

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: Бархатова Н.А.

Факультет: ИКТ

Группа: К3239

Преподаватель: Говорова М.М.



Санкт-Петербург 2023

Оглавление

Цель работы	3
Практическое задание	3
Выполнение.....	Ошибка! Закладка не определена.
Создайте хранимые процедуры/функции.....	3
1. Вывести сведения о заказах заданного официанта на заданную дату.	3
2. Выполнить расчет стоимости заданного заказа.	4
3. Повышения оклада заданного сотрудника на 30 % при повышении его категории. ..	5
Триггеры	7
Вывод.....	9

Цель работы

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

Практическое задание

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

Создайте хранимые процедуры/функции

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

```
CREATE OR REPLACE FUNCTION get_orders_by_waiter_and_date(  
    p_waiter_id INTEGER,  
    p_date DATE  
)  
RETURNS TABLE (  
    order_id INTEGER,  
    order_date TIMESTAMP,  
    menu_item_id INTEGER,  
    menu_item_name VARCHAR,  
    item_count INTEGER  
)  
AS $$  
BEGIN  
    RETURN QUERY  
    SELECT  
        o.id AS order_id,  
        o.date_time AS order_date,  
        oi.menu_item_id,  
        mi.name AS menu_item_name,  
        oi.count AS item_count  
    FROM  
        schema.order o  
    JOIN  
        schema.order_composition oi ON o.id = oi.order_id  
    JOIN  
        schema.menu_item mi ON oi.menu_item_id = mi.id  
    WHERE  
        o.employee_id = p_waiter_id  
        AND DATE_TRUNC('day', o.date_time) = DATE_TRUNC('day', p_date);  
END;  
$$ LANGUAGE plpgsql;
```

```

restaurant_database=# CREATE OR REPLACE FUNCTION get_orders_by_waiter_and_date(
restaurant_database(#   p_waiter_id INTEGER,
restaurant_database(#   p_date DATE
restaurant_database(# )
restaurant_database=# RETURNS TABLE (
restaurant_database(#   order_id INTEGER,
restaurant_database(#   order_date TIMESTAMP,
restaurant_database(#   menu_item_id INTEGER,
restaurant_database(#   menu_item_name VARCHAR(128),
restaurant_database(#   item_count INTEGER
restaurant_database(# )
restaurant_database=# AS $$
restaurant_database=# BEGIN
restaurant_database=# RETURN QUERY
restaurant_database=# SELECT
restaurant_database=#     o.id AS order_id,
restaurant_database=#     o.date_time AS order_date,
restaurant_database=#     oi.menu_item_id,
restaurant_database=#     mi.name AS menu_item_name,
restaurant_database=#     oi.count AS item_count
restaurant_database=# FROM
restaurant_database=#     schema.order o
restaurant_database=# JOIN
restaurant_database=#     schema.order_composition oi ON o.id = oi.order_id
restaurant_database=# JOIN
restaurant_database=#     schema.menu_item mi ON oi.menu_item_id = mi.id
restaurant_database=# WHERE
restaurant_database=#     o.employee_id = p_waiter_id
restaurant_database=#     AND DATE_TRUNC('day', o.date_time) = DATE_TRUNC('day', p_date);
restaurant_database=# END;
restaurant_database=# $$ LANGUAGE plpgsql;
CREATE FUNCTION

```

```

restaurant_database=# SELECT * FROM get_orders_by_waiter_and_date(14, '2023-11-01');

