Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное
учреждение высшего образования
“Национальный исследовательский университет ИТМО”

Факультет инфокоммуникационных технологий

ЛАБОРАТОРНАЯ РАБОТА №3

ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL.

по дисциплине:
«Проектирование и реализация баз данных»

Выполнил:
студент 2 курса ИКТ
группы К3241
Попов Ньургун

Санкт-Петербург
2022

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

Практическое задание:

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4);
2. 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.

Индивидуальное задание: Вариант 20. БД «Автозаправка»

Описание предметной области: Фирмы–поставщики автомобильного топлива имеют сеть заправочных автостанций. На автозаправках реализуется автомобильное топливо всех видов. Топливо продается за безналичный расчет с помощью специальных пластиковых карт. База данных предназначена для анализа продаж автомобильного топлива клиентам по видам топлива в сети заправок конкретной фирмы-производителя (поставщика топлива), спроса на автомобильное топливо и т. д. Каждая фирма имеет несколько автозаправок. Каждый вид топлива предоставляется несколькими фирмами.

БД должна содержать следующий минимальный набор сведений: Карта-счет клиента. Сумма на счете клиента. Ф.И.О. клиента. Адрес клиента. Телефон клиента. Код автозаправки. Адрес автозаправки. Название фирмы. Юридический адрес. Телефон. Код топлива. Вид топлива. Единица измерения. Цена (руб.) за литр. Дата продажи топлива. Количество топлива. Код фирмы-поставщика. Фирма-поставщик топлива. Юридический адрес. Сроки действия цены на топливо.

Задание 4. Создать хранимые процедуры:

- Вывести сведения обо всех покупках одного из клиентов за заданную дату (данные клиента, дата, объем топлива, уплаченная сумма);
- Количество видов топлива, поставляемых каждой фирмой-поставщиком;
- Самый непопулярный вид топлива за прошедшую неделю.

Задание 5. Создать необходимые триггеры.

Выполнение:

1. Создать процедуры/функции согласно индивидуальному заданию и (согласно индивидуальному заданию, часть 4):
   – Вывести сведения обо всех покупках одного из клиентов за заданную дату (данные клиента, дата, объем топлива, уплаченная сумма):

   Скрипт кода функции:

```
CREATE OR REPLACE FUNCTION customer_info(customer_name text, date_of_sale date)
RETURNS TABLE (customer_data text, sale_date date, sale_quantity int, paid_sum int)
AS
$$
BEGIN
    RETURN QUERY
    SELECT full_name, sales_date, sales_quantity, sales_quantity*final_price_for_liter
    FROM gss.bill, gss.fuel_sales
    WHERE full_name = customer_name
        AND sales_date = date_of_sale;
END;
$$ LANGUAGE plpgsql;
```

   Скрипт кода функции в psql:

```
gas=# CREATE OR REPLACE FUNCTION customer_info(customer_name text, date_of_sale date)
gas-# RETURNS TABLE (customer_data text, sale_date date, sale_quantity int, paid_sum int)
gas-# AS
gas-# $$
gas$# BEGIN
gas$# RETURN QUERY
gas$# SELECT full_name, sales_date, sales_quantity, sales_quantity*final_price_for_liter
gas$# FROM gss.bill, gss.fuel_sales
gas$# WHERE full_name = customer_name
gas$# AND sales_date = date_of_sale;
gas$# END;
gas$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

   Результаты в psql:

```
gas=# select * from customer_info('Смирнов Владимир Максимович', '2022-07-23');
        customer_data        | sale_date  | sale_quantity | paid_sum
-----------------------------+------------+---------------+----------
 Смирнов Владимир Максимович | 2022-07-23 |            18 |     1584
(1 row)
```

– Количество видов топлива, поставляемых каждой фирмой-поставщиком:

Скрипт кода функции:

