

Национальный исследовательский университет ИТМО



Лабораторная работа №3

«ЗАПРОСЫ НА ВЫБОРКУ И МОДИФИКАЦИЮ ДАННЫХ,
ПРЕДСТАВЛЕНИЯ И ИНДЕКСЫ В POSTGRESQL»

По дисциплине
«Проектирование и реализация баз данных»

Выполнил:
Кривцов П.А.
Группа:
К3240
Преподаватель:
Говорова М.М.

Санкт-Петербург
2022 г

ЦЕЛЬ РАБОТЫ

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

ПРАКТИЧЕСКОЕ ЗАДАНИЕ (ВАРИАНТ 9 – Оптовая база)

Задание 1. Создайте хранимую процедуру:

- для снижения цены на заданный процент для товаров, у которых срок пребывания на складе превысил заданный норматив.
- для расчета стоимости всех партий товаров, проданных за прошедшие сутки.

Задание 2. Создать необходимые триггеры.

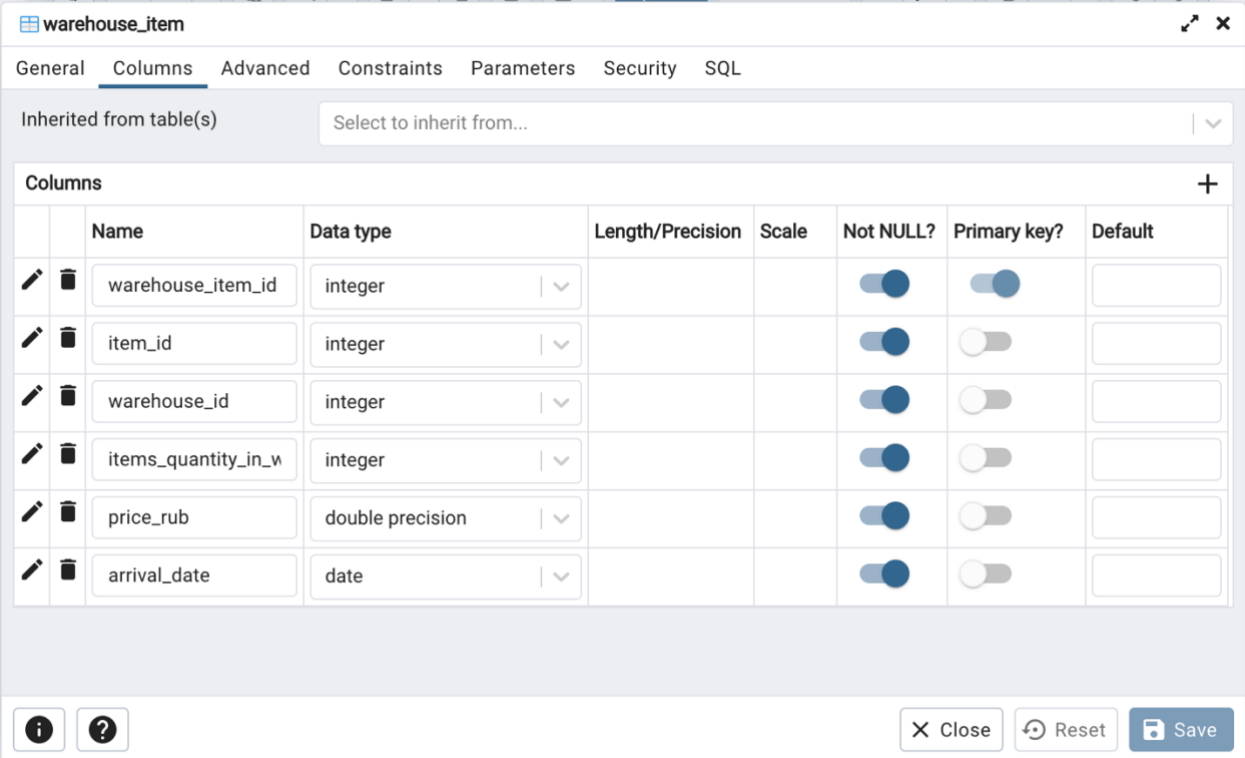
Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL. Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

ВЫПОЛНЕНИЕ

Создание хранимой процедуры.

