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 Смирновой Карины
Факультет прикладной информатики
Группа К3241
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 (min - 6 баллов, max - 10 баллов, доп. баллы - 3):

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:
   Вариант 2.1. 3 триггера - 3 балла (min). Допустимо использовать триггеры логирования из практического занятия по функциям и триггерам.
   Вариант 2.2. 7 оригинальных триггеров - 7 баллов (max).

ВЫПОЛНЕНИЕ

Создание 3-х процедур по варианту:

- Для снижения цен на книги, которые находятся на базе в количестве, превышающем 1000 штук.

```
CREATE OR REPLACE PROCEDURE fecrease_price_more_than_1000qbooks()
LANGUAGE SQL
AS $$
	UPDATE "mainS".edition
	SET retail_price = retail_price * 0.8
	WHERE remaining > 1000;
$$
```

- Для удаления заказов, имеющих статус “отменен”

```
CREATE OR REPLACE PROCEDURE delete_solded_tz()
LANGUAGE SQL
AS $$
	DELETE FROM "mainS".edition
	WHERE status = 'распродан';
$$;
DROP PROCEDURE delete_solded_tz();
```

- Для ввода нового заказа.

```
CREATE OR REPLACE PROCEDURE create_tz(tz_id integer, retail_price integer, quantity integer)
LANGUAGE SQL
AS $$
	INSERT INTO "mainS".edition (tz_id, edition_date, retail_price, status, quantity, remaining)
	VALUES (tz_id, CURRENT_DATE, retail_price, 'подготовка', quantity, quantity);
$$;
DROP PROCEDURE create_tz(tz_id integer, retail_price integer, quantity integer);
```

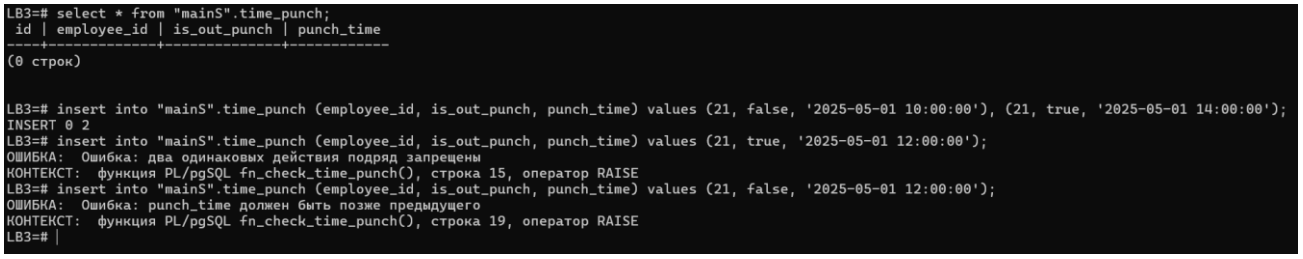
Создание триггеров:

1. Контроль начала и конца работы редакторов.

```
create or replace function fn_check_time_punch() returns trigger as $psql$
	declare
	last_is_out BOOLEAN;
	last_time TIMESTAMP;
  begin
	select is_out_punch, punch_time
```

```
        into last_is_out, last_time
        from "mainS".time_punch
        where employee_id = NEW.employee_id
        order by id desc
        limit 1;

        if found then
    if NEW.is_out_punch = last_is_out then
        raise exception 'Ошибка: два одинаковых действия подряд запрещены';
    end if;

    if NEW.punch_time <= last_time then
        raise exception 'Ошибка: punch_time должен быть позже предыдущего';
    end if;
end if;

return new;
end;
$psql$ language plpgsql;

create trigger check_time_punch before insert on "mainS".time_punch
for each row execute procedure fn_check_time_punch();
```

Проверка работы:



2. При добавлении "строка заказа" проверять, чтобы не было такого договора и тиража в таблице и выводить id «строки заказа», если есть.

```
create or replace function fn_duplicate_contract_in_order_line() returns trigger
AS $$
BEGIN
        IF EXISTS (
                SELECT 1 FROM "mainS".order_line
                WHERE contract_id = NEW.contract_id
        ) THEN
                RAISE EXCEPTION 'Договор % уже существует. ID строки заказа: %',
                NEW.contract_id, (SELECT order_line_id FROM "mainS".order_line WHERE
contract_id = NEW.contract_id);
        END IF;

        RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER trg_duplicate_contract_in_order_line
BEFORE INSERT ON "mainS".order_line
FOR EACH ROW EXECUTE FUNCTION fn_duplicate_contract_in_order_line();
```

