

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ  
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ  
ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ**

**Факультет инфокоммуникационных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №3  
ПО ДИСЦИПЛИНЕ «БАЗЫ ДАННЫХ»  
ПРОЦЕДУРЫ, ФУНКЦИИ, ТРИГГЕРЫ В POSTGRESQL**

Студент: Зайцева Анастасия Алексеевна

Группа: К3240

Вариант: 5

Преподаватель: Говорова Марина Михайловна

Санкт-Петербург

2022

## Цель работы

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

## Практическое задание

### Вариант 1

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)

Вариант 5. БД «Издательство компьютерной литературы»

**Задание 4.** Создать хранимые процедуры:

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

**Задание 5.** Создать необходимые триггеры. .

## Выполнение

### I. Хранимые процедуры

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

```
1. CREATE OR REPLACE PROCEDURE discount_books(min_quantity INTEGER = 1000, discount_sum
   INTEGER = 100)
2. AS $$
3. BEGIN
4.     UPDATE it.edition -- Понизить цену
5.     SET price = price - discount_sum
6.     WHERE it.edition.id IN (
7.         SELECT it.edition.id -- Издания книг с ценой больше или равной 199
8.         FROM it.edition
9.         WHERE it.edition.book_id IN (
```

```

10.         SELECT book_id -- Список книг, оставшихся в количестве больше 1000
11.         FROM (
12.             SELECT book_id, SUM(volume) AS used_sum -- Заказанное количество
13.             каждой книги
14.             FROM it.books_order INNER JOIN it.order_book ON it.books_order.id =
15.             it.order_book.order_id
16.             GROUP BY book_id
17.             ) used LEFT JOIN (
18.             SELECT it.book.id, SUM(volume) AS stock_sum -- Изданное количество
19.             каждой книги
20.             FROM it.book INNER JOIN it.edition ON it.book.id = it.edition.book_id
21.             GROUP BY it.book.id
22.             ) stock ON stock.id = book_id
23.         WHERE (stock_sum - used_sum) > min_quantity
24.         ) AND it.edition.price >= 199
25.     );
26. END;
27. $$ LANGUAGE plpgsql;

```

```

SQL Shell (psql)
publisher=# CREATE OR REPLACE PROCEDURE discount_books(min_quantity integer = 1000, discount_sum integer = 100)
publisher=# AS $$
publisher=# BEGIN
publisher=# UPDATE it.edition -- Понизить цену
publisher=# SET price = price - discount_sum
publisher=# WHERE it.edition.id IN (
publisher=# SELECT it.edition.id -- Издания книг с ценой больше или равной 199
publisher=# FROM it.edition
publisher=# WHERE it.edition.book_id IN (
publisher=# SELECT book_id -- Список книг, оставшихся в количестве больше 1000
publisher=# FROM (
publisher=# SELECT book_id, SUM(volume) AS used_sum -- Заказанное количество каждой книги
publisher=# FROM it.books_order INNER JOIN it.order_book ON it.books_order.id = it.order_book.order_id
publisher=# GROUP BY book_id
publisher=# ) used LEFT JOIN (
publisher=# SELECT it.book.id, SUM(volume) AS stock_sum -- Изданное количество каждой книги
publisher=# FROM it.book INNER JOIN it.edition ON it.book.id = it.edition.book_id
publisher=# GROUP BY it.book.id
publisher=# ) stock ON stock.id = book_id
publisher=# WHERE (stock_sum - used_sum) > min_quantity
publisher=# ) AND it.edition.price >= 199
publisher=# );
publisher=# END;
publisher=# $$ LANGUAGE plpgsql;
CREATE PROCEDURE
publisher=#

