Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

«ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL»

по дисциплине «Проектирование и реализация баз данных»

Обучающийся Проскуряков Роман Владимирович
Факультет прикладной информатики
Группа K3239
Направление подготовки 09.03.03 Прикладная информатика
Образовательная программа Мобильные и сетевые технологии 2023
Преподаватель Говорова Марина Михайловна

Санкт-Петербург
2024/2025

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

Практическое задание:

1. Создать 3 процедуры для индивидуальной БД согласно варианту (часть 4 ЛР 2). Допустимо использование IN/OUT параметров. Допустимо создать авторские процедуры. (3 балла)
2. Создать триггеры для индивидуальной БД согласно варианту:
   7 оригинальных триггеров - 7 баллов (max).

Часть 4 ЛР 2. Создать хранимые процедуры:

- Вывести сведения обо всех покупках одного из клиентов за заданную дату (данные клиента, дата, объем топлива, уплаченная сумма).
- Посчитать количество видов топлива, поставляемых каждой фирмой-поставщиком.
- Добавить новую АЗГС фирмы-производителя.

Выполнение:

1. Создать хранимые процедуры

- Вывести сведения обо всех покупках одного из клиентов за заданную дату (данные клиента, дата, объем топлива, уплаченная сумма)

```
CREATE OR REPLACE function clientPurchasesPerDay (IN idClient INT, IN dayDate DATE)
	RETURNS TABLE(surname CHARACTER VARYING, name CHARACTER VARYING,
patronymic CHARACTER VARYING,
		phone_number bigint, address CHARACTER VARYING,
		sale_date timestamp without time zone, sold_liters_volume int, id_card int, money_spent
int)
	LANGUAGE plpgsql
AS $$
BEGIN
	RAISE NOTICE 'Данные о покупках клиента с ID: % за: %', idClient, dayDate;

	RETURN QUERY
	SELECT clients.surname, clients.name, clients.patronymic,
		clients.phone_number, clients.address,
		sales.sale_date, sales.sold_liters_volume, sales.id_card,
		((sales.sold_liters_volume * fuel_prices.per_liter) * (100 - discount_percent) / 100 -
discount_rub) AS money
	FROM
	clients
	JOIN client_cards
	ON clients.id_client=client_cards.id_client
	JOIN sales
	ON client_cards.id_card=sales.id_card
	JOIN fuel_prices
	ON sales.id_fuel_price=fuel_prices.id_fuel_price
	WHERE clients.id_client=idClient
		AND   sales.sale_date::timestamp::date = dayDate;
END;
$$;
```

```
refill=# SELECT * FROM clientPurchasesPerDay(1, '2025-07-04');
ЗАМЕЧАНИЕ:  Данные о покупках клиента с ID: 1 за: 2025-07-04
 surname | name | patronymic | phone_number | address |      sale_date      | sold_liters_volume | id_card | money_spent
---------+------+------------+--------------+---------+---------------------+--------------------+---------+-------------
 Иванов  | Иван | Иванович   |  89001234567 | Москва  | 2025-07-04 03:20:00 |                 40 |       1 |        1850
 Иванов  | Иван | Иванович   |  89001234567 | Москва  | 2025-07-04 03:20:00 |                 32 |       4 |        1600
(2 строки)
```

- Посчитать количество видов топлива, поставляемых каждой фирмой-поставщиком.

```
CREATE OR REPLACE procedure fuelTypesCountBySupplier(INOUT __resultTable refcursor)
	LANGUAGE plpgsql
AS $$
BEGIN
	OPEN __resultTable FOR
		SELECT companies.id_company, count(id_kind_fuel) as amount
		FROM companies
		LEFT JOIN produced_fuel
		ON companies.id_company=produced_fuel.id_company_factory
		WHERE type_company=1
		GROUP BY id_company;
END;
$$;
```

```
refill=# BEGIN;
BEGIN
refill=*# CALL fuelTypesCountBySupplier('cursor');
 __resulttable
---------------
 cursor
(1 строка)


refill=*# FETCH ALL FROM cursor;
 id_company | amount
------------+--------
          9 |      3
         12 |      2
         13 |      2
(3 строки)


refill=*# COMMIT;
COMMIT
```

- Добавить новую АЗГС фирмы-производителя.

```
CREATE OR REPLACE procedure addFillingStations(idCompanyOwner int, address CHARACTER VARYING)
	LANGUAGE plpgsql
AS $$
BEGIN
	IF 0 not in (SELECT type_company FROM companies WHERE id_company=idCompanyOwner)
	THEN
		RAISE NOTICE 'Компания с ID % не является компанией владельцем', idCompanyOwner;
		RETURN;
	END IF;

	INSERT INTO filling_stations(id_company_owner, station_address) VALUES(idCompanyOwner, address);
END;
$$;
```

```
refill=# CALL addFillingStations(11, 'Проверка успеха');
CALL
refill=# CALL addFillingStations(12, 'Проверка неудачи');
ЗАМЕЧАНИЕ:  Компания с ID 12 не является компанией владельцем
CALL
```

