



**Министерство науки и высшего образования
Российской Федерации Федеральное государственное
бюджетное образовательное учреждение высшего
образования «Московский государственный
технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»
Кафедра ИУ5 «Системы обработки информации и управления»**

Курс «Базы данных»

Отчет по лабораторной работе №5

«Использование триггеров и хранимых процедур PostgreSQL»

Выполнил:
студент группы ИУ5-41Б
Кашима А.

Проверил:
преподаватель каф. ИУ5
Ковалева Н.А.

Москва, 2024 г.

Цель лабораторно работ:

изучить хранимые процедуры и триггеры в базах данных, приобрести практические навыки создания хранимых процедур и триггеров в среде PostgreSQL.

а. Пример из теоретической части:

```
1 CREATE OR REPLACE PROCEDURE InsertNewProducts()  
2 LANGUAGE SQL  
3 AS $$  
4 INSERT INTO products (product_name, price, description)  
5 VALUES  
6     ('Product 6', 19.99, 'Description of Product 6'),  
7     ('Product 7', 34.99, 'Description of Product 7'),  
8     ('Product 8', 42.99, 'Description of Product 8');  
9 $$;  
10
```

Data Output	Messages	Notifications
CREATE PROCEDURE		
Query returned successfully in 57 msec.		

CALL InsertNewProducts();

1. функция

а. Пример из теоретической части:

1)

```
CREATE OR REPLACE FUNCTION OrderPrice(x NUMERIC)
RETURNS TABLE(order_id INT, total NUMERIC) AS $$
BEGIN
    RETURN QUERY
    SELECT
        o.order_id,
        SUM(i.quantity * p.price) AS total
    FROM
        orders o
    JOIN
        items i ON o.order_id = i.order_id
    JOIN
        products p ON i.product_id = p.product_id
    GROUP BY
        o.order_id
    HAVING
        SUM(i.quantity * p.price) > x
    ORDER BY
        o.order_id;
END;
$$ LANGUAGE plpgsql;
```

```

18
19 CREATE OR REPLACE FUNCTION OrderPrice(x NUMERIC)
20 RETURNS TABLE(order_id INT, total NUMERIC) AS $$
21 BEGIN
22     RETURN QUERY
23     SELECT
24         o.order_id,
25         SUM(i.quantity * p.price) AS total
26     FROM
27         orders o
28     JOIN
29         items i ON o.order_id = i.order_id
30     JOIN
31         products p ON i.product_id = p.product_id
32     GROUP BY
33         o.order_id
34     HAVING
35         SUM(i.quantity * p.price) > x
36     ORDER BY
37         o.order_id;
38 END;
39 $$ LANGUAGE plpgsql;
40

```

Эта функция рассчитает общую цену для каждого заказа, умножив количество заказанных товаров на соответствующую цену за единицу, а затем суммируя эти значения. Он вернет order_id и общую цену для ордеров, общая цена которых превышает входной параметр x.

```

47
48 SELECT * FROM OrderPrice(15)
49

```

Data Output Messages Notifications



	order_id integer	total numeric
1	1	147.94
2	2	91.97
3	3	159.94

2)

CREATE OR REPLACE PROCEDURE

MeanValue (

Value1 REAL DEFAULT 0,

Value2 REAL DEFAULT 0,

Value3 REAL DEFAULT 0,

INOUT outputers REAL DEFAULT 0

)

LANGUAGE plpgsql

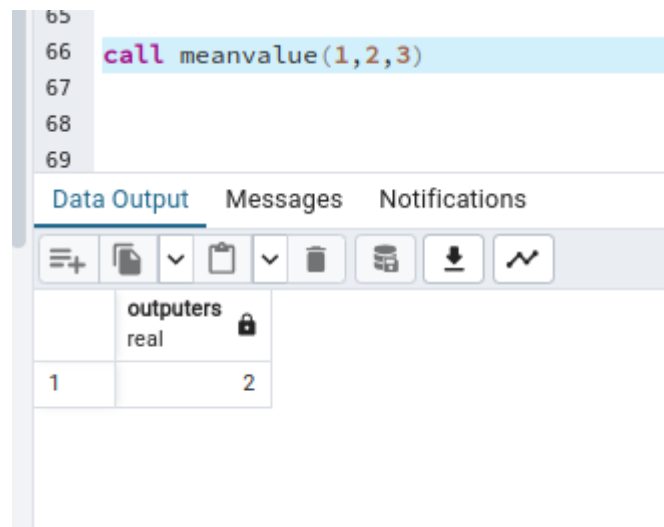
AS \$\$

BEGIN

SELECT (Value1 + Value2 + Value3) / 3 INTO outputers;