```

Рисунок 1. Создание процедуры discount\_books()

id	book_id	date	volume	price
2	2	2022-03-03	1000	9699
3	3	2022-03-03	10000	9699
4	1	2009-03-03	10000	9699
5	2	2022-03-03	200	9699
6	4	2022-03-02	1000	449
7	5	2021-03-02	100	1199
8	6	2022-01-24	1500	1699
9	7	2022-01-24	10000	699
10	8	2022-02-23	500	499
11	9	2018-05-02	500	2199
12	10	2022-03-29	2000	939
13	11	2022-04-01	3000	1099
14	6	2016-08-15	500	1399
15	5	2018-11-15	5000	699

Рисунок 2. Таблица edition до выполнения процедуры discount\_books()

```
SQL Shell (psql)

publisher=# call discount_books();
CALL
publisher=#
```

Рисунок 3. Выполнение процедуры *discount\_books()*

SQL Shell (psql)

```
publisher=# SELECT * FROM it.edition;
```

id	book_id	date	volume	price
2	2	2022-03-03	1000	9599
3	3	2022-03-03	10000	9599
4	1	2009-03-03	10000	9599
5	2	2022-03-03	200	9599
6	4	2022-03-02	1000	349
7	5	2021-03-02	100	1099
8	6	2022-01-24	1500	1599
9	7	2022-01-24	10000	599
10	8	2022-02-23	500	399
11	9	2018-05-02	500	2099
12	10	2022-03-29	2000	839
13	11	2022-04-01	3000	999
14	6	2016-08-15	500	1299
15	5	2018-11-15	5000	599

Рисунок 4. Таблица *edition* после выполнения процедуры *discount\_books()*

## 2. Для ввода новой книги

```
1. CREATE OR REPLACE PROCEDURE add_book(  
2.     title VARCHAR,  
3.     page_count INTEGER,  
4.     book_category_id INTEGER = NULL,  
5.     book_category_name VARCHAR = NULL,  
6.     has_illustrations BOOLEAN = NULL,  
7.     isbn bigint = NULL  
8. ) AS $$  
9. DECLARE  
10.     found_book_category_id INTEGER := NULL;  
11. BEGIN  
12.     IF book_category_id IS NOT NULL THEN  
13.         IF book_category_id IN (  
14.             SELECT id FROM it.book_category  
15.         ) THEN  
16.             INSERT INTO it.book(book_category_id, title, isbn, page_count,  
17. has_illustrations)  
18.             VALUES (book_category_id, title, isbn, page_count, has_illustrations);  
19.         ELSE  
20.             RAISE NOTICE 'Ошибка: указанный book_category_id не найден среди записей  
book_category!';  
20.             RETURN;
```

```

21.     END IF;
22.     ELIF book_category_name IS NOT NULL THEN
23.         SELECT id INTO found_book_category_id
24.         FROM it.book_category
25.         WHERE it.book_category.name LIKE book_category_name;
26.         IF found_book_category_id IS NOT NULL THEN
27.             INSERT INTO it.book(book_category_id, title, isbn, page_count,
has_illustrations)
28.             VALUES (found_book_category_id, title, isbn, page_count, has_illustrations);
29.         ELSE RAISE NOTICE 'Ошибка: указанный book_category_name не найден среди записей
book_category!';
30.         END IF;
31.         ELSE RAISE NOTICE 'Ошибка: вы должны указать book_category_id или
book_category_name!';
32.         END IF;
33. END;
34. $$ LANGUAGE plpgsql;