```

order_id	order_date	menu_item_id	menu_item_name	item_count
98	2023-11-01 12:30:36	13	Паста с креветками в сливочном соусе	2
98	2023-11-01 12:30:36	13	Паста с креветками в сливочном соусе	2
98	2023-11-01 12:30:36	1	Картофельное пюре с говяжьим стейком	2
99	2023-11-01 14:30:12	8	Банановый десерт с медом и орехами	1
99	2023-11-01 14:30:12	1	Картофельное пюре с говяжьим стейком	3
100	2023-11-01 16:30:00	10	Овсяный завтрак с фруктами и сметаной	2
100	2023-11-01 16:30:00	7	Рис с овощами и курицей по-тайски	2
100	2023-11-01 16:30:00	1	Картофельное пюре с говяжьим стейком	3
101	2023-11-01 17:30:50	2	Жареный картофель с капустой и свиной	3
101	2023-11-01 17:30:50	7	Рис с овощами и курицей по-тайски	2
102	2023-11-01 18:09:34	12	Куриные котлеты с овощами на гриле	2
102	2023-11-01 18:09:34	11	Сельдь под шубой	3
102	2023-11-01 18:09:34	2	Жареный картофель с капустой и свиной	2
102	2023-11-01 18:09:34	10	Овсяный завтрак с фруктами и сметаной	3
103	2023-11-01 21:30:00	13	Паста с креветками в сливочном соусе	3
103	2023-11-01 21:30:00	2	Жареный картофель с капустой и свиной	1
103	2023-11-01 21:30:00	12	Куриные котлеты с овощами на гриле	1
104	2023-11-01 14:50:03	12	Куриные котлеты с овощами на гриле	2
104	2023-11-01 14:50:03	9	Пшеничная каша с яблоками и корицей	3
104	2023-11-01 14:50:03	10	Овсяный завтрак с фруктами и сметаной	3
104	2023-11-01 14:50:03	6	Гречка с сыром и орехами	1
105	2023-11-01 18:10:38	13	Паста с креветками в сливочном соусе	1
105	2023-11-01 18:10:38	10	Овсяный завтрак с фруктами и сметаной	1
105	2023-11-01 18:10:38	5	Лосось с овощами на гриле	1
106	2023-11-01 21:30:49	14	Салат из свежих овощей с оливковым маслом	1
106	2023-11-01 21:30:49	3	Морковный суп с курицей	1

(26 строк)

2. Выполнить расчет стоимости заданного заказа.

```

CREATE OR REPLACE FUNCTION calculate_order_cost(p_order_id INTEGER)
RETURNS INTEGER
AS $$
DECLARE
    total_cost INTEGER:= 0;
BEGIN
    SELECT SUM(mip.price * oc.count)
    INTO total_cost

```

```

FROM
  schema.order_composition oc
JOIN
  schema.menu_item_price mip ON oc.menu_item_id = mip.menu_item_id
WHERE
  oc.order_id = p_order_id;

RETURN total_cost;
END;
$$ LANGUAGE plpgsql;

```

```

restaurant_database=# CREATE OR REPLACE FUNCTION calculate_order_cost(p_order_id INTEGER)
restaurant_database=# RETURNS INTEGER
restaurant_database=# AS $$
restaurant_database$$ DECLARE
restaurant_database$$   total_cost INTEGER:= 0;
restaurant_database$$ BEGIN
restaurant_database$$   SELECT SUM(mip.price * oc.count)
restaurant_database$$   INTO total_cost
restaurant_database$$   FROM
restaurant_database$$     schema.order_composition oc
restaurant_database$$   JOIN
restaurant_database$$     schema.menu_item_price mip ON oc.menu_item_id = mip.menu_item_id
restaurant_database$$   WHERE
restaurant_database$$     oc.order_id = p_order_id;
restaurant_database$$   RETURN total_cost;
restaurant_database$$ END;
restaurant_database$$ $$ LANGUAGE plpgsql;
CREATE FUNCTION

```

```

restaurant_database=# SELECT * FROM calculate_order_cost(98);
calculate_order_cost
-----
                4416
(1 строка)

```

3. Повышения оклада заданного сотрудника на 30 % при повышении его категории.

```

restaurant_database=# SELECT * FROM schema.job_title;

```

id	salary	title	category
11	200	Повар	2
12	300	Шеф	6
13	250	Кондитер	1
14	150	Официант	4
15	180	Официант	2
16	200	Официант	3
17	220	Администратор	4
18	250	Су шеф	4
19	280	Повар универсал	5

(9 строк)

```

CREATE OR REPLACE PROCEDURE increase_salary_if_category_higher(employee_id INTEGER,
new_category INTEGER)
AS $$
DECLARE
    current_salary INTEGER;
    current_category INTEGER;
BEGIN
    SELECT jt.salary, jt.category INTO current_salary, current_category
    FROM schema.job_title jt
    JOIN schema.employee e ON e.job_title_id = jt.id
    WHERE e.personnel_number = employee_id;

    IF new_category > current_category AND new_category <= 6 THEN
        UPDATE schema.job_title jt
        SET salary = current_salary * POWER(1.3, new_category - current_category), category = new_category
        WHERE jt.id = (
            SELECT job_title_id
            FROM schema.employee
            WHERE personnel_number = employee_id
        );
    END IF;
END;

```

```

$$ LANGUAGE plpgsql;