END; \$\$;

```
50
51 CREATE OR REPLACE PROCEDURE
52 MeanValue (
53     Value1 REAL DEFAULT 0,
54     Value2 REAL DEFAULT 0,
55     Value3 REAL DEFAULT 0,
56     INOUT outputers REAL DEFAULT 0
57 )
58 LANGUAGE plpgsql
59 AS $$
60 BEGIN
61     SELECT (Value1 + Value2 + Value3) / 3 INTO outputers;
62 END; $$;
63
64
65
```



с. Хранимая процедура для поиска по диапазону цен.

```
CREATE OR REPLACE FUNCTION GetProductsByPriceRange(min_price NUMERIC, max_price
NUMERIC)
RETURNS TABLE(product_name VARCHAR, price MONEY) AS $$
BEGIN
    RETURN QUERY
    SELECT
        products.product_name,
        products.price::MONEY -- Cast the price column to MONEY type
    FROM
        products
    WHERE
        products.price BETWEEN min_price AND max_price;
END;
$$ LANGUAGE plpgsql;
```

```

7 CREATE OR REPLACE FUNCTION GetProductsByPriceRange(min_price NUMERIC, max_price NUMERIC)
8 RETURNS TABLE(product_name VARCHAR, price MONEY) AS $$
9 BEGIN
10     RETURN QUERY
11     SELECT
12         products.product_name,
13         products.price::MONEY -- Cast the price column to MONEY type
14     FROM
15         products
16     WHERE
17         products.price BETWEEN min_price AND max_price;
18 END;
19 $$ LANGUAGE plpgsql;

```

```

100
101
102 SELECT * FROM GetProductsByPriceRange(15.99, 24.99);
103
104

```

Data Output Messages Notifications

	product_name character varying	price money
1	Product 4	\$15.99
2	Product 2	\$24.99

d. Хранимая процедура для поиска заказов по дате заказа и диапазону дат заказа. ,,

```

CREATE OR REPLACE FUNCTION GetOrdersByDateRange(start_date DATE, end_date
DATE)
RETURNS TABLE(order_id INT, customer_name CHAR, order_date DATE)
AS $$
BEGIN
    RETURN QUERY
    SELECT
        o.order_id,
        c.customer_name::CHAR, -- Explicitly cast to CHAR type
        o.order_date
    FROM
        orders o
    JOIN
        customers c ON o.customer_id = c.customer_id
    WHERE
        o.order_date BETWEEN start_date AND end_date;
END;
$$ LANGUAGE plpgsql;

```

```

119
120 -- CREATE OR REPLACE FUNCTION GetOrdersByDateRange(start_date DATE, end_date DATE)
121 -- RETURNS TABLE(order_id INT, customer_name CHAR, order_date DATE)
122 -- AS $$
123 -- BEGIN
124 --     RETURN QUERY
125 --     SELECT
126 --         o.order_id,
127 --         c.customer_name::CHAR, -- Explicitly cast to CHAR type
128 --         o.order_date
129 --     FROM
130 --         orders o
131 --     JOIN
132 --         customers c ON o.customer_id = c.customer_id
133 --     WHERE
134 --         o.order_date BETWEEN start_date AND end_date;
135 -- END;
136 -- $$ LANGUAGE plpgsql;
137
138
139 SELECT * FROM GetOrdersByDateRange('2024-02-01', '2024-02-22');
140
141
142

```

Data Output Messages Notifications



	order_id integer	customer_name character	order_date date
1	4	J	2024-02-22
2	1	J	2024-02-19
3	3	J	2024-02-21
4	2	J	2024-02-20

е. По заданию: варианта

1) получить сгруппированный по городу список с информацией (№заказа, дата заказа, дата доставки) за интервал временной. Отсортировать список по дате доставки. Интервал вводятся как параметры.

```
DROP FUNCTION IF EXISTS GetOrdersByCityAndDateRange(date, date);
```

```
CREATE OR REPLACE FUNCTION GetOrdersByCityAndDateRange(start_date  
DATE, end_date DATE)
```

```
RETURNS TABLE(city TEXT, order_id INT, order_date DATE, ship_date  
DATE)
```

```
AS $$
```

```
BEGIN
```

```
    RETURN QUERY
```

```
    SELECT
```

```
        c.address AS city,
```

```
        o.order_id,
```

```
        o.order_date,
```

```
        o.ship_date
```

```
    FROM
```

```
        orders o
```

```
    JOIN
```

```
        customers c ON o.customer_id = c.customer_id
```

```
    WHERE
```

```
        o.ship_date BETWEEN start_date AND end_date
```

```
    ORDER BY
```

```
        o.ship_date;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
--- - SELECT * FROM GetOrdersByCityAndDateRange('2024-02-01', '2024-02-  
28');
```

```

144 CREATE OR REPLACE FUNCTION GetOrdersByCityAndDateRange(start_date DATE, end_date DATE)
145 RETURNS TABLE(city TEXT, order_id INT, order_date DATE, ship_date DATE)
146 AS $$
147 BEGIN
148     RETURN QUERY
149     SELECT
150         c.address AS city,
151         o.order_id,
152         o.order_date,
153         o.ship_date
154     FROM
155         orders o
156     JOIN
157         customers c ON o.customer_id = c.customer_id
158     WHERE
159         o.ship_date BETWEEN start_date AND end_date
160     ORDER BY
161         o.ship_date;
162 END;
163 $$ LANGUAGE plpgsql;
164
165
166 SELECT * FROM GetOrdersByCityAndDateRange('2024-02-01', '2024-02-28');
167

