 2: CALL get_amount_for_sold_deliveries(1);
^
postgres=# CALL get_amount_for_sold_deliveries(1);
CALL
postgres=# SELECT get_amount_for_sold_deliveries(1);
ОШИБКА: get_amount_for_sold_deliveries(integer) – процедура
СТРОКА 1: SELECT get_amount_for_sold_deliveries(1);
^
ПОДСКАЗКА: Для вызова процедуры используйте CALL.
postgres=# CALL get_amount_for_sold_deliveries(1);
CALL
postgres=#
```

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.

```
CREATE OR REPLACE PROCEDURE add_action(tgop text, tgrelname text, tglevel text, tgrelid text)
LANGUAGE SQL
AS $$
INSERT INTO actions(tg_op, tg_relname, tg_level, tg_relid) VALUES(tgop, tgrelname, tglevel,
tgrelid);
$$;

CREATE OR REPLACE FUNCTION write_action()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
CALL add_action(TG_OP::text, TG_RELNAME::text, TG_LEVEL::text, TG_RELID::text);
RETURN NEW;
END;
$$;

CREATE OR REPLACE TRIGGER logging
AFTER UPDATE OR INSERT OR DELETE ON vendors
FOR EACH STATEMENT
EXECUTE FUNCTION write_action();
```

	id		name	
	[PK] integer		text	
1	14		Новый поставщик	
2	1		ИП Карелин Эдуард Викторович	
3	2		ООО "РДС МАРКЕТ"	

#	type	table	statement	id
8	UPDATE	vendors	STATEMENT	16466
9	DELETE	vendors	STATEMENT	16466
10	INSERT	vendors	STATEMENT	16466

Вывод: овладел практическими навыками создания представлений и запросов на выборку данных к базе данных PostgreSQL, использования подзапросов при модификации данных и индексов.