2. Создать 7 оригинальных триггеров для индивидуальной БД
    1. Триггер 1 – проверка, что цена соответствует времени продажи

```
create or replace function fn_check_fuel_price_is_actual() returns
trigger as $$
begin
	if (EXISTS
		(
			SELECT id_fuel_price FROM fuel_prices
			WHERE new.id_fuel_price=fuel_prices.id_fuel_price AND
			(start_time<=new.sale_date and (end_time IS NULL OR
new.sale_date<end_time))
		)
	) then
		return new;
	end if;
	return null;
end;
$$ language plpgsql;

create or replace trigger check_fuel_price_is_actual before insert or update on sales
for each row execute procedure fn_check_fuel_price_is_actual();
```

Не вставилось

```
refill=# SELECT count(*) FROM sales;
 count
-------
    44
(1 строка)


refill=# INSERT INTO sales(id_fuel_price, id_card, sale_date, sold_liters_volume)
refill-# values(1, 1, '2024-01-01 00:00:00', 20242024);
INSERT 0 0
refill=# SELECT count(*) FROM sales;
 count
-------
    44
(1 строка)


refill=# SELECT * FROM fuel_prices LIMIT 1;
 id_fuel_price | id_fuel_offered |     start_time      |      end_time       | per_liter
---------------+-----------------+---------------------+---------------------+-----------
             1 |               5 | 2023-01-01 00:00:00 | 2023-01-31 23:59:59 |        50
(1 строка)
```

Вставилось

```
refill=# SELECT * FROM fuel_prices LIMIT 5;
 id_fuel_price | id_fuel_offered |     start_time      |      end_time       | per_liter
---------------+-----------------+---------------------+---------------------+-----------
             1 |               5 | 2023-01-01 00:00:00 | 2023-01-31 23:59:59 |        50
             2 |               6 | 2023-02-01 00:00:00 | 2023-02-28 23:59:59 |        55
             3 |               7 | 2023-03-01 00:00:00 | 2023-03-31 23:59:59 |        60
             4 |               8 | 2023-04-01 00:00:00 | 2023-04-30 23:59:59 |        65
             5 |               7 | 2023-04-01 00:00:00 |                     |      1000
(5 строк)


refill=# INSERT INTO sales(id_fuel_price, id_card, sale_date, sold_liters_volume)
refill-# values(5, 1, '2024-01-01 00:00:00', 20242024)
refill-# ;
INSERT 0 1
```

    2. Триггер 2 – нельзя совершить покупку по недействительной карте

```
create or replace function fn_check_validity_client_card() returns
trigger as $$
begin
	if (Exists(
		select 1 from client_cards
			where client_cards.id_card=new.id_card
			and new.sale_date >= start_date
			and (end_date is NULL or new.sale_date < end_date))
	) then
		return new;
	end if;
	return null;
end;
$$ language plpgsql;

create or replace trigger check_validity_client_card before insert or update on sales
for each row execute procedure fn_check_validity_client_card();
```

```
refill=# select * from client_cards LIMIT 1;
 id_card | id_company | id_client | start_date |  end_date  | balance | discount_percent | discount_rub
---------+------------+-----------+------------+------------+---------+------------------+--------------
       2 |          9 |         2 | 2023-02-01 | 2024-02-01 |    2000 |               10 |          100
(1 строка)


refill=# INSERT INTO sales(id_fuel_price, id_card, sale_date, sold_liters_volume)
refill-# values(7, 2, '2023-02-01 00:00:00', 20242024);
INSERT 0 1
refill=# INSERT INTO sales(id_fuel_price, id_card, sale_date, sold_liters_volume)
refill-# values(7, 2, '2024-02-01 00:00:00', 20242024);
INSERT 0 0
```

3. Триггер 3 – изменять в карте можно только владельца (человека и даже компанию), баланс и двигать конец действия в рамках будущего

```
create or replace function fn_check_client_card_changes() returns
trigger as $$
begin
	if (new.start_date!=old.start_date
		or new.discount_rub!=old.discount_rub
		or new.discount_percent!=old.discount_percent
		or (new.end_date is not null and new.end_date <= current_date)
	) then
		return null;
	end if;
	return new;
end;
$$ language plpgsql;

create or replace trigger check_client_card_changes before update on client_cards
for each row execute procedure fn_check_client_card_changes();
```

```
refill=# update client_cards set end_date='2024.06.23' where id_card=4;
UPDATE 0
refill=# update client_cards set end_date='2025.06.23' where id_card=4;
UPDATE 0
refill=# update client_cards set end_date='2025.06.27' where id_card=4;
UPDATE 1
refill=# update client_cards set end_date='2025.06.26' where id_card=4;
UPDATE 0
```