```
CREATE OR REPLACE FUNCTION fuel_info()
RETURNS TABLE (supplier text, fuel_quantity bigint)
AS
$$
BEGIN
    RETURN QUERY
    SELECT supplier_name, COUNT(*)
    FROM gss.supplier_company, gss.fuel
    WHERE supplier_company.supplier_code = fuel.supplier_code
    group by supplier_name;
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода функции в psql:

```
gas=# CREATE OR REPLACE FUNCTION fuel_info()
gas-# RETURNS TABLE (supplier text, fuel_quantity bigint)
gas-# AS
gas-# $$
gas$# BEGIN
gas$# RETURN QUERY
gas$# SELECT supplier_name, COUNT(*)
gas$# FROM gss.supplier_company, gss.fuel
gas$# WHERE supplier_company.supplier_code = fuel.supplier_code
gas$# group by supplier_name;
gas$# END;
gas$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

Результаты в psql:

```
gas=# select * from fuel_info();
   supplier   | fuel_quantity
--------------+---------------
 Лукойл       |             3
 Газпромнефть |             2
 Роснефть     |             4
 ПТК          |             2
(4 rows)
```

– Самый непопулярный вид топлива за прошедшую неделю:

Скрипт кода функции:

```
CREATE OR REPLACE FUNCTION unpopular_fuel()
RETURNS TABLE (type text)
AS
$$
BEGIN
    RETURN QUERY
    SELECT fuel_type
    FROM gss.fuel_guide, gss.fuel, gss.fuel_sales
    WHERE fuel_guide.fuel_code = fuel.fuel_code
        AND fuel.fuel_code_supplied = fuel_sales.fuel_code_supplied
            AND (sales_date >= current_date - integer '7'
                AND sales_date < current_date)
    GROUP BY fuel_type
    HAVING COUNT(*) <= ALL (SELECT COUNT(*)
                            FROM gss.fuel_guide, gss.fuel, gss.fuel_sales
                            WHERE fuel_guide.fuel_code = fuel.fuel_code
                                AND fuel.fuel_code_supplied = fuel_sales.fuel_code_supplied
                                    AND (sales_date >= current_date - integer '7'
                                        AND sales_date < current_date)
                            GROUP BY fuel_type);
END;
$$ LANGUAGE plpgsql;
```

Скрипт кода функции в psql:

```
gas=# CREATE OR REPLACE FUNCTION unpopular_fuel()
gas-# RETURNS TABLE (type text)
gas-# AS
gas-# $$
gas$# BEGIN
gas$# RETURN QUERY
gas$# SELECT fuel_type
gas$# FROM gss.fuel_guide, gss.fuel, gss.fuel_sales
gas$# WHERE fuel_guide.fuel_code = fuel.fuel_code
gas$# AND fuel.fuel_code_supplied = fuel_sales.fuel_code_supplied
gas$# AND (sales_date >= current_date - integer '7'
gas$# AND sales_date < current_date)
gas$# GROUP BY fuel_type
gas$# HAVING COUNT(*) <= ALL (SELECT COUNT(*)
gas$#         FROM gss.fuel_guide, gss.fuel, gss.fuel_sales
gas$# WHERE fuel_guide.fuel_code = fuel.fuel_code
gas$#     AND fuel.fuel_code_supplied = fuel_sales.fuel_code_supplied
gas$# AND (sales_date >= current_date - integer '7'
gas$# AND sales_date < current_date)
gas$#       GROUP BY fuel_type);
gas$# END;
gas$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

Результаты в psql:

```
gas=# select * from unpopular_fuel();
        type
--------------------
 дизельное топливо
(1 row)
```

2. 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:
   Для начала создал таблицу gss.events:

```
create table gss.events(option_name text, table_name name, changes varchar(254), adding_time date)
```

