Национальный исследовательский Университет ИТМО
Мегафакультет информационных и трансляционных технологий
Факультет мобильных и сетевых технологий

Проектирование и реализация баз данных

Лабораторная работа №5

Процедуры, функции, триггеры в PostgreSQL

Работу выполнил:
Фадеев Д.А.
Группа: К3239
Преподаватель:
Говорова М.М.

Санкт-Петербург
2023

Цель Работы

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

Практическое задание

- Создать процедуры/функции согласно индивидуальному заданию (часть 4).
- Модифицировать триггер (триггерную функцию) на проверку корректности входа и выхода сотрудника (см. Практическое задание 1 Лабораторного практикума (Приложение)) с максимальным учетом «узких» мест некорректных данных по входу и выходу).
- Создать авторский триггер по варианту индивидуального задания.

Выполнение

Процедуры и функции

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

```
CREATE OR REPLACE PROCEDURE wholesale_base.reduce_the_price_of_purchase_list()
LANGUAGE SQL
AS $$
    UPDATE wholesale_base.purchase_list pl
    SET price = price * 0.8
    FROM wholesale_base.purchase pu
    WHERE pu.purchase_id = pl.purchase_id
    AND pu.date_of_purchase <= CURRENT_DATE - INTERVAL '4 year'
$$

CREATE OR REPLACE PROCEDURE wholesale_base.increase_the_price_of_purchase_list()
LANGUAGE SQL
AS $$
    UPDATE wholesale_base.purchase_list pl
    SET price = price * 1.25
    FROM wholesale_base.purchase pu
    WHERE pu.purchase_id = pl.purchase_id
    AND pu.date_of_purchase <= CURRENT_DATE - INTERVAL '4 year'
$$
```

  - До применения

```
wholesale_base=# SELECT pl.purchase_list_id, pu.date_of_purchase, pl.price, pl.c
ount
FROM wholesale_base.purchase_list pl
JOIN wholesale_base.purchase pu ON pl.purchase_id = pu.purchase_id
WHERE date_of_purchase <= CURRENT_DATE - INTERVAL '4 year';
 purchase_list_id | date_of_purchase | price | count
------------------+------------------+-------+-------
                6 | 2019-05-15       |   251 |    83
               36 | 2019-05-15       |   152 |    49
               75 | 2019-06-20       |   495 |    83
               80 | 2019-08-30       |   117 |    64
                2 | 2019-08-30       |   577 |    51
               17 | 2019-10-10       |   555 |    74
               35 | 2019-07-25       |   315 |    21
               16 | 2019-10-10       |   275 |    55
               27 | 2019-07-25       |   579 |    27
               30 | 2019-09-05       |   171 |    24
               42 | 2019-08-30       |   366 |    75
               69 | 2019-10-10       |   481 |    69
               83 | 2019-07-25       |   392 |    37
(13 строк)
```

– После применения

```
wholesale_base=# CAll wholesale_base.reduce_the_price_of_purchase_list();
CALL
wholesale_base=# SELECT pl.purchase_list_id, pu.date_of_purchase, pl.price, pl.c
ount
FROM wholesale_base.purchase_list pl
JOIN wholesale_base.purchase pu ON pl.purchase_id = pu.purchase_id
WHERE date_of_purchase <= CURRENT_DATE - INTERVAL '4 year';
 purchase_list_id | date_of_purchase | price | count
------------------+------------------+-------+-------
                6 | 2019-05-15       |   201 |    83
               36 | 2019-05-15       |   122 |    49
               75 | 2019-06-20       |   396 |    83
               80 | 2019-08-30       |    94 |    64
                2 | 2019-08-30       |   462 |    51
               17 | 2019-10-10       |   444 |    74
               35 | 2019-07-25       |   252 |    21
               16 | 2019-10-10       |   220 |    55
               27 | 2019-07-25       |   463 |    27
               30 | 2019-09-05       |   137 |    24
               42 | 2019-08-30       |   293 |    75
               69 | 2019-10-10       |   385 |    69
               83 | 2019-07-25       |   314 |    37
(13 строк)
```

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

– Код

```
CREATE OR REPLACE PROCEDURE wholesale_base.calculate_income_for_last_month()
LANGUAGE plpgsql
AS $$
DECLARE
    total_income INT;
BEGIN
    SELECT SUM(income) INTO total_income
    FROM(
        SELECT o.date_of_sale AS date_day, SUM(ol.count * ol.price) AS income
        FROM wholesale_base.order o
        JOIN wholesale_base.order_list ol ON o.order_id = ol.order_id
        WHERE o.date_of_sale BETWEEN CURRENT_DATE - INTERVAL '1 year' AND CURRENT_DATE
        GROUP BY o.date_of_sale
    );
    RAISE NOTICE 'Total income for last year is: %', total_income;
END;
$$;
```