```

```

publisher=# CREATE OR REPLACE PROCEDURE add_book(
publisher(# title varchar,
publisher(# page_count integer,
publisher(# book_category_id integer = null,
publisher(# book_category_name varchar = null,
publisher(# has_illustrations boolean = null,
publisher(# isbn bigint = null
publisher(# ) AS $$
publisher$# DECLARE
publisher$# found_book_category_id integer := null;
publisher$# BEGIN
publisher$# IF book_category_id IS NOT NULL THEN
publisher$# IF book_category_id IN (
publisher$# SELECT id FROM it.book_category
publisher$# ) THEN
publisher$# INSERT INTO it.book(book_category_id, title, isbn, page_count, has_illustrations)
publisher$# VALUES (book_category_id, title, isbn, page_count, has_illustrations);
publisher$# ELSE
publisher$# RAISE NOTICE 'Ошибка: указанный book_category_id не найден среди записей book_category!';
publisher$# RETURN;
publisher$# END IF;
publisher$# ELIF book_category_name IS NOT NULL THEN
publisher$# SELECT id INTO found_book_category_id
publisher$# FROM it.book_category
publisher$# WHERE it.book_category.name LIKE book_category_name;
publisher$# IF found_book_category_id IS NOT NULL THEN
publisher$# INSERT INTO it.book(book_category_id, title, isbn, page_count, has_illustrations)
publisher$# VALUES (found_book_category_id, title, isbn, page_count, has_illustrations);
publisher$# ELSE RAISE NOTICE 'Ошибка: указанный book_category_name не найден среди записей book_category!';
publisher$# END IF;
publisher$# ELSE RAISE NOTICE 'Ошибка: вы должны указать book_category_id или book_category_name!';
publisher$# END IF;
publisher$# END;
publisher$# $$ LANGUAGE plpgsql;
CREATE PROCEDURE
publisher=#

```

Рисунок 5. Создание процедуры add\_book

id	book_category_id	title	isbn	page_count	has_illustrations
1	2	Введение в реляционные базы данных	9785941577705	464	t
2	1	Код. Тайный язык информатики	9780735605053	393	t
3	3	Полный справочник по C++, 4-е издание	9785845904898	800	f
4	2	Изучаем PostgreSQL 10	9785970606438	400	t
5	2	Основы технологий баз данных. Учебное пособие	9785940748205	240	f
6	2	Oracle Database 11g. Программирование на языке PL/SQL	9785855823110	880	t
7	2	Оптимизация запросов PostgreSQL	9785970609637	278	t
8	2	Изучаем SQL. Генерация, выборка и обработка данных	9785907365544	400	t
9	2	Инновации SQL SERVER 2019	9785970605950	408	t
10	2	Данные: хранение и обработка. Учебник	9785160156637	205	f
11	2	Технологии проектирования баз данных	9785970607374	498	t
12	3	Изучаем Python. 3-е издание	9785932861387	830	t
13	3	Программирование на Java для начинающих	9785699894758	706	t
14	3	Изучай Haskell во имя добра!	9785940747499	491	t

(14 строк)

Рисунок 6. Таблица book до выполнения процедуры add\_book

1. CALL add\_book('Основы Python. Научитесь думать как программист', 306);
2. CALL add\_book('Основы Python. Научитесь думать как программист', 306, 4);
3. CALL add\_book('Основы Python. Научитесь думать как программист', 306, NULL, 'ЯП');
4. CALL add\_book('Основы Python. Научитесь думать как программист', 306, 3);
5. CALL add\_book('Программирование на PHP в примерах и задачах', 354, NULL, 'Языки программирования', FALSE, 9785041578664);

```

SQL Shell (psql)
publisher=# CALL add_book('Основы Python. Научитесь думать как программист', 306);
ЗАМЕЧАНИЕ: Ошибка: вы должны указать book_category_id или book_category_name!
CALL
publisher=# CALL add_book('Основы Python. Научитесь думать как программист', 306, 4);
ЗАМЕЧАНИЕ: Ошибка: указанный book_category_id не найден среди записей book_category!
CALL
publisher=# CALL add_book('Основы Python. Научитесь думать как программист', 306, null, 'ЯП');
ЗАМЕЧАНИЕ: Ошибка: указанный book_category_name не найден среди записей book_category!
CALL
publisher=# CALL add_book('Основы Python. Научитесь думать как программист', 306, 3);
CALL
publisher=# CALL add_book('Программирование на PHP в примерах и задачах', 354, null, 'Языки програ
ммирования', false, 9785041578664);
CALL
publisher=#

```

Рисунок 7. Выполнение процедуры add\_book

```

SQL Shell (psql)
publisher=# SELECT * FROM it.books;

