Министерство науки и высшего образования Российской Федерации

федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский университет ИТМО»

Факультет инфокоммуникационных технологий

Отчет

по лабораторной работе № 3

«ПРОЦЕДУРЫ И ТРИГГЕРЫ В POSTGRESQL»

по дисциплине «**Проектирование и реализация баз данных**»

Выполнил:
студент 2 курса ИКТ
группы К32422
Демидов Максим Евгеньевич

Преподаватель:
Говорова Марина Михайловна

Санкт-Петербург
2023

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию, часть 4.
2. Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5).

Вариант 9. БД «Оптовая база»

Описание предметной области: Оптовая база закупает товары у компаний-поставщиков и поставляет их компаниям – покупателям. Доход оптовой базы составляет не менее 5% от стоимости товара, проданного компании – покупателю. Один и тот же товар может доставляться несколькими поставщиками, и один и тот же поставщик может поставлять несколько видов товаров. Цены поставки товара у разных поставщиков могут отличаться. Поставки и заказы обслуживают менеджеры по работе с клиентами (по поставкам и продажам).

1. Создание хранимой процедуры

- для снижения цены на заданный процент для товаров, у которых срок пребывания на складе превысил заданный норматив:

  Для этого мною был создан атрибут «arrival_date» в таблице storage (в таблице sale есть дата заказа и дата вывоза, но неясно, когда товар поступил на склад).

item_code integer	purchase_item integer	item_amount integer	arrival_date date
992649	489223	43	2023-05-09
292462	327233	23	2023-03-22
332074	958928	12	2023-05-13

Снижаем цену на 10% для товаров, которые лежат на складе более 30 дней:

CREATE OR REPLACE PROCEDURE warehouse.reduce_price() LANGUAGE SQL AS $$ UPDATE warehouse.sales_item SET price_for_sale = price_for_sale * 0.9 WHERE CURRENT_DATE - (SELECT arrival_date FROM warehouse.storage WHERE storage.storage_code=sales_item.storage_code) > 30 $$;

```
Вы подключены к базе данных "warehouse2" как пользователь "postgres".
warehouse2=# CREATE OR REPLACE PROCEDURE warehouse.reduce_price() LANGUAGE SQL AS $$ UPDATE warehouse.sales_item SET pri
ce_for_sale = price_for_sale * 0.9 WHERE CURRENT_DATE - (SELECT arrival_date FROM warehouse.storage WHERE storage.storag
e_code=sales_item.storage_code) > 30 $$;
CREATE PROCEDURE
warehouse2=# SELECT * FROM warehouse.sales_item;
 lot_number | delivered_item | price_for_sale | product_quantity | storage_code
------------+----------------+----------------+------------------+--------------
     427492 |         249231 |             70 |               60 |       111111
     471224 |         465232 |             54 |              240 |       222222
     830132 |         554532 |            143 |               80 |       333333
(3 ёЄЁюъш)


warehouse2=# CALL warehouse.reduce_price();
CALL
warehouse2=# SELECT * FROM warehouse.sales_item;
 lot_number | delivered_item | price_for_sale | product_quantity | storage_code
------------+----------------+----------------+------------------+--------------
     427492 |         249231 |             70 |               60 |       111111
     830132 |         554532 |            143 |               80 |       333333
     471224 |         465232 |             49 |              240 |       222222
(3 ёЄЁюъш)


warehouse2=#
```

- для расчета стоимости всех партий товаров, проданных за прошедшие сутки:

lot_number [PK] integer	employee_code integer	organization_code integer	export_date date	order_date date
427492	927943	137421	2021-07-07	2021-06-01
471224	309128	749274	2022-12-14	2022-11-08
830132	283244	589329	2023-05-28	2023-05-27

lot_number integer	delivered_item [PK] integer	price_for_sale integer	product_quantity integer
427492	249231	70	60
471224	465232	49	240
830132	554532	143	80

CREATE OR REPLACE FUNCTION warehouse.total_price_last_day() RETURNS TABLE (lot_number integer, order_date date, total numeric)