– Применине

```
wholesale_base=# CALL wholesale_base.calculate_income_for_last_year();
NOTICE:  Total income for last year is: 254947
CALL
wholesale_base=#
```

Триггеры

- Триггер проверяет достаточно ли остатков для конкретного товара на складе, для того чтобы использовать во вставке к покупке конкретного товара со склада
  - Код функции

```
CREATE OR REPLACE FUNCTION wholesale_base.check_if_leftovers_enough()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    IF (SELECT leftovers FROM wholesale_base.purchase_list WHERE purchase_list_id = NEW.purchase_list_id) < NEW.cou
    THEN
        RAISE EXCEPTION 'The count cannot be greater than leftovers';
    END IF;

    RETURN NEW;
END;
$BODY$;
```

  - Код триггера

```
CREATE OR REPLACE TRIGGER check_if_leftovers_enough_before_insert
    BEFORE INSERT
    ON wholesale_base.order_list
    FOR EACH ROW
    EXECUTE FUNCTION wholesale_base.check_if_leftovers_enough();
```

  - Применение

```
wholesale_base=# INSERT INTO wholesale_base.order_list(price,
wholesale_base(# count, order_id, purchase_list_id)
wholesale_base-# VALUES (258,29,31,19);
ERROR:  The count cannot be greater than leftovers
КОНТЕКСТ:  PL/pgSQL function wholesale_base.check_if_leftovers_enough() line 5 a
t RAISE
wholesale_base=#
```

- Триггер автоматически вычитает количество купленного товара из его остатков на складе
  - Код функции

```
CREATE OR REPLACE FUNCTION wholesale_base.subtract_from_leftovers()
    RETURNS trigger
    LANGUAGE 'plpgsql'
    COST 100
    VOLATILE NOT LEAKPROOF
AS $BODY$
BEGIN
    UPDATE wholesale_base.purchase_list
    SET leftovers = leftovers - NEW.count
    WHERE purchase_list_id = NEW.purchase_list_id;

    RETURN NEW;
END;
$BODY$;

ALTER FUNCTION wholesale_base.subtract_from_leftovers()
    OWNER TO postgres;
```

  - Код триггера

```
CREATE OR REPLACE TRIGGER subtract_count_from_leftovers
    AFTER INSERT
    ON wholesale_base.order_list
    FOR EACH ROW
    EXECUTE FUNCTION wholesale_base.subtract_from_leftovers();
```

– Применение

```
wholesale_base=# SELECT leftovers FROM wholesale_base.purchase_list
WHERE purchase_list_id = 32;
 leftovers
-----------
        79
(1 строка)

wholesale_base=# INSERT INTO wholesale_base.order_list(price,
count, order_id, purchase_list_id)
VALUES (258,29,31,32);
INSERT 0 1
wholesale_base=# SELECT leftovers FROM wholesale_base.purchase_list
WHERE purchase_list_id = 32;
 leftovers
-----------
        50
(1 строка)

wholesale_base=#
```

Модифицирование триггера

- Код функции

```
create or replace function fn_check_time_punch() returns trigger as $psql$
    begin
        if new.punch_time > now()

        or new.is_out_punch = (
          select tps.is_out_punch
          from time_punch tps
          where tps.employee_id = new.employee_id
          order by tps.id desc limit 1
        )
        or new.punch_time <= (
            select tps.punch_time from time_punch tps
            where tps.employee_id = new.employee_id
            order by tps.id desc limit 1
        )
        then
            return null;
        end if;
        return new;
    end;
$psql$ language plpgsql;
```

- Код триггера

```
drop trigger if exists check_time_punch on time_punch;
create trigger check_time_punch before insert on time_punch
for each row execute procedure fn_check_time_punch();
```

- Применение

```
emp_time=# INSERT INTO time_punch(employee_id, is_out_punch, punch_time)
VALUES
(3, false, '2022-01-01 00:00:00'),
(3, true, '2022-01-01 02:00:00');
INSERT 0 2
emp_time=# --Выход раньше входа--
emp_time=# INSERT INTO time_punch(employee_id, is_out_punch, punch_time)
VALUES
(3, false, '2022-01-01 04:00:00'),
(3, true, '2022-01-01 03:00:00');
INSERT 0 1
emp_time=# --Второй выход к ряду--
emp_time=# INSERT INTO time_punch(employee_id, is_out_punch, punch_time)
VALUES
(3, false, '2022-01-01 05:00:00');
INSERT 0 0
emp_time=#
```

Вывод

В ходе выполнения лабораторной работы были созданы и протестированны процедуры, функции и триггеры.