```

Data Output Messages Notifications



	city text	order_id integer	order_date date	ship_date date
1	123 Main St	4	2024-02-22	2024-02-23
2	123 Main St	5	2024-02-23	2024-02-24
3	123 Main St	6	2024-02-24	2024-02-25
4	456 Elm St	7	2024-02-25	2024-02-26
5	123 Main St	8	2024-02-26	2024-02-27
6	123 Main St	9	2024-02-27	2024-02-28

2) подсчитать количество заказов по городам

```
SELECT
    c.address AS city,
    COUNT(o.order_id) AS order_count
FROM
    orders o
JOIN
    customers c ON o.customer_id = c.customer_id
GROUP BY
    c.address;
```

```
168
169 SELECT
170     c.address AS city,
171     COUNT(o.order_id) AS order_count
172 FROM
173     orders o
174 JOIN
175     customers c ON o.customer_id = c.customer_id
176 GROUP BY
177     c.address;
178
```

Data Output Messages Notifications



	city text	order_count bigint
1	456 Elm St	3
2	123 Main St	10

создать функция

```
CREATE FUNCTION calculate_average_price()  
RETURNS NUMERIC  
LANGUAGE SQL  
AS $$  
    SELECT AVG(price) FROM products;  
$$;
```

вызывать функция

```
SELECT calculate_average_price();
```

```
38  
39 -- CREATE FUNCTION calculate_average_price()  
40 -- RETURNS NUMERIC  
41 -- LANGUAGE SQL  
42 -- AS $$  
43 --     SELECT AVG(price) FROM products;  
44 -- $$;  
45  
46  
47 SELECT calculate_average_price();  
48
```

Data Output		Messages	Notifications
	calculate_average_price numeric		
1	30.7757142857142857		

This function calculates the average price of all products in your products table. You can call this function to get the average price.

3. Создать триггер INSERT

```
CREATE OR REPLACE FUNCTION update_product_stock()
RETURNS TRIGGER AS $$
BEGIN
    -- Decrease the stock quantity of the ordered product
    UPDATE products
    SET stock_quantity = stock_quantity - NEW.quantity
    WHERE product_id = NEW.product_id;

    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER update_stock_on_order
AFTER INSERT ON items
FOR EACH ROW
EXECUTE FUNCTION update_product_stock();
```

```
55
56 CREATE OR REPLACE FUNCTION update_product_stock()
57 RETURNS TRIGGER AS $$
58 BEGIN
59     -- Decrease the stock quantity of the ordered product
60     UPDATE products
61     SET stock_quantity = stock_quantity - NEW.quantity
62     WHERE product_id = NEW.product_id;
63
64     RETURN NULL;
65 END;
66 $$ LANGUAGE plpgsql;
67
68 CREATE TRIGGER update_stock_on_order
69 AFTER INSERT ON items
70 FOR EACH ROW
71 EXECUTE FUNCTION update_product_stock();
72
```

4. Создать триггер DELETE

```
CREATE OR REPLACE FUNCTION deletefn() RETURNS TRIGGER AS $$
DECLARE
    x varchar := 'John Doe';
BEGIN
    RAISE NOTICE '%', x;
    RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE OR REPLACE TRIGGER delete_tr
AFTER DELETE ON customers
EXECUTE PROCEDURE deletefn();
```

```
83
84 CREATE OR REPLACE FUNCTION deletefn() RETURNS TRIGGER AS $$
85 DECLARE
86     x varchar := 'John Doe';
87 BEGIN
88     RAISE NOTICE '%', x;
89     RETURN NULL;
90 END;
91 $$ LANGUAGE plpgsql;
92
93 CREATE OR REPLACE TRIGGER delete_tr
94 AFTER DELETE ON customers
95 EXECUTE PROCEDURE deletefn();
96
97
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 50 msec.

Создать триггер UPDATE:

```
28
29 CREATE OR REPLACE FUNCTION products_update_trigger()
30 RETURNS TRIGGER AS $$
31 BEGIN
32     -- Вставляем обновленные данные в таблицу products_audit
33     INSERT INTO products_audit (product_id, old_product_name, new_product_name, old_price, new_price, old_description, new_description,
34     VALUES (OLD.product_id, OLD.product_name, NEW.product_name, OLD.price, NEW.price, OLD.description, NEW.description, NOW());
35
36     RETURN NEW;
37 END;
38 $$ LANGUAGE plpgsql;
39
40 CREATE TRIGGER update_products_trigger
41 AFTER UPDATE ON products
42 FOR EACH ROW
43 EXECUTE FUNCTION products_update_trigger();
44
45
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 118 msec.

```
46 UPDATE products
47 SET product_name = 'New Product Name'
48 WHERE product_id = 1;
49
```

Data Output Messages Notifications

UPDATE 1

Query returned successfully in 55 msec.

6. Создать триггер, который при удалении записи из таблицы Products сначала

удаляет все связанные с ней записи из таблицы Items, а затем удаляет саму запись из таблицы Products.

Допустим, у вас есть две таблицы: товары и заказы, и вы хотите удалить все заказы, связанные с удаленным товаром, прежде чем удалять сам товар. Вот как можно создать такой триггер:

```
63
64 CREATE OR REPLACE FUNCTION delete_orders_for_product()
65 RETURNS TRIGGER AS $$
66 BEGIN
67     -- Delete all orders related to the deleted product
68     DELETE FROM orders WHERE product_id = OLD.product_id;
69
70     RETURN OLD;
71 END;
72 $$ LANGUAGE plpgsql;
73
74 CREATE TRIGGER delete_orders_for_product_trigger
75 BEFORE DELETE ON products
76 FOR EACH ROW
77 EXECUTE FUNCTION delete_orders_for_product();
78
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 44 msec.

7. Создать триггер, с использованием временной таблицы NEW.

```
80 CREATE OR REPLACE FUNCTION before_insert_products_trigger()
81 RETURNS TRIGGER AS $$
82 BEGIN
83     -- Выводим новые значения перед вставкой в таблицу
84     RAISE NOTICE 'New product_id: %, product_name: %, price: %', NEW.product_id, NEW.product_name, NEW.price;
85
86     -- Возвращаем NEW, чтобы продолжить вставку
87     RETURN NEW;
88 END;
89 $$ LANGUAGE plpgsql;
90
91 CREATE TRIGGER before_insert_products_trigger
92 BEFORE INSERT ON products
93 FOR EACH ROW
94 EXECUTE FUNCTION before_insert_products_trigger();
95
96
```

Data Output Messages Notifications

CREATE TRIGGER

Query returned successfully in 43 msec.

```
96
97 INSERT INTO products (product_name, price, description)
98 VALUES ('New Product', 29.99, 'Description of the new product');
99 |
```

Data Output Messages Notifications

NOTICE: New product_id: 33, product_name: New Product, price: 29.99

INSERT 0 1

Query returned successfully in 213 msec.

8. Создать триггер DDL, который предотвратит удаление или изменение таблиц в базе данных.

Запрет всех команд работы с таблицами

```
102 CREATE OR REPLACE FUNCTION prevent_ddl_changes()
103 RETURNS event_trigger AS $$
104 BEGIN
105     IF (TG_OP = 'DROP TABLE' OR TG_OP = 'ALTER TABLE') THEN
106         RAISE EXCEPTION 'Changes to tables are not allowed';
107     END IF;
108 END;
109 $$ LANGUAGE plpgsql;
110
111 CREATE EVENT TRIGGER prevent_ddl_trigger
112 ON ddl_command_start
113 EXECUTE FUNCTION prevent_ddl_changes();
114
115
```

Data Output Messages Notifications

CREATE EVENT TRIGGER

Query returned successfully in 141 msec.

```
118 DROP TABLE products_audit;
119
120
```

Data Output Messages Notifications

ERROR: column "tg_op" does not exist

LINE 1: (TG_OP = 'DROP TABLE' OR TG_OP = 'ALTER TABLE')
 ^

QUERY: (TG_OP = 'DROP TABLE' OR TG_OP = 'ALTER TABLE')

CONTEXT: PL/pgSQL function prevent_ddl_changes() line 3 at IF

SQL state: 42703

Выводы:

Изучены хранимые процедуры и триггеры в базах данных, приобретены практические навыки создания хранимых процедур и триггеров в среде PostgreSQL.