Национальный исследовательский Университет ИТМО
Мегафакультет информационных и трансляционных технологий
Факультет инфокоммуникационных технологий

Базы данных

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

Лабораторная работа №3

Работу выполнил:
Нестеров В.А.
Группа: К3243
Преподаватель:
Говорова М.М.

Санкт-Петербург
2022

Содержание

1. Цель работы

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

2. Практическое задание

- Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4).
- Создать триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5). Допустимо создать универсальный триггер или отдельные триггеры на логирование действий.

3. Выполнение

3.1. Вариант 1

Скрипты кода разработанных объектов (процедур/функций и триггера) и подтверждающие скриншоты работы и результатов в psql согласно индивидуальному заданию (часть 4 и 5).

3.1.1. Хранимая процедура для снижения цены на заданный процент для товаров, у которых срок пребывания на складе превысил заданный норматив.

Снизим стоимость товаров, находящихся на Базе дольше 90 дней на 20%

	order_code character varying	order_date date	delivery_date date	selling_price integer
1	1	2022-06-22	2022-02-09	3283
2	3	2022-05-05	2021-09-06	4068
3	4	2022-11-23	2022-03-09	1013
4	7	2022-09-21	2021-10-02	2838
5	10	2022-07-22	2021-07-24	6638
6	12	2022-04-16	2021-09-03	441
7	16	2023-01-25	2022-02-18	7419
8	18	2023-03-03	2021-07-11	4023
9	19	2022-08-28	2021-08-25	1523
10	20	2023-02-28	2021-11-24	7876
11	21	2022-12-31	2022-03-07	6288
12	22	2022-05-03	2021-12-31	4111
13	23	2022-12-21	2022-05-30	6523
14	24	2022-08-05	2021-06-28	3716

Рисунок 3.1. Таблица до выполнения функции

```
CREATE PROCEDURE PUBLIC.SPOILED_GOODS_PRICE_REDUCE() LANGUAGE SQL AS $$
UPDATE "Order_list"
SET SELLING_PRICE = "Order_list".SELLING_PRICE / 0.8
FROM
(SELECT "Order_list".ORDER_CODE,
ORDER_DATE,
DELIVERY_DATE,
SELLING_PRICE
FROM "Order_list"
LEFT JOIN "Orders" ON "Order_list".ORDER_CODE = "Orders".ORDER_CODE
LEFT JOIN "Delivery" ON "Orders".EMPL_CODE = "Delivery".EMPL_CODE
WHERE (ORDER_DATE - DELIVERY_DATE) > '90') AS FOO
$$;

CALL PUBLIC.SPOILED_GOODS_PRICE_REDUCE();
```

	order_code character varying	order_date date	delivery_date date	selling_price integer
1	1	2022-06-22	2022-02-09	2626
2	3	2022-05-05	2021-09-06	3254
3	4	2022-11-23	2022-03-09	810
4	7	2022-09-21	2021-10-02	2270
5	10	2022-07-22	2021-07-24	5310
6	12	2022-04-16	2021-09-03	353
7	16	2023-01-25	2022-02-18	5935
8	18	2023-03-03	2021-07-11	3218
9	19	2022-08-28	2021-08-25	1218
10	20	2023-02-28	2021-11-24	6301
11	21	2022-12-31	2022-03-07	5030
12	22	2022-05-03	2021-12-31	3289
13	23	2022-12-21	2022-05-30	5218
14	24	2022-08-05	2021-06-28	2973

Рисунок 3.2. Таблица после выполнения функции

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

```
CREATE PROCEDURE PUBLIC.total_value_last_day() LANGUAGE SQL AS $$
SELECT "Orders".ORDER_CODE,
ORDER_EXECUTION_DATE,
SUM(SELLING_PRICE) AS TOTAL
FROM "Orders"
LEFT JOIN "Order_list" ON "Orders".ORDER_CODE = "Order_list".ORDER_CODE
WHERE ORDER_EXECUTION_DATE = (CURRENT_DATE - interval '1 day')
GROUP BY "Orders".ORDER_CODE
$$;

CALL PUBLIC.total_value_last_day();
```

	order_code [PK] character varying	order_execution_date date	total bigint
1	101	2022-06-29	2654

Рисунок 3.3. Результат работы функции

3.1.3. Создать триггеры для логирования событий вставки, удаления, редактирования данных.

Была добавлена таблица logs для логирования событий

```
CREATE OR REPLACE FUNCTION PUBLIC.ADD_TO_LOG() RETURNS TRIGGER AS $$
DECLARE
mstr varchar(30);
astr varchar(100);
retstr varchar(254);
BEGIN
IF TG_OP = 'INSERT' THEN
astr = NEW;
mstr := 'Added new data: ';
retstr := mstr||astr;
INSERT INTO public.logs(text, added, table_name) values (retstr, NOW(), TG_TABLE_NAME);
return NEW;

ELSIF TG_OP = 'UPDATE' THEN
astr = NEW;
mstr := 'Updated data: ';
retstr := mstr||astr;
INSERT INTO public.logs(text, added, table_name) values (retstr, NOW(), TG_TABLE_NAME);
RETURN NEW;

ELSIF TG_OP = 'DELETE' THEN
astr = OLD;
mstr := 'Remove data ';
retstr := mstr || astr;
INSERT INTO public.logs(text, added, table_name) values (retstr, NOW(), TG_TABLE_NAME);
RETURN OLD;

END IF;
END;
$$ LANGUAGE PLPGSQL;

CREATE TRIGGER tr_supplier AFTER INSERT OR UPDATE OR DELETE
ON public."Supplier" FOR EACH ROW EXECUTE PROCEDURE public.add_to_log();
INSERT INTO public."Supplier"(supplier_code, supplier_name, supplier_address)
VALUES ('101', 'OZON LLC', 'Somewhere idk');
UPDATE public."Supplier" SET supplier_address = 'Moscow, ul. Krasnogo Textilschika' WHER
DELETE FROM public."Supplier" WHERE supplier_code = '101';
```

```
SELECT * FROM public.logs
```

	text text	added date	table_name character varying
1	Added new data: (101,"OZON LLC","Somewhere idk")	2022-06-30	Supplier
2	Updated data: (101,"OZON LLC","Moscow, ul. Krasnogo Textilschika")	2022-06-30	Supplier
3	Remove data (101,"OZON LLC","Moscow, ul. Krasnogo Textilschika")	2022-06-30	Supplier

Рисунок 3.4. Результат работы триггера

4. Выводы

В данной работе мы овладели практическими навыками по созданию, заполнению рабочими данными и восстановлению базы данных PostgreSQL с использованием pgAdmin4.