   Создание триггерной функции:

```
CREATE OR REPLACE FUNCTION add_new_events() RETURNS TRIGGER AS $$
DECLARE
    what_happened text;
    main varchar(254);
BEGIN
    IF TG_OP = 'INSERT' THEN
        what_happened := 'Added a new';
        main = NEW;
        INSERT INTO gss.events(option_name, table_name, changes, adding_time)
        values(what_happened, TG_RELNAME, main, now());
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        what_happened := 'Updated';
        main = NEW;
        INSERT INTO gss.events(option_name, table_name, changes, adding_time)
        values(what_happened, TG_RELNAME, main, now());
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        what_happened := 'Deleted';
        main = OLD;
        INSERT INTO gss.events(option_name, table_name, changes, adding_time)
        values(what_happened, TG_RELNAME, main, now());
        RETURN OLD;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

```
gas=# CREATE OR REPLACE FUNCTION add_new_events() RETURNS TRIGGER AS $$
gas$# DECLARE
gas$# what_happened text;
gas$# main varchar(254);
gas$# BEGIN
gas$# IF TG_OP = 'INSERT' THEN
gas$# what_happened := 'Added a new';
gas$# main = NEW;
gas$# INSERT INTO gss.events(option_name, table_name, changes, adding_time)
gas$#
OVERRIDING  SELECT      TABLE       VALUES
gas$# values(what_happened, TG_RELNAME, main, now());
gas$# RETURN NEW;
gas$# ELSIF TG_OP = 'UPDATE' THEN
gas$# what_happened := 'Updated';
gas$# main = NEW;
gas$# INSERT INTO gss.events(option_name, table_name, changes, adding_time)
gas$#
OVERRIDING  SELECT      TABLE       VALUES
gas$# values(what_happened, TG_RELNAME, main, now());
gas$# RETURN NEW;
gas$# ELSIF TG_OP = 'DELETE' THEN
gas$# what_happened := 'Deleted';
gas$# main = OLD;
gas$# INSERT INTO gss.events(option_name, table_name, changes, adding_time)
gas$#
OVERRIDING  SELECT      TABLE       VALUES
gas$# values(what_happened, TG_RELNAME, main, now());
gas$# RETURN OLD;
gas$# END IF;
gas$# END;
gas$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
```

Создание самого триггера:

```
1  CREATE TRIGGER customer_trigger AFTER INSERT OR UPDATE OR DELETE
2      ON gss.customer FOR EACH ROW EXECUTE PROCEDURE add_new_events()
```

```
gas=# CREATE TRIGGER customer_trigger AFTER INSERT OR UPDATE OR DELETE
gas-# ON gss.customer FOR EACH ROW EXECUTE PROCEDURE add_new_events();
CREATE TRIGGER
```

Реализация insert into, update, delete:

```
INSERT INTO gss.customer (full_name, customer_address, customer_phone_number)
    VALUES ('Адамов Адам Адамович', 'Бармалеева 9', 89812345678);
UPDATE gss.customer
    SET customer_address = 'Сытнинская 16'
        WHERE full_name = 'Адамов Адам Адамович';
DELETE FROM gss.customer
    WHERE full_name = 'Адамов Адам Адамович';
```

```
gas=# INSERT INTO gss.customer (full_name, customer_address, customer_phone_number)
gas-# VALUES ('Адамов Адам Адамович', 'Бармалеева 9', 89812345678);
INSERT 0 1
gas=# UPDATE gss.customer
gas-# SET customer_address = 'Сытнинская 16'
gas-# WHERE full_name = 'Адамов Адам Адамович';
UPDATE 1
gas=# DELETE FROM gss.customer
gas-# WHERE full_name = 'Адамов Адам Адамович';
DELETE 1
```

Результаты:

```
gas=# select * from gss.events;
 option_name | table_name |                        changes                         | adding_time
-------------+------------+--------------------------------------------------------+-------------
 Added a new | customer   | ("Адамов Адам Адамович","Бармалеева 9",89812345678)   | 2022-05-10
 Updated     | customer   | ("Адамов Адам Адамович","Сытнинская 16",89812345678) | 2022-05-10
 Deleted     | customer   | ("Адамов Адам Адамович","Сытнинская 16",89812345678) | 2022-05-10
(3 rows)
```

Выводы:

В результате выполненной работы:

- были созданы процедуры/функции
- был создан триггер для логирования событий вставки, удаления, редактирования данных.