Проверка работы:

```
LB3=# select * from "mainS".order_line;
 order_line_id | edition_id | contract_id | quantity
---------------+------------+-------------+----------
            11 |          6 |          10 |       50
            12 |          2 |          11 |       30
            13 |          3 |          12 |       40
            15 |          5 |          14 |       35
            14 |          4 |          13 |       13
(5 строк)


LB3=# insert into "mainS".order_line (edition_id, contract_id, quantity) values(2, 11, 50);
ОШИБКА:  Договор 11 уже существует. ID строки заказа: 12
КОНТЕКСТ:  функция PL/pgSQL fn_duplicate_contract_in_order_line(), строка 7, оператор RAISE
LB3=#
```

3. Логирование действий с таблицей order_line

```
create table "mainS".logs (
        text text,
        added timestamp without time zone
);
drop table "mainS".logs;

CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
DECLARE
   mstr varchar(30);
   astr varchar(100);
   retstr varchar(254);
BEGIN
   IF    TG_OP = 'INSERT' THEN
      astr = NEW.id;
      mstr := 'Add new user ';
      retstr := mstr||astr;
      INSERT INTO logs(text,added) values (retstr,NOW());
      RETURN NEW;
   ELSIF TG_OP = 'UPDATE' THEN
      astr = NEW.id;
      mstr := 'Update user ';
      retstr := mstr||astr;
      INSERT INTO logs(text,added) values (retstr,NOW());
      RETURN NEW;
   ELSIF TG_OP = 'DELETE' THEN
      astr = OLD.id;
      mstr := 'Remove user ';
      retstr := mstr || astr;
      INSERT INTO logs(text,added) values (retstr,NOW());
      RETURN OLD;
   END IF;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER t_logs_employee AFTER INSERT OR UPDATE OR DELETE ON
"mainS".employee FOR EACH ROW EXECUTE PROCEDURE add_to_log();
```

Проверка работы:

```
LB3=# insert into "mainS".employee (post_id, first_name, surname, last_name, email)
LB3-# values (2, 'Полина', 'Павловна', 'Сергеевна', 'pavlovna2@gmail.ru');
INSERT 0 1
LB3=# select * from "mainS".logs;
      text       |           added
-----------------+----------------------------
 Add new user 37 | 2025-05-06 12:48:00.518335
 Add new user 38 | 2025-05-06 15:28:26.944746
(2 строки)
```

4. Триггер, чтобы не допустить remaining < 0 при обновлении edition

```
create or replace function fn_remaning_more_than_zero() returns trigger
as $$
begin
	if new.remaining < 0
	then
		raise exception 'Вы пытаетесть установить остаток книги меньший 0';
	end if;
	return new;
end;
$$ language plpgsql;

create trigger tgr_remaning_more_than_zero before update on "mainS".edition for each row execute function fn_remaning_more_than_zero();
```

Проверка работы:

```
LB3=# select * from "mainS".edition;
 edition_id | tz_id | edition_date | retail_price |   status   | quantity | remaining
------------+-------+--------------+--------------+------------+----------+-----------
          2 |     1 | 2025-03-10   |          500 | подготовка |     1000 |       100
          4 |     3 | 2025-03-12   |          600 | в продаже  |      300 |        50
          5 |     4 | 2025-03-13   |          800 | распродан  |      400 |         0
          6 |     5 | 2025-03-14   |          900 | подготовка |      600 |       200
          7 |     6 | 2025-03-11   |          500 | подготовка |     1000 |       100
          8 |     7 | 2025-03-12   |          700 | в печати   |      500 |       300
          9 |     8 | 2025-03-13   |          600 | в продаже  |      300 |        50
         10 |     9 | 2025-03-14   |          800 | распродан  |      400 |         0
         11 |    10 | 2025-03-15   |          900 | подготовка |      600 |       200
         12 |    11 | 2025-04-07   |          700 | в продаже  |      200 |       187
          3 |     2 | 2025-03-11   |          700 | распродан  |      500 |         0
(11 строк)


LB3=# update "mainS".edition set remaining = -2 where edition_id = 5;
ОШИБКА:  Вы пытаетесть установить остаток книги меньший 0
КОНТЕКСТ:  функция PL/pgSQL fn_remaning_more_than_zero(), строка 5, оператор RAISE
LB3=#
```