1. Для снижения цены на заданный процент для товаров, у которых срок пребывания на складе превысил заданный норматив.

Для выполнения этого задания я добавил атрибут “arrival_date” в сущность warehouse_item, чтобы отслеживать дату прибытия товара на склад (рисунок 1).



The screenshot shows the 'warehouse_item' table configuration interface. The 'Columns' tab is selected, displaying a table with the following columns:

	Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
	warehouse_item_id	integer			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	item_id	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	warehouse_id	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	items_quantity_in_w	integer			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	price_rub	double precision			<input checked="" type="checkbox"/>	<input type="checkbox"/>	
	arrival_date	date			<input checked="" type="checkbox"/>	<input type="checkbox"/>	

At the bottom of the interface, there are buttons for 'Close', 'Reset', and 'Save'.

Рисунок 1 - добавление атрибута

```

CREATE OR REPLACE PROCEDURE
wholesale_warehouse.reduce_obsolete_prices(percentage_reduction double precision,
aging_interval interval)
LANGUAGE SQL
AS $$
    UPDATE wholesale_warehouse.warehouse_item
    SET price_rub = price_rub * (1 - percentage_reduction / 100)
    WHERE (TIMESTAMP 'today' - arrival_date) > aging_interval
$$;

```

```

(wholesale_warehouse=# CREATE OR REPLACE PROCEDURE
(wholesale_warehouse=# wholesale_warehouse.reduce_obsolete_prices(percentage_reduction double precision, aging_interval interval)
(wholesale_warehouse=# LANGUAGE SQL
(wholesale_warehouse=# AS $$
(wholesale_warehouse=# UPDATE wholesale_warehouse.warehouse_item
(wholesale_warehouse=# SET price_rub = price_rub * (1 - percentage_reduction / 100)
(wholesale_warehouse=# WHERE (TIMESTAMP 'today' - arrival_date) > aging_interval
(wholesale_warehouse=# $$;
CREATE PROCEDURE
(wholesale_warehouse=#
(wholesale_warehouse=#
(wholesale_warehouse=# SELECT warehouse_item_id, price_rub FROM wholesale_warehouse.warehouse_item;
warehouse_item_id | price_rub
-----+-----
1 | 30
2 | 60
(2 rows)

(wholesale_warehouse=#
(wholesale_warehouse=#
(wholesale_warehouse=# CALL wholesale_warehouse.reduce_obsolete_prices(15, interval '10 days');
CALL
(wholesale_warehouse=#
(wholesale_warehouse=#
(wholesale_warehouse=# SELECT warehouse_item_id, price_rub FROM wholesale_warehouse.warehouse_item;
warehouse_item_id | price_rub
-----+-----
1 | 25.5
2 | 51
(2 rows)

```

Рисунок 2 - создание и вызов процедуры 1

2. Для расчета стоимости всех партий товаров, проданных за прошедшие сутки.

```

CREATE OR REPLACE FUNCTION
wholesale_warehouse.yesterday_realizations_profit()
RETURNS TABLE(id int, income double precision)
LANGUAGE SQL
AS $$
    SELECT realization.realization_id, SUM(salable_item.salable_item_price_rub *
salable_item.salable_items_quantity)
FROM wholesale_warehouse.realization

```

```

JOIN wholesale_warehouse.salable_item
ON realization.realization_id = salable_item.realization_id
WHERE realization.order_date = TIMESTAMP 'yesterday'
      AND realization.realization_payment_state IN ('предоплата', 'оплачено')
GROUP BY realization.realization_id
$$;

