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:

K3240

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).

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 1 - добавление атрибутa

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

$$;

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 2 - создание и вызов процедуры 1

1. 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

$$;

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 - создание и вызов функции 2

**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;

$$;

Изображение выглядит как текст

Автоматически созданное описание

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();

Изображение выглядит как текст

Автоматически созданное описание

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).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 6 - Результаты логгирования

ВЫВОДЫ

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