5. Проверка на уникальность сочетания автора и книги в author_ship

```
create or replace function fn_unique_author_book_ship() returns trigger
as $$
begin
	if exists (
		select 1 from "mainS".authorship
		where new.book_id = book_id and new.author_id = author_id
	) then
		raise exception 'Такое соотношение автора и книги уже существует';
	end if;

	return new;
end;
$$ language plpgsql;
```

```
create trigger trg_unique_author_book_ship before insert or update on "mainS".authorship
for each row execute function fn_unique_author_book_ship();
```

Проверка работы:

```
LB3=# select * from "mainS".authorship;
 authorship_id | author_id | book_id | author_order
---------------+-----------+---------+--------------
             1 |         1 |       1 |            1
             2 |         2 |       2 |            1
             3 |         3 |       3 |            1
             4 |         4 |       4 |            1
             5 |         5 |       5 |            1
             6 |         6 |       6 |            1
             7 |         7 |       7 |            1
             8 |         8 |       8 |            1
             9 |         9 |       9 |            1
            10 |        10 |      10 |            1
            11 |        11 |      11 |            1
(11 строк)


LB3=# insert into "mainS".authorship (author_id, book_id, author_order) values (3, 3, 2);
ОШИБКА:  Такое соотношение автора и книги уже существует
КОНТЕКСТ:  функция PL/pgSQL fn_unique_author_book_ship(), строка 7, оператор RAISE
LB3=#
```

6. Расчёт суммы счета при добавлении записи в таблицу voice

```
create or replace function fn_calc_invoice_amount() returns trigger
as $$
declare
	total numeric := 0;
begin
	select sum (ol.quantity * ed.retail_price)
	into total
	from "mainS".order_line ol
	join "mainS".edition ed on ol.edition_id = ed.edition_id
	where ol.contract_id = new.contract_id;

	new.invoice_amount := coalesce(total, 0);
	return new;
end;
$$ language plpgsql;

CREATE TRIGGER tgr_calc_invoice_amount
BEFORE INSERT ON "mainS".invoice
FOR EACH ROW
EXECUTE FUNCTION fn_calc_invoice_amount();
```

Проверка работы:

```
LB3=# insert into "mainS".invoice (contract_id, payment_date, payment_status, invoice_date)
LB3-# values (11, '2025-05-06', true, '2025-05-02');
INSERT 0 1
LB3=# select * from "mainS".invoice where contract_id = 11;
 invoice_id | contract_id | payment_date | payment_status | invoice_date | invoice_amount
------------+-------------+--------------+----------------+--------------+----------------
         10 |          11 | 2023-07-15   | t              | 2023-07-10   |          40000
         18 |          11 | 2025-05-06   | t              | 2025-05-02   |          15000
(2 строки)
```

7. Автоматическое присвоение статуса распродан, если остаток 0

```
create or replace function fn_make_status_sold() returns trigger
as $$
begin
	if new.remaining = 0 and old.remaining <> 0
```

```
        then new.status = 'распродан';
        end if;

        return new;
end;
$$ language plpgsql;

create trigger trg_make_status_sold
before update on "mainS".edition for each row execute function fn_make_status_sold();
```