```

```

restaurant_database=# CREATE OR REPLACE PROCEDURE increase_salary_if_category_higher(employee_id INTEGER, new_category I
INTEGER)
restaurant_database=# AS $$
restaurant_database=# DECLARE
restaurant_database=#     current_salary INTEGER;
restaurant_database=#     current_category INTEGER;
restaurant_database=# BEGIN
restaurant_database=#     SELECT jt.salary, jt.category INTO current_salary, current_category
restaurant_database=#     FROM schema.job_title jt
restaurant_database=#     JOIN schema.employee e ON e.job_title_id = jt.id
restaurant_database=#     WHERE e.personnel_number = employee_id;
restaurant_database=#
restaurant_database=#     IF new_category > current_category AND new_category <= 6 THEN
restaurant_database=#         UPDATE schema.job_title jt
restaurant_database=#         SET salary = current_salary * POWER(1.3, new_category - current_category), category = new_
category
restaurant_database=#         WHERE jt.id = (
restaurant_database=#             SELECT job_title_id
restaurant_database=#             FROM schema.employee
restaurant_database=#             WHERE personnel_number = employee_id
restaurant_database=#         );
restaurant_database=#     END IF;
restaurant_database=# END;
restaurant_database=# $$ LANGUAGE plpgsql;
CREATE PROCEDURE

```

```

restaurant_database=# CALL increase_salary_if_category_higher(14, 5);
CALL

```

```

restaurant_database=# SELECT * FROM schema.job_title;

```

id	salary	title	category
11	200	Повар	2
12	300	Шеф	6
13	250	Кондитер	1
15	180	Официант	2
16	200	Официант	3
17	220	Администратор	4
18	250	Су шеф	4
19	280	Повар универсал	5
14	195	Официант	5

```

(9 строк)

```

Триггеры

Триггер на создание записи в таблицу menu_item_price для нового блюда с автоматической генераций цены блюда через стоимость ингредиентов.

```
CREATE OR REPLACE FUNCTION calculate_menu_item_price_update()  
RETURNS TRIGGER AS $$  
DECLARE  
    total_ingredient_cost INTEGER;  
BEGIN  
    SELECT COALESCE (SUM (pc.ingredient_price * mic.ingredient_value), 0)  
    INTO total_ingredient_cost  
    FROM schema.menu_item_composition mic  
    JOIN schema.purchase_composition pc ON mic.ingredient_id =  
pc.ingredient_id  
    WHERE mic.menu_item_id = NEW.menu_item_id;  
    UPDATE schema.menu_item_price SET price = total_ingredient_cost * 8  
    WHERE menu_item_id = NEW.menu_item_id;  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER calculate_menu_item_price_trigger_update  
BEFORE UPDATE ON schema.menu_item_composition  
FOR EACH ROW  
EXECUTE FUNCTION calculate_menu_item_price_update();
```

```
CREATE OR REPLACE FUNCTION calculate_menu_item_price_insert()  
RETURNS TRIGGER AS $$  
DECLARE  
    total_ingredient_cost INTEGER;  
BEGIN  
    SELECT COALESCE (SUM(pc.ingredient_price * NEW.ingredient_value), 0)  
    INTO total_ingredient_cost  
    FROM schema.menu_item_composition mic  
    JOIN schema.purchase_composition pc ON mic.ingredient_id =  
pc.ingredient_id  
    WHERE mic.menu_item_id = NEW.menu_item_id;  
  
    INSERT INTO schema.menu_item_price (menu_item_id, price)  
    VALUES (NEW.menu_item_id, total_ingredient_cost * 8)  
    ON CONFLICT (menu_item_id) DO NOTHING;  
  
    RETURN NEW;  
END;  
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER calculate_menu_item_price_trigger_insert  
AFTER INSERT ON schema.menu_item_composition  
FOR EACH ROW  
EXECUTE FUNCTION calculate_menu_item_price_insert();
```

```
restaurant_database=# select * from schema.menu_item_composition;
 menu_item_id | ingredient_id | ingredient_value | measure | id
-----+-----+-----+-----+-----
          1 |          17 |          0.003 | kg      |  4
          1 |          18 |          0.002 | kg      |  5
          2 |           6 |           0.6 | kg      |  6
          2 |           2 |           0.3 | kg      |  7
          2 |           4 |           0.1 | kg      |  8
          2 |          16 |          0.005 | kg      |  9
          2 |          17 |          0.003 | kg      | 10
          3 |           3 |           0.3 | kg      | 11
          3 |           7 |           0.4 | kg      | 12
          3 |          16 |          0.003 | kg      | 13
          3 |          19 |          0.005 | kg      | 14
          4 |           8 |           0.2 | kg      | 15
          4 |          15 |           0.03 | kg      | 16
          4 |          17 |          0.002 | kg      | 17
          5 |           9 |           0.7 | kg      | 18
          5 |           2 |           0.3 | kg      | 19
```

```
restaurant_database=# select * from schema.menu_item_price;
 price | start_date | end_date | menu_item_id | id
-----+-----+-----+-----+-----
    899 | 2023-01-02 | 2024-01-01 |           2 |  2
    389 | 2023-01-02 | 2024-01-01 |           3 |  3
    729 | 2023-01-02 | 2024-01-01 |           4 |  4
   1089 | 2023-01-02 | 2024-01-01 |           5 |  5
    349 | 2023-01-02 | 2024-01-01 |           6 |  6
    399 | 2023-01-02 | 2024-01-01 |           7 |  7
    229 | 2023-01-02 | 2024-01-01 |           8 |  8
    179 | 2023-01-02 | 2024-01-01 |           9 |  9
    450 | 2023-01-02 | 2024-01-01 |          10 | 10
    359 | 2023-01-02 | 2024-01-01 |          11 | 11
    349 | 2023-01-02 | 2024-01-01 |          12 | 12
    879 | 2023-01-02 | 2024-01-01 |          13 | 13
    249 | 2023-01-02 | 2024-01-01 |          14 | 14
```



```
restaurant_database=# update schema.menu_item_composition set ingredient_value = 1000 where id = 6;
UPDATE 1
restaurant_database=# select * from schema.menu_item_price;
```