```

id	book_category_id	title	isbn	page_count	has_illustrations
1	2	Введение в реляционные базы данных	9785941577705	464	t
2	1	Код. Тайный язык информатики	9780735605053	393	t
3	3	Полный справочник по C++, 4-е издание	9785845904898	800	f
4	2	Изучаем PostgreSQL 10	9785970606438	400	t
5	2	Основы технологий баз данных. Учебное пособие	9785940748205	240	f
6	2	Oracle Database 11g. Программирование на языке PL/SQL	9785855823110	880	t
7	2	Оптимизация запросов PostgreSQL	9785970609637	278	t
8	2	Изучаем SQL. Генерация, выборка и обработка данных	9785907365544	400	t
9	2	ИновацииSQL SERVER 2019	9785970605950	408	t
10	2	Данные: хранение и обработка. Учебник	9785160156637	205	f
11	2	Технологии проектирования баз данных	9785970607374	498	t
12	2	Изучаем Python. 3-е издание	9785932861387	830	t
13	3	Программирование на Java для начинающих	9785699894758	706	t
14	3	Изучай Haskell во имя добра!	9785940747499	491	t
15	3	Основы Python. Научитесь думать как программист		306	
16	3	Программирование на PHP в примерах и задачах	9785041578664	354	f

(16 строк)

Рисунок 8. Таблица book после выполнения процедуры add\_book

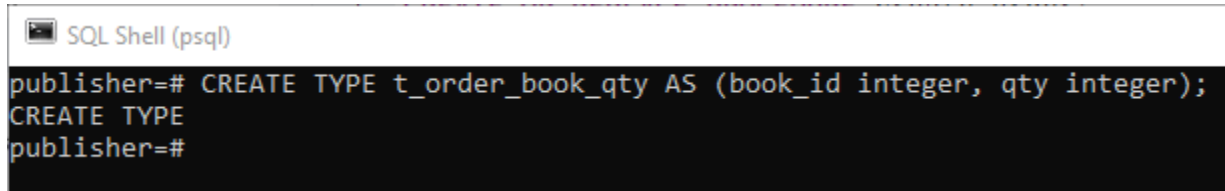
### 3. Для ввода нового заказа.

1. CREATE TYPE t\_order\_book\_qty AS (book\_id INTEGER, qty INTEGER);
- 2.
3. CREATE OR REPLACE PROCEDURE create\_order(
4.     customer\_id INTEGER,
5.     order\_manager\_id INTEGER,
6.     date\_until DATE,
7.     books t\_order\_book\_qty[]
8. ) AS \$\$
9. DECLARE
10.     created\_order\_id INTEGER;
11.     book\_qty t\_order\_book\_qty;
12. BEGIN
13.     INSERT INTO it.books\_order(customer\_id, order\_manager\_id, date\_created, date\_until,
14.     status)
15.     VALUES (customer\_id, order\_manager\_id, CURRENT\_DATE, date\_until, 'В обработке');

```

15.
16.     SELECT id INTO created_order_id
17.     FROM it.books_order
18.     ORDER BY id DESC LIMIT 1;
19.
20.     FOREACH book_qty IN ARRAY books
21.     LOOP
22.         INSERT INTO it.order_book(order_id, book_id, volume)
23.         VALUES (created_order_id, book_qty.book_id, book_qty.qty);
24.     END LOOP;
25. END;
26. $$ LANGUAGE plpgsql;