Проверка работы:

```
LB3=# select * from "mainS".edition;
 edition_id | tz_id | edition_date | retail_price |   status   | quantity | remaining
------------+-------+--------------+--------------+------------+----------+-----------
          2 |     1 | 2025-03-10   |          500 | подготовка |     1000 |       100
          4 |     3 | 2025-03-12   |          600 | в продаже  |      300 |        50
          5 |     4 | 2025-03-13   |          800 | распродан  |      400 |         0
          6 |     5 | 2025-03-14   |          900 | подготовка |      600 |       200
          7 |     6 | 2025-03-11   |          500 | подготовка |     1000 |       100
          8 |     7 | 2025-03-12   |          700 | в печати   |      500 |       300
          9 |     8 | 2025-03-13   |          600 | в продаже  |      300 |        50
         10 |     9 | 2025-03-14   |          800 | распродан  |      400 |         0
         11 |    10 | 2025-03-15   |          900 | подготовка |      600 |       200
         12 |    11 | 2025-04-07   |          700 | в продаже  |      200 |       187
          3 |     2 | 2025-03-11   |          700 | распродан  |      500 |         0
(11 строк)
```

```
LB3=# update "mainS".edition set remaining = 0
LB3-# where edition_id = 4;
UPDATE 1
LB3=# select * from "mainS".edition;
 edition_id | tz_id | edition_date | retail_price |   status   | quantity | remaining
------------+-------+--------------+--------------+------------+----------+-----------
          2 |     1 | 2025-03-10   |          500 | подготовка |     1000 |       100
          5 |     4 | 2025-03-13   |          800 | распродан  |      400 |         0
          6 |     5 | 2025-03-14   |          900 | подготовка |      600 |       200
          7 |     6 | 2025-03-11   |          500 | подготовка |     1000 |       100
          8 |     7 | 2025-03-12   |          700 | в печати   |      500 |       300
          9 |     8 | 2025-03-13   |          600 | в продаже  |      300 |        50
         10 |     9 | 2025-03-14   |          800 | распродан  |      400 |         0
         11 |    10 | 2025-03-15   |          900 | подготовка |      600 |       200
         12 |    11 | 2025-04-07   |          700 | в продаже  |      200 |       187
          3 |     2 | 2025-03-11   |          700 | распродан  |      500 |         0
          4 |     3 | 2025-03-12   |          600 | распродан  |      300 |         0
(11 строк)
```

8. Автоматическое изменение статуса работы по договору после оплаты счета

```
create or replace function fn_change_work_status() returns trigger
as $$
begin
        if (TG_OP = 'INSERT' and new.payment_status = true) or
        (TG_OP = 'UPDATE' and new.payment_status = true and old.payment_status = false)
        then
        update "mainS".contract set work_status = 'выполняется'
        where contract_id = new.contract_id;
        end if;

        return new;
end;
$$ language plpgsql;

create trigger trg_change_work_status after update or insert on "mainS".invoice
for each row execute function fn_change_work_status();
```

До:

```
LB3=# select * from "mainS".invoice where contract_id = 12;
 invoice_id | contract_id | payment_date | payment_status | invoice_date | invoice_amount
------------+-------------+--------------+----------------+--------------+----------------
         11 |          12 | 2023-10-30   | f              | 2023-10-20   |          45000
(1 строка)


LB3=# select * from "mainS".contract where contract_id = 12;
 contract_id | customer_id | employee_id | service_cost | signing_date | service_date |  work_act   | work_status |    delivery_address
-------------+-------------+-------------+--------------+--------------+--------------+-------------+-------------+------------------------
          12 |           3 |          23 |       120000 | 2023-05-05   | 2023-08-25   | act_003.pdf | в очереди   | Казань, ул. Баумана, 17
(1 строка)
```

После:

```
LB3=# update "mainS".invoice set payment_status = true where contract_id = 12;
UPDATE 1
LB3=# select * from "mainS".invoice where contract_id = 12;
 invoice_id | contract_id | payment_date | payment_status | invoice_date | invoice_amount
------------+-------------+--------------+----------------+--------------+----------------
         11 |          12 | 2023-10-30   | t              | 2023-10-20   |          45000
(1 строка)


LB3=# select * from "mainS".contract where contract_id = 12;
 contract_id | customer_id | employee_id | service_cost | signing_date | service_date |  work_act   | work_status |    delivery_address
-------------+-------------+-------------+--------------+--------------+--------------+-------------+-------------+------------------------
          12 |           3 |          23 |       120000 | 2023-05-05   | 2023-08-25   | act_003.pdf | выполняется | Казань, ул. Баумана, 17
(1 строка)
```