4. Триггер 4 – новая цена на топливо не пересекается по времени с другими на этот же товар (не должно быть двух одновременно действующих цен на одно продаваемое топливо)

```
create or replace function fn_check_uniqueness_of_price() returns
```

```
trigger as $$
begin
	if(exists(
		select 1 from fuel_prices
		where fuel_prices.id_fuel_offered=new.id_fuel_offered
			and start_time>=new.start_time
	)) then
		return null;
	end if;

	CREATE TEMP TABLE old_actings_prices(id_fuel_price int);

	INSERT INTO old_actings_prices select id_fuel_price from fuel_prices
	where fuel_prices.id_fuel_offered=new.id_fuel_offered
		and ((end_time is null) or
			(end_time>new.start_time));

	if(exists(SELECT * FROM old_actings_prices))
	then
		update fuel_prices set end_time=new.start_time
		where fuel_prices.id_fuel_price in (SELECT * FROM old_actings_prices);
	end if;

	DROP TABLE old_actings_prices;
	return new;
end;
$$ language plpgsql;

create or replace trigger check_uniqueness_of_price before insert on fuel_prices
for each row execute procedure fn_check_uniqueness_of_price();
```

Ничего не добавилось

```
refill=# SELECT * FROM fuel_prices ORDER By id_fuel_offered limit 5;
 id_fuel_price | id_fuel_offered |     start_time      |      end_time       | per_liter
---------------+-----------------+---------------------+---------------------+-----------
             1 |               5 | 2023-01-01 00:00:00 | 2023-01-31 23:59:59 |        50
             2 |               6 | 2023-02-01 00:00:00 | 2023-02-28 23:59:59 |        55
             3 |               7 | 2023-03-01 00:00:00 | 2023-03-31 23:59:59 |        60
             5 |               7 | 2023-04-01 00:00:00 |                     |      1000
             4 |               8 | 2023-04-01 00:00:00 | 2023-04-30 23:59:59 |        65
(5 строк)


refill=# INSERT INTO fuel_prices(id_fuel_offered, start_time, end_time, per_liter)
refill-# values(5, '2023-01-01 00:00:00', '2023-01-31 23:59:59', 50);
INSERT 0 0
```

Предыдущая цена автоматически завершилась на начале новой добавленной

```
refill=# SELECT * FROM fuel_prices ORDER By id_fuel_offered limit 5;
 id_fuel_price | id_fuel_offered |     start_time      |      end_time       | per_liter
---------------+-----------------+---------------------+---------------------+-----------
             1 |               5 | 2023-01-01 00:00:00 | 2023-01-31 23:59:59 |        50
             2 |               6 | 2023-02-01 00:00:00 | 2023-02-28 23:59:59 |        55
             3 |               7 | 2023-03-01 00:00:00 | 2023-03-31 23:59:59 |        60
             5 |               7 | 2023-04-01 00:00:00 |                     |      1000
             4 |               8 | 2023-04-01 00:00:00 | 2023-04-30 23:59:59 |        65
(5 строк)


refill=# INSERT INTO fuel_prices(id_fuel_offered, start_time, end_time, per_liter)
refill-# values(7, '2023-4-15 00:00:00', null, 66);
INSERT 0 1
refill=# SELECT * FROM fuel_prices ORDER By id_fuel_offered limit 5;
 id_fuel_price | id_fuel_offered |     start_time      |      end_time       | per_liter
---------------+-----------------+---------------------+---------------------+-----------
             1 |               5 | 2023-01-01 00:00:00 | 2023-01-31 23:59:59 |        50
             2 |               6 | 2023-02-01 00:00:00 | 2023-02-28 23:59:59 |        55
             5 |               7 | 2023-04-01 00:00:00 | 2023-04-15 00:00:00 |      1000
             3 |               7 | 2023-03-01 00:00:00 | 2023-03-31 23:59:59 |        60
            26 |               7 | 2023-04-15 00:00:00 |                     |        66
(5 строк)
```

5. Триггер 5 – у цены можно менять только дату окончания и только на будущее значение

```
create or replace function fn_check_price_changes() returns
trigger as $$
begin
	if (new.start_time!=old.start_time or
		new.per_liter!=old.per_liter or
		(new.end_time is not null and new.end_time < NOW()))
	then
		return null;
	end if;
	return new;
end;
$$ language plpgsql;

create or replace trigger check_price_changes before update on fuel_prices
for each row execute procedure fn_check_price_changes();
```

```
refill=# select * from fuel_prices LIMIT 1;
 id_fuel_price | id_fuel_offered |     start_time      |      end_time       | per_liter
---------------+-----------------+---------------------+---------------------+-----------
             1 |               5 | 2023-01-01 00:00:00 | 2023-01-31 23:59:59 |        50
(1 строка)


refill=# UPDATE fuel_prices set end_time='2024-01-01 00:00:00' WHERE id_fuel_price=1;
UPDATE 0
refill=# UPDATE fuel_prices set end_time=NOW() WHERE id_fuel_price=1;
UPDATE 1

refill=# select * from fuel_prices WHERE id_fuel_price=1;
 id_fuel_price | id_fuel_offered |     start_time      |          end_time          | per_liter
---------------+-----------------+---------------------+----------------------------+-----------
             1 |               5 | 2023-01-01 00:00:00 | 2025-06-26 14:38:38.069003 |        50
(1 строка)
```

