

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО»**

Факультет инфокоммуникационных технологий

Дисциплина:

«Проектирование и реализация баз данных»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL»

Выполнил:

студент группы К32392

Стукалов Артем Сергеевич

(подпись)

Проверил(а):

Говорова Марина Михайловна

(отметка о выполнении)

(подпись)

Санкт-Петербург

2023 г.

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

Практическое задание:

Вариант 2

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
2. 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.
3. Создать авторский триггер по варианту индивидуального задания.

Выполнение:

Индивидуальное задание БД «Ресторан»

Процедуры\функции:

1. Вывести сведения о заказах заданного официанта на заданную дату.
2. Выполнить расчет стоимости данного заказа.
3. Повышения оклада заданного сотрудника на 30 % при повышении его категории.

Триггеры:

1. Не позволять добавить в блюда заказ, если на складе не хватит продуктов для их приготовления.

Процедуры\функции:

1) Вывести сведения о заказах заданного официанта на заданную дату.

```
CREATE OR REPLACE FUNCTION get_orders_info_for_date(_employee_id
BIGINT, _order_date DATE) RETURNS TABLE(
    order_id BIGINT,
    table_id BIGINT,
    is_paid BOOLEAN,
    amount INT
) AS $$ BEGIN RETURN QUERY
SELECT o.order_id,
    o.table_id,
    o.is_paid,
    o.amount
FROM orders o
WHERE o.employee_id = _employee_id
    AND DATE(o.open_datetime) = _order_date;
END;
$$ LANGUAGE 'plpgsql';
```

Рис. 1 – Функция №1

```
restaurant=# CREATE OR REPLACE FUNCTION get_orders_info_for_date(_employee_id BIGINT, _order_date DATE) RETURNS TABLE(  
    order_id BIGINT,  
    table_id BIGINT,  
    is_paid BOOLEAN,  
    amount INT  
    ) AS $$ BEGIN RETURN QUERY  
SELECT o.order_id,  
    o.table_id,  
    o.is_paid,  
    o.amount  
FROM orders o  
WHERE o.employee_id = _employee_id  
    AND DATE(o.open_datetime) = _order_date;  
END;  
$$ LANGUAGE 'plpgsql';  
CREATE FUNCTION  
restaurant=# SELECT get_orders_info_for_date(5, '2024-03-13');  
get_orders_info_for_date  
-----  
(4,2,t,383)  
(6,1,t,610)  
(7,2,t,406)  
(9,5,t,631)  
(10,1,t,577)  
(11,4,t,1213)  
(12,1,t,617)  
(13,4,t,759)  
(16,3,t,430)  
(18,2,t,554)  
(21,1,t,427)  
(23,4,t,512)  
(12 rows)  
restaurant=#
```

Рис. 2 – Результат выполнения функции №1

2) Вывести сведения о заказах заданного официанта на заданную дату.

```
CREATE OR REPLACE FUNCTION get_order_cost(_order_id BIGINT) RETURNS INT  
AS $$  
DECLARE _res INT;  
BEGIN _res = (  
    SELECT amount  
    FROM orders  
    WHERE order_id = _order_id  
);  
IF _res IS NULL THEN RETURN 0;  
END IF;  
RETURN _res;  
END;
```

Рис. 3 – Функция №2

```

restaurant=# SELECT * FROM orders LIMIT 10;

```

order_id	table_id	employee_id	open_datetime	close_datetime	is_paid	amount
1	2	6	2024-03-13 09:00:00	2024-03-13 09:22:00	t	923
2	3	6	2024-03-13 09:32:00	2024-03-13 09:47:00	t	881
3	5	6	2024-03-13 10:12:00	2024-03-13 10:38:00	t	764
4	2	5	2024-03-13 10:54:00	2024-03-13 11:19:00	t	383
5	3	6	2024-03-13 11:37:00	2024-03-13 11:55:00	t	1230
6	1	5	2024-03-13 12:18:00	2024-03-13 12:40:00	t	610
7	2	5	2024-03-13 13:00:00	2024-03-13 13:19:00	t	406
8	4	7	2024-03-13 13:35:00	2024-03-13 14:02:00	t	481
9	5	5	2024-03-13 14:18:00	2024-03-13 14:46:00	t	631
10	1	5	2024-03-13 14:56:00	2024-03-13 15:12:00	t	577

```

(10 rows)

restaurant=# SELECT get_orde
restaurant=# SELECT get_orde
restaurant=# SELECT get_order_cost(1);
get_order_cost
-----
      923
(1 row)

restaurant=# SELECT get_order_cost(10);
get_order_cost
-----
      577
(1 row)

restaurant=# SELECT get_order_cost(1000000000);
get_order_cost
-----
         0
(1 row)

restaurant=# 