price	start_date	end_date	menu_item_id	id
389	2023-01-02	2024-01-01	3	3
729	2023-01-02	2024-01-01	4	4
1089	2023-01-02	2024-01-01	5	5
349	2023-01-02	2024-01-01	6	6
399	2023-01-02	2024-01-01	7	7
229	2023-01-02	2024-01-01	8	8
179	2023-01-02	2024-01-01	9	9
450	2023-01-02	2024-01-01	10	10
359	2023-01-02	2024-01-01	11	11
349	2023-01-02	2024-01-01	12	12
879	2023-01-02	2024-01-01	13	13
349	2023-01-02	2024-01-01	14	14
249	2023-01-02	2024-01-01	15	15
336	2023-01-02	2023-12-10	16	24
404800	2023-01-02	2024-01-01	1	1
4800520	2023-01-02	2024-01-01	2	2

(16 строк)

```
restaurant_database=# INSERT INTO schema.menu_item_composition (menu_item_id, ingredient_id,ingredient_value,measure) VA
LUES (16, 31, 0.05, 'kg'), (16, 10, 0.1, 'kg'), (16, 15, 0.015, 'l');
INSERT 0 3
restaurant_database=# SELECT * FROM schema.menu_item_price;
```

price	start_date	end_date	menu_item_id	id
450	2023-01-02	2024-01-01	1	1
899	2023-01-02	2024-01-01	2	2
389	2023-01-02	2024-01-01	3	3
729	2023-01-02	2024-01-01	4	4
1089	2023-01-02	2024-01-01	5	5
349	2023-01-02	2024-01-01	6	6
399	2023-01-02	2024-01-01	7	7
229	2023-01-02	2024-01-01	8	8
179	2023-01-02	2024-01-01	9	9
450	2023-01-02	2024-01-01	10	10
359	2023-01-02	2024-01-01	11	11
349	2023-01-02	2024-01-01	12	12
879	2023-01-02	2024-01-01	13	13
349	2023-01-02	2024-01-01	14	14
249	2023-01-02	2024-01-01	15	15
336	2023-01-02	2023-12-10	16	24

(16 строк)

Вывод

В ходе выполнения лабораторной работы были успешно освоены практические навыки создания и использования процедур, функций и триггеров в базе данных PostgreSQL. Работа велась в компьютерном классе с использованием СУБД PostgreSQL и SQL Shell (psql). Полученные навыки предоставляют возможность эффективного управления данными в базе, автоматизации рутинных операций и улучшению общей производительности системы. Это важный шаг в обучении и практике работы с базами данных, который может быть полезен как в академическом, так и в профессиональном контексте. Таким образом, цель работы достигнута, а полученные знания и опыт позволят легко адаптироваться к решению разнообразных задач в области баз данных.