6. Триггер 6 – у производимого топлива компания должна быть компанией-поставщиком

```
create or replace function fn_check_produced_fuel_company() returns
trigger as $$
begin
	if (EXISTS(
		SELECT 1 FROM companies WHERE companies.id_company=new.id_company_factory
AND type_company=1
```

```
        )) then
                return new;
        end if;
        return null;
end;
$$ language plpgsql;

create or replace trigger check_produced_fuel_company before insert or update on produced_fuel
for each row execute procedure fn_check_produced_fuel_company();
```

```
refill=# select * from companies;
 id_company | type_company |           legal_address            |  company_title
------------+--------------+------------------------------------+------------------
         11 |            0 | Кудыкина гора                      | Рак
         12 |            1 | Эребор                             | Гимли и компания
          9 |            1 | Санкт-Петербург, пр. Невский, д. 15 | *** 5G
          8 |            0 | Москва, ул. Ленина, д. 1           | LOL
         10 |            0 | Дно                                | с водой
         13 |            1 | Колодец                            | Жги воду
(6 строк)


refill=# update produced_fuel set id_company_factory=9 WHERE id_produced_fuel=14;
UPDATE 1
refill=# update produced_fuel set id_company_factory=11 WHERE id_produced_fuel=14;
UPDATE 0
```

7. Триггер 7 – у заправок копания должна быть компанией-владельцем

```
create or replace function fn_check_filling_stations_company() returns
trigger as $$
begin
        if (EXISTS(
                SELECT 1 FROM companies WHERE companies.id_company=new.id_company_owner
AND type_company=0
        )) then
                return new;
        end if;
        return null;
end;
$$ language plpgsql;

create or replace trigger check_filling_stations_company before insert or update on filling_stations
for each row execute procedure fn_check_filling_stations_company();
```

```
refill=# SELECT * FROM companies;
 id_company | type_company |           legal_address            |  company_title
------------+--------------+------------------------------------+------------------
         11 |            0 | Кудыкина гора                      | Рак
         12 |            1 | Эребор                             | Гимли и компания
          9 |            1 | Санкт-Петербург, пр. Невский, д. 15 | *** 5G
          8 |            0 | Москва, ул. Ленина, д. 1           | LOL
         10 |            0 | Дно                                | с водой
         13 |            1 | Колодец                            | Жги воду
(6 строк)


refill=# update filling_stations set id_company_owner=9 WHERE id_filling_station=9;
UPDATE 0
refill=# update filling_stations set id_company_owner=11 WHERE id_filling_station=9;
UPDATE 1
```

8. Триггер 8 – у карточек клиентов копания должна быть владельцем компанией-владельцем

```
create or replace function fn_check_client_cards_company() returns
```

```
trigger as $$
begin
	if (EXISTS(
		SELECT 1 FROM companies WHERE companies.id_company=new.id_company AND type_company=0
	)) then
		return new;
	end if;
	return null;
end;
$$ language plpgsql;

create or replace trigger check_client_cards_company before insert or update on client_cards
for each row execute procedure fn_check_client_cards_company();
```

```
refill=# select * from companies;
 id_company | type_company |            legal_address            |  company_title
------------+--------------+-------------------------------------+-----------------
         11 |            0 | Кудыкина гора                       | Рак
         12 |            1 | Эребор                              | Гимли и компания
          9 |            1 | Санкт-Петербург, пр. Невский, д. 15 | *** 5G
          8 |            0 | Москва, ул. Ленина, д. 1            | LOL
         10 |            0 | Дно                                 | с водой
         13 |            1 | Колодец                             | Жги воду
(6 строк)


refill=# update client_cards set id_company=9 WHERE id_card=2;
UPDATE 0
refill=# update client_cards set id_company=11 WHERE id_card=2;
UPDATE 1
```

Выводы

Триггеры – это круто. Они позволяют автоматизировать то, без чего в базе данных была бы куча ошибок, например, сложные проверки, автоматическое обновление всех зависящих столбцов.

Все внешние ключи называть так же, как и первичный на который они ссылаются. Так не придётся реализовывать отдельный триггер для каждого такого нового названия ключа как в триггерах 6, 7, 8.