```

Рис. 4 – Результат выполнения функции №2

3) Повышения оклада заданного сотрудника на 30 % при повышении его категории. Выполнение данного задания не представляется возможным, так как первоначально категория сотрудника была задана в текстовом виде и не была ограничена каким-то определенным набором возможных значений. Поэтому задание было немного видоизменено. Теперь нужно выполнить повышение оклада сотрудника на 30% при повышении сотрудника с должности “заготовщик” до “Повар”.

```

CREATE OR REPLACE PROCEDURE up_employee_kitchen_rank(_employee_id
BIGINT) AS $$
DECLARE _emp_position VARCHAR(80);
        _emp_category VARCHAR(80);
        _emp_salary INT;
BEGIN CREATE TEMP TABLE _employee AS (
        SELECT position,
                category,
                salary
        FROM employees
        WHERE employee_id = _employee_id
);
        _emp_position = (
        SELECT position
        FROM _employee
);
        _emp_category = (

```

```

SELECT category
FROM _employee
);
emp_salary = (
SELECT salary
FROM _employee
);
IF _emp_position != 'заготовщик'
OR _emp_category != 'Кухня' THEN DROP TABLE _employee;
RETURN;
END IF;
UPDATE employees
SET salary = salary * 1.3,
    position = 'Повар'
WHERE employee_id = _employee_id;
DROP TABLE IF EXISTS _employee;
END;
$$ LANGUAGE 'plpgsql';

```

Рис. 5 – Процедура №1

```

restaurant=# SELECT * FROM employees;

```

employee_id	passport	fio	position	category	salary
1	1234 123456	Стукалов Артем Сергеевич	Шеф-повар	Кухня	100000
2	1234 123457	Денис Аксенов Иванович	Су-шеф	Кухня	70000
3	1234 123458	Егор Лавров Ефимович	Повар	Кухня	50000
4	1234 123459	Кристина Гончарова Владимировна	Управляющий	Администрация	100000
5	1234 123460	Влад Анисимов Юрьевич	Официант	Обслуживание	50000
6	2234 123460	Иван Инванович Иванов	Официант	Обслуживание	50000
7	1234 123461	Петр Анисимов Юрьевич	Официант	Обслуживание	50000
8	1234 125456	Евгений Сергеевич	заготовщик	Кухня	39000

```

(8 rows)

restaurant=# CALL up_employee_kitchen_rank(8);
CALL
restaurant=# SELECT * FROM employees;

```

employee_id	passport	fio	position	category	salary
1	1234 123456	Стукалов Артем Сергеевич	Шеф-повар	Кухня	100000
2	1234 123457	Денис Аксенов Иванович	Су-шеф	Кухня	70000
3	1234 123458	Егор Лавров Ефимович	Повар	Кухня	50000
4	1234 123459	Кристина Гончарова Владимировна	Управляющий	Администрация	100000
5	1234 123460	Влад Анисимов Юрьевич	Официант	Обслуживание	50000
6	2234 123460	Иван Инванович Иванов	Официант	Обслуживание	50000
7	1234 123461	Петр Анисимов Юрьевич	Официант	Обслуживание	50000
8	1234 125456	Евгений Сергеевич	Повар	Кухня	50700

```

(8 rows)

restaurant=# CALL up_employee_kitchen_rank(1);
CALL
restaurant=# SELECT * FROM employees;