LANGUAGE SQL AS $$ SELECT s.lot_number, s.order_date, SUM(si.price_for_sale) as total FROM warehouse.sale s JOIN warehouse.sales_item si ON s.lot_number = si.lot_number WHERE s.order_date = (CURRENT_DATE - interval '1 day') GROUP BY s.lot_number, s.order_date $$;

```
warehouse2=# CREATE OR REPLACE FUNCTION warehouse.total_price_last_day() RETURN
ate, total numeric) LANGUAGE SQL AS $$ SELECT s.lot_number, s.order_date, SUM(s
.sale s JOIN warehouse.sales_item si ON s.lot_number = si.lot_number WHERE s.or
y') GROUP BY s.lot_number, s.order_date $$;
CREATE FUNCTION
warehouse2=# SELECT * from warehouse.total_price_last_day();

 lot_number | order_date | total
------------+------------+-------
     830132 | 2023-05-27 |   143
(1 строка)
```

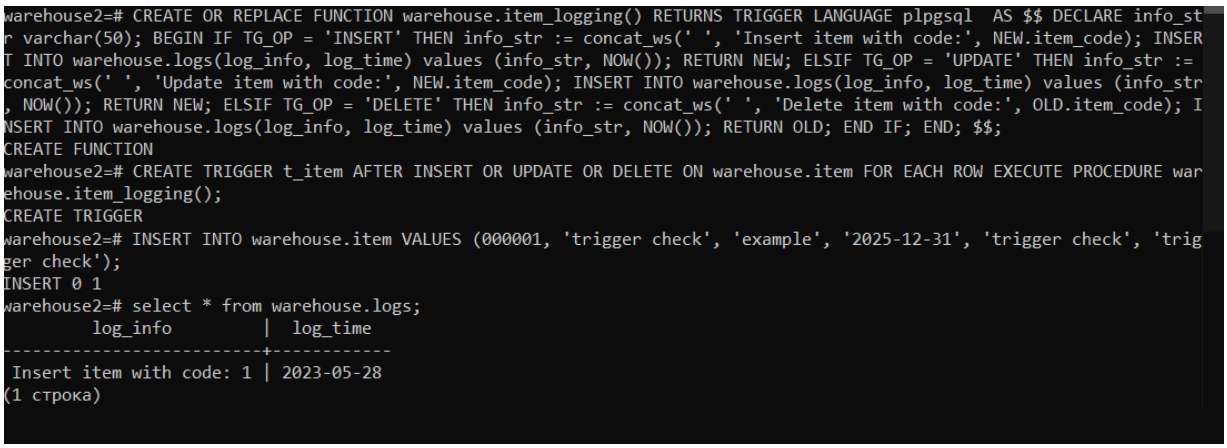
2. **Создание триггера для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL**

Функция:

CREATE OR REPLACE FUNCTION warehouse.item_logging() RETURNS TRIGGER LANGUAGE plpgsql AS $$ DECLARE info_str varchar(50); BEGIN IF TG_OP = 'INSERT' THEN info_str := concat_ws(' ', 'Insert item with code:', NEW.item_code); INSERT INTO warehouse.logs(log_info, log_time) values (info_str, NOW()); RETURN NEW; ELSIF TG_OP = 'UPDATE' THEN info_str := concat_ws(' ', 'Update item with code:', NEW.item_code); INSERT INTO warehouse.logs(log_info, log_time) values (info_str, NOW()); RETURN NEW; ELSIF TG_OP = 'DELETE' THEN info_str := concat_ws(' ', 'Delete item with code:', OLD.item_code); INSERT INTO warehouse.logs(log_info, log_time) values (info_str, NOW()); RETURN OLD; END IF; END; $$;

Триггер:

CREATE TRIGGER t_item AFTER INSERT OR UPDATE OR DELETE ON warehouse.item FOR EACH ROW EXECUTE PROCEDURE warehouse.item_logging();



Теперь происходит логирование этих событий в таблице logs.

Выводы: в процессе работы были изучены процедуры, функции и триггеры в базе данных PostgreSQL, которые были реализованы на практике. Научился работать с командной строкой psql.