```

```

(wholesale_warehouse=# CREATE OR REPLACE FUNCTION wholesale_warehouse.yesterday_realizations_profit()
(wholesale_warehouse=# RETURNS TABLE(id int, income double precision)
(wholesale_warehouse=# LANGUAGE SQL
(wholesale_warehouse=# AS $$
(wholesale_warehouse$# SELECT realization.realization_id, SUM(salable_item.salable_item_price_rub * salable_item.salable_items_quantity)
(wholesale_warehouse$# FROM wholesale_warehouse.realization
(wholesale_warehouse$# JOIN wholesale_warehouse.salable_item
(wholesale_warehouse$# ON realization.realization_id = salable_item.realization_id
(wholesale_warehouse$# WHERE realization.order_date = TIMESTAMP 'yesterday'
(wholesale_warehouse$# AND realization.realization_payment_state IN ('предоплата', 'оплачено')
(wholesale_warehouse$# GROUP BY realization.realization_id
(wholesale_warehouse$# $$;
CREATE FUNCTION
(wholesale_warehouse=#
(wholesale_warehouse=#
(wholesale_warehouse=# SELECT * FROM wholesale_warehouse.yesterday_realizations_profit();
 id | income
-----+-----
  2 | 5425000
(1 row)

```

Рисунок 3 - создание и вызов функции 2

3. Процедура, обеспечивающая вставку товара

```

CREATE OR REPLACE PROCEDURE wholesale_warehouse.add_item(
    i_name character(30),
    i_description text,
    i_measure_unit character(6)
)
LANGUAGE SQL
AS $$
    INSERT INTO wholesale_warehouse.item (item_name, item_description,
item_measure_unit)
    VALUES (i_name, i_description, i_measure_unit)
$$;

```

```

(wholesale_warehouse=#
(wholesale_warehouse=# CREATE OR REPLACE PROCEDURE wholesale_warehouse.add_item(
(wholesale_warehouse=#   i_name character(30),
(wholesale_warehouse=#   i_description text,
(wholesale_warehouse=#   i_measure_unit character(6)
(wholesale_warehouse=# )
(wholesale_warehouse=# LANGUAGE SQL
(wholesale_warehouse=# AS $$
(wholesale_warehouse$#   INSERT INTO wholesale_warehouse.item (item_name, item_description, item_measure_unit)
(wholesale_warehouse$#   VALUES (i_name, i_description, i_measure_unit)
(wholesale_warehouse$# $$;
CREATE PROCEDURE
(wholesale_warehouse=#
(wholesale_warehouse=# SELECT * FROM wholesale_warehouse.item
(wholesale_warehouse=# ;
  item_id |      item_name      |      item_description      | item_measure_unit
-----|-----|-----|-----
      1 | Black gel pen      | A pen with recyclable dyes invented at ITMO University | шт
      2 | Blue gel pen      | A pen with recyclable dyes invented at ITMO University | шт
(2 rows)

(wholesale_warehouse=#
(wholesale_warehouse=# CALL wholesale_warehouse.add_item('Green gel pen', 'Vibrant green gel ink pen', 'шт')
(wholesale_warehouse=# ;
CALL
(wholesale_warehouse=#
(wholesale_warehouse=# SELECT * FROM wholesale_warehouse.item
(wholesale_warehouse=# ;
  item_id |      item_name      |      item_description      | item_measure_unit
-----|-----|-----|-----
      1 | Black gel pen      | A pen with recyclable dyes invented at ITMO University | шт
      2 | Blue gel pen      | A pen with recyclable dyes invented at ITMO University | шт
      5 | Green gel pen      | Vibrant green gel ink pen | шт
(3 rows)

wholesale_warehouse=# _

```

Рисунок 4 – создание и вызов процедуры 3

Создание триггера

Логгирование INSERT, DELETE, UPDATE для таблицы realization.

- Функция триггера:

```
CREATE OR REPLACE FUNCTION wholesale_warehouse.realization_logging()
```

```
RETURNS TRIGGER
```

```
LANGUAGE plpgsql
```

```
AS $$
```

```
    DECLARE
```

```
        info_str varchar(50);
```

```
    BEGIN
```

```
        IF TG_OP = 'INSERT' THEN
```

```
            info_str := concat_ws(' ', 'Insert realization with id:',
```

```
NEW.realization_id);
```

```
            INSERT INTO wholesale_warehouse.logs(log_info, log_time)
```

```
values (info_str, NOW());
```

```

        RETURN NEW;

    ELSIF TG_OP = 'UPDATE' THEN
        info_str := concat_ws(' ', 'Update realization with id:',
NEW.realization_id);

        INSERT INTO wholesale_warehouse.logs(log_info, log_time)
values (info_str, NOW());

        RETURN NEW;

    ELSIF TG_OP = 'DELETE' THEN
        info_str := concat_ws(' ', 'Delete realization with id:',
OLD.realization_id);

        INSERT INTO wholesale_warehouse.logs(log_info, log_time)
values (info_str, NOW());

        RETURN OLD;

    END IF;

END;

$$;

```

```

\wholesale_warehouse=#
\wholesale_warehouse=#
\wholesale_warehouse=# CREATE OR REPLACE FUNCTION wholesale_warehouse.realization_logging()
\wholesale_warehouse=# RETURNS TRIGGER
\wholesale_warehouse=# LANGUAGE plpgsql
\wholesale_warehouse=# AS $$
\wholesale_warehouse$# DECLARE
\wholesale_warehouse$#     info_str varchar(50);
\wholesale_warehouse$# BEGIN
\wholesale_warehouse$#     IF TG_OP = 'INSERT' THEN
\wholesale_warehouse$#         info_str := concat_ws(' ', 'Insert realization with id:', NEW.realization_id);
\wholesale_warehouse$#
\wholesale_warehouse$#         INSERT INTO wholesale_warehouse.logs(log_info, log_time) values (info_str, NOW());
\wholesale_warehouse$#         RETURN NEW;
\wholesale_warehouse$#
\wholesale_warehouse$#     ELSIF TG_OP = 'UPDATE' THEN
\wholesale_warehouse$#         info_str := concat_ws(' ', 'Update realization with id:', NEW.realization_id);
\wholesale_warehouse$#
\wholesale_warehouse$#         INSERT INTO wholesale_warehouse.logs(log_info, log_time) values (info_str, NOW());
\wholesale_warehouse$#         RETURN NEW;
\wholesale_warehouse$#
\wholesale_warehouse$#     ELSIF TG_OP = 'DELETE' THEN
\wholesale_warehouse$#         info_str := concat_ws(' ', 'Delete realization with id:', OLD.realization_id);
\wholesale_warehouse$#
\wholesale_warehouse$#         INSERT INTO wholesale_warehouse.logs(log_info, log_time) values (info_str, NOW());
\wholesale_warehouse$#         RETURN OLD;
\wholesale_warehouse$#
\wholesale_warehouse$#     END IF;
\wholesale_warehouse$# END;
\wholesale_warehouse$# $$;
\wholesale_warehouse=#
\wholesale_warehouse=#
\wholesale_warehouse=#

```

Рисунок 4 - создание функции триггера

- Триггер:

```

CREATE TRIGGER t_realization
AFTER INSERT OR UPDATE OR DELETE
ON wholesale_warehouse.realization
FOR EACH ROW EXECUTE PROCEDURE
    wholesale_warehouse.realization_logging();

```

```

\wholesale_warehouse=#
\wholesale_warehouse=#
\wholesale_warehouse=# CREATE TRIGGER t_realization
\wholesale_warehouse=# AFTER INSERT OR UPDATE OR DELETE
\wholesale_warehouse=# ON wholesale_warehouse.realization
\wholesale_warehouse=# FOR EACH ROW EXECUTE PROCEDURE wholesale_warehouse.realization_logging();
\wholesale_warehouse=# CREATE TRIGGER
\wholesale_warehouse=#

```

Рисунок 5 - создание триггера

Теперь, в ответ на изменение, добавление и удаление данных в сущности realization, происходит логгирование этих событий в сущности logs (см. рисунок 6).


```
[wholesale_warehouse=# SELECT * FROM wholesale_warehouse.logs;
log_info | log_time
-----+-----
Update realization with id: 4 | 2022-05-24 22:15:54.845609
Update realization with id: 4 | 2022-05-24 22:15:59.928178
Insert realization with id: 7 | 2022-05-24 22:28:53.774494
Delete realization with id: 7 | 2022-05-24 22:29:09.551543
(4 rows)

wholesale_warehouse=#
```

Рисунок 6 - Результаты логгирования

ВЫВОДЫ

В процессе работы были изучены процедуры, функции и триггеры в базе данных PostgreSQL, которые были реализованы на практике.