```

employee_id	passport	fio	position	category	salary
1	1234 123456	Стукалов Артем Сергеевич	Шеф-повар	Кухня	100000
2	1234 123457	Денис Аксенов Иванович	Су-шеф	Кухня	70000
3	1234 123458	Егор Лавров Ефимович	Повар	Кухня	50000
4	1234 123459	Кристина Гончарова Владимировна	Управляющий	Администрация	100000
5	1234 123460	Влад Анисимов Юрьевич	Официант	Обслуживание	50000
6	2234 123460	Иван Инванович Иванов	Официант	Обслуживание	50000
7	1234 123461	Петр Анисимов Юрьевич	Официант	Обслуживание	50000
8	1234 125456	Евгений Сергеевич	Повар	Кухня	50700

```

(8 rows)

restaurant=#

```

Рис. 6 – Результат выполнения процедуры №1

Модификация триггера из практической работы:

```
create or replace function fn_check_time_punch() returns trigger as
$psql$
DECLARE prev_tp RECORD;
begin
SELECT punch_time,
       is_out_punch INTO prev_tp
FROM time_punch
WHERE employee_id = new.employee_id
ORDER BY punch_time DESC
LIMIT 1;
IF prev_tp IS NULL THEN RETURN new;
END IF;
IF prev_tp.is_out_punch = new.is_out_punch
OR prev_tp.punch_time >= new.punch_time
OR new.punch_time > now() THEN RETURN NULL;
END IF;
RETURN new;
end;
$psql$ language plpgsql;
```

Рис. 7 – Триггер из практической работы

Триггеры:

1) Не позволять добавить в блюда заказ, если на складе не хватит продуктов для их приготовления.

```
CREATE OR REPLACE FUNCTION fn_check_if_can_order() RETURNS TRIGGER AS
$psql$
BEGIN IF (
    SELECT pd.product_id
    FROM products_in_dishes pd
        LEFT JOIN products p USING(product_id)
    WHERE pd.dish_id = NEW.dish_id
        AND pd.product_volume * NEW.dish_count > p.stock_volume
    LIMIT 1
) IS NOT NULL THEN RETURN NULL;
END IF;
RETURN NEW;
END;
$psql$ LANGUAGE 'plpgsql';
--
CREATE TRIGGER check_if_can_order BEFORE
INSERT ON cooking_progress FOR EACH ROW EXECUTE PROCEDURE
fn_check_if_can_order();
```

Рис. 8 – Триггер №1

```
restaurant=# INSERT INTO cooking_progress (
dish_id,
order_id,
employee_id,
dish_count,
is_ready,
guest_wishes
)
VALUES (1, 1, 1, 10000, FALSE, '');
INSERT 0 0
restaurant=# SELECT * FROM cooking_progress WHERE employee_id = 1 AND order_id = 1;
 cooking_progress_id | dish_id | order_id | employee_id | dish_count | is_ready | guest_wishes
-----+-----+-----+-----+-----+-----+-----
1 | 48 | 1 | 1 | 2 | t |
(1 row)

restaurant=# INSERT INTO cooking_progress (
dish_id,
order_id,
employee_id,
dish_count,
is_ready,
guest_wishes
)
VALUES (1, 1, 1, 1, FALSE, '');
INSERT 0 1
restaurant=# SELECT * FROM cooking_progress WHERE employee_id = 1 AND order_id = 1;
 cooking_progress_id | dish_id | order_id | employee_id | dish_count | is_ready | guest_wishes
-----+-----+-----+-----+-----+-----+-----
1 | 48 | 1 | 1 | 2 | t |
690584 | 1 | 1 | 1 | 1 | f |
(2 rows)

restaurant=#
```

Рис. 19 – Результат выполнения триггера №1

Выводы:

В процессе выполнения данной лабораторной работы удалось овладеть навыками написания и использования процедур, функций и триггеров в PostgreSQL. Также для себя я рассмотрел использование возможностей функций и процедур как ЯП. И на данный момент для себя могу отметить, что несмотря на схожесть с различными ЯП, SQL не обладает достаточной гибкостью.