9. После изменения количества книг в order_line, должна пересчитываться стоимость заказа в счете

```
create or replace function fn_recalc_invoice_amount() returns trigger
as $$
declare
	total numeric = 0;
begin
	select sum (ol.quantity * ed.retail_price)
	into total
	from "mainS".order_line ol
	join "mainS".edition ed on ol.edition_id = ed.edition_id
	where ol.contract_id = new.contract_id;

	update "mainS".invoice
	set invoice_amount = coalesce(total, 0)
	where contract_id = new.contract_id;

	return null;
end;
$$ language plpgsql;

create trigger trg_recalc_invoice_amount
after insert or update or delete on "mainS".order_line
for each row execute function fn_recalc_invoice_amount();
```

До

```
LB3=# select * from "mainS".order_line where contract_id = 13;
 order_line_id | edition_id | contract_id | quantity
---------------+------------+-------------+----------
            14 |          4 |          13 |       20
(1 строка)


LB3=# select * from "mainS".invoice where contract_id = 13;
 invoice_id | contract_id | payment_date | payment_status | invoice_date | invoice_amount
------------+-------------+--------------+----------------+--------------+----------------
         12 |          13 | 2023-07-10   | t              | 2023-07-01   |          70000
(1 строка)
```

После

```
LB3=# update "mainS".order_line set quantity = 10 where contract_id = 13;
UPDATE 1
LB3=# select * from "mainS".invoice where contract_id = 13;
 invoice_id | contract_id | payment_date | payment_status | invoice_date | invoice_amount
------------+-------------+--------------+----------------+--------------+----------------
         12 |          13 | 2023-07-10   | t              | 2023-07-01   |           6000
(1 строка)


LB3=# select * from "mainS".order_line where contract_id = 13;
 order_line_id | edition_id | contract_id | quantity
---------------+------------+-------------+----------
            14 |          4 |          13 |       10
(1 строка)
```

10. Если меняется количество книг в заказе, то меняется предполагаемая дата выполнение заказа по договору

```
create or replace function fn_change_service_date()
returns trigger as $$
begin
        if new.quantity > old.quantity then
        update "mainS".contract
        set service_date = service_date + make_interval(weeks := new.quantity - old.quantity)
        where contract_id = new.contract_id;
        elsif new.quantity < old.quantity then
        update "mainS".contract
        set service_date = service_date + make_interval(weeks := new.quantity - old.quantity)

        where contract_id = new.contract_id;
        end if;

        return null;
end;
$$ language plpgsql;

create trigger trg_change_service_date
after update on "mainS".order_line
for each row execute function fn_change_service_date();
```

До

```
LB3=# select * from "mainS".order_line where contract_id = 13;
 order_line_id | edition_id | contract_id | quantity
---------------+------------+-------------+----------
            14 |          4 |          13 |       10
(1 строка)


LB3=# select * from "mainS".contract where contract_id = 13;
 contract_id | customer_id | employee_id | service_cost | signing_date | service_date |  work_act   | work_status |      delivery_address
-------------+-------------+-------------+--------------+--------------+--------------+-------------+-------------+---------------------------
          13 |           4 |          24 |        35000 | 2023-05-11   | 2023-08-04   | act_004.pdf | выполняется | Москва, ул.Космонавтов, 15
(1 строка)
```

После

```
LB3=# update "mainS".order_line set quantity = 13 where contract_id = 13;
UPDATE 1
LB3=# select * from "mainS".order_line where contract_id = 13;
 order_line_id | edition_id | contract_id | quantity
---------------+------------+-------------+----------
            14 |          4 |          13 |       13
(1 строка)


LB3=# select * from "mainS".contract where contract_id = 13;
 contract_id | customer_id | employee_id | service_cost | signing_date | service_date |  work_act   | work_status |      delivery_address
-------------+-------------+-------------+--------------+--------------+--------------+-------------+-------------+---------------------------
          13 |           4 |          24 |        35000 | 2023-05-11   | 2023-08-25   | act_004.pdf | выполняется | Москва, ул.Космонавтов, 15
(1 строка)
```

Вывод:

В данной лабораторной работе я овладела практическими навыками создания и использования процедур, функций и триггеров в базе данных PostgreSQL.