```

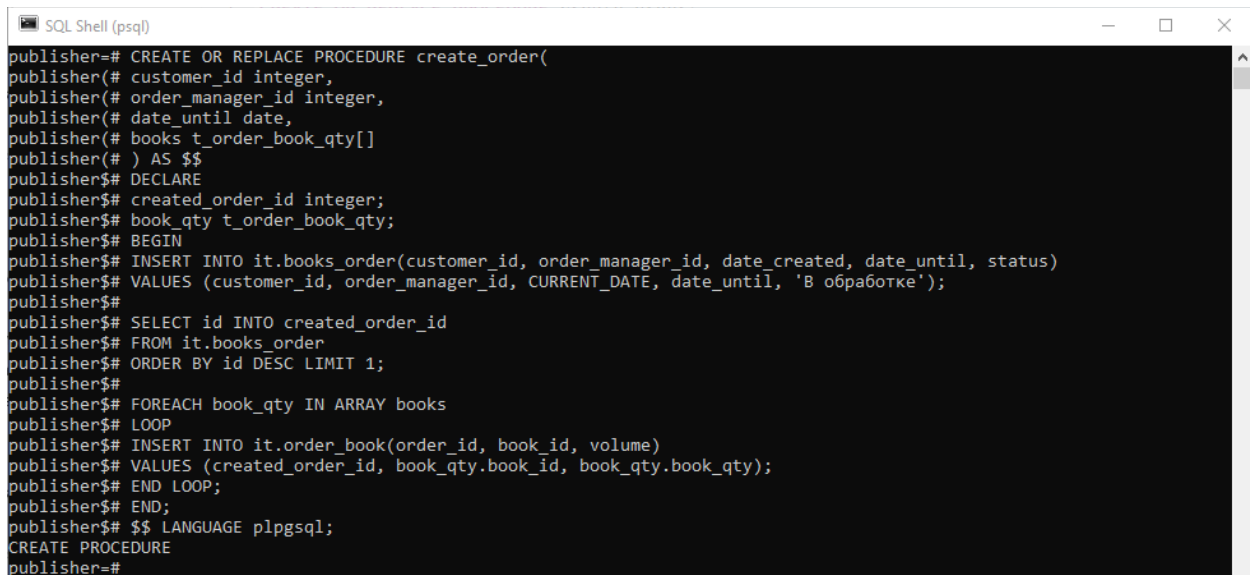


```

SQL Shell (psql)
publisher=# CREATE TYPE t_order_book_qty AS (book_id integer, qty integer);
CREATE TYPE
publisher=#

```

*Рисунок 9. Создание типа t\_order\_book\_qty*



```

SQL Shell (psql)
publisher=# CREATE OR REPLACE PROCEDURE create_order(
publisher(# customer_id integer,
publisher(# order_manager_id integer,
publisher(# date_until date,
publisher(# books t_order_book_qty[]
publisher(# ) AS $$
publisher$# DECLARE
publisher$# created_order_id integer;
publisher$# book_qty t_order_book_qty;
publisher$# BEGIN
publisher$# INSERT INTO it.books_order(customer_id, order_manager_id, date_created, date_until, status)
publisher$# VALUES (customer_id, order_manager_id, CURRENT_DATE, date_until, 'В обработке');
publisher$#
publisher$# SELECT id INTO created_order_id
publisher$# FROM it.books_order
publisher$# ORDER BY id DESC LIMIT 1;
publisher$#
publisher$# FOREACH book_qty IN ARRAY books
publisher$# LOOP
publisher$# INSERT INTO it.order_book(order_id, book_id, volume)
publisher$# VALUES (created_order_id, book_qty.book_id, book_qty.book_qty);
publisher$# END LOOP;
publisher$# END;
publisher$# $$ LANGUAGE plpgsql;
CREATE PROCEDURE
publisher=#

```

*Рисунок 10. Создание процедуры create\_order*

```
publisher=# SELECT * FROM it.books_order ORDER BY id DESC LIMIT 10;
```

id	customer_id	order_manager_id	date_created	date_until	status
200	5	6	2022-03-12	2022-04-04	В обработке
199	7	3	2022-03-11	2022-04-05	В обработке
198	1	3	2022-03-30	2022-04-01	Отправлен
197	5	1	2022-03-27	2022-04-07	В обработке
196	2	1	2022-03-24	2022-04-05	Ожидает оплаты
195	2	3	2022-03-05	2022-04-01	Отправлен
194	3	2	2022-03-15	2022-04-03	Отправлен
193	10	4	2022-04-02	2022-04-03	Отменен
192	6	1	2022-03-14	2022-04-05	В обработке
191	10	3	2022-03-04	2022-04-08	В обработке

(10 строк)

Рисунок 11. Таблица books\_order до выполнения процедуры create\_order

1. CALL create\_order(5, 3, '2022-05-30', ARRAY[ROW(10,50),ROW(11,100)]::t\_order\_book\_qty[]);

```
SQL Shell (psql)
publisher=# CALL create_order(5, 3, '2022-05-30', array[row(10,50),row(11,100)]::t_order_book_qty[]);
CALL
publisher=#
```

Рисунок 12. Выполнение процедуры create\_order

```
SQL Shell (psql)
publisher=# SELECT * FROM it.books_order ORDER BY id DESC LIMIT 3;
```

id	customer_id	order_manager_id	date_created	date_until	status
202	5	3	2022-05-03	2022-05-30	В обработке
200	5	6	2022-03-12	2022-04-04	В обработке
199	7	3	2022-03-11	2022-04-05	В обработке

(3 строки)

```
publisher=# SELECT * FROM it.order_book WHERE order_id = 202;
```

order_id	book_id	volume
202	10	50
202	11	100

(2 строки)

Рисунок 11. Таблицы books\_order и order\_book после выполнения процедуры create\_order

## II. Триггеры

Была создана таблица logs.



it
logs
id integer
added time without time zone
table_name text
text text

Рисунок 12. Таблица logs

```

1. CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
2. DECLARE
3.     mstr VARCHAR(30);
4.     astr VARCHAR(254);
5.     retstr VARCHAR(254);
6. BEGIN
7.     IF TG_OP = 'INSERT' THEN
8.         astr = NEW;
9.         mstr := 'Добавлено ';
10.        retstr := mstr || astr;
11.        INSERT INTO it.logs(text, added, table_name) VALUES (retstr, NOW(),
12.            TG_TABLE_NAME);
13.        RETURN NEW;
14.    ELSIF TG_OP = 'UPDATE' THEN
15.        astr = NEW;
16.        mstr := 'Изменено ';
17.        retstr := mstr || astr;
18.        INSERT INTO it.logs(text, added, table_name) VALUES (retstr, NOW(),
19.            TG_TABLE_NAME);
20.        RETURN NEW;
21.    ELSIF TG_OP = 'DELETE' THEN
22.        astr = OLD;
23.        mstr := 'Удалено ';
24.        retstr := mstr || astr;
25.        INSERT INTO it.logs(text, added, table_name) VALUES (retstr, NOW(),
26.            TG_TABLE_NAME);
27.        RETURN OLD;
28.    END IF;
29. END;
30. $$ LANGUAGE plpgsql;

```

```

SQL Shell (psql)
publisher=# CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
publisher$# DECLARE
publisher$#     mstr varchar(30);
publisher$#     astr varchar(254);
publisher$#     retstr varchar(254);
publisher$# BEGIN
publisher$#     IF TG_OP = 'INSERT' THEN
publisher$#         astr = NEW;
publisher$#         mstr := 'Добавлено ';
publisher$#         retstr := mstr || astr;
publisher$#         INSERT INTO logs(text, added, table_name) VALUES (retstr, NOW(), TG_TABLE_NAME);
publisher$#         RETURN NEW;
publisher$#     ELSIF TG_OP = 'UPDATE' THEN
publisher$#         astr = NEW;
publisher$#         mstr := 'Изменено ';
publisher$#         retstr := mstr || astr;
publisher$#         INSERT INTO logs(text, added, table_name) VALUES (retstr, NOW(), TG_TABLE_NAME);
publisher$#         RETURN NEW;
publisher$#     ELSIF TG_OP = 'DELETE' THEN
publisher$#         astr = OLD;
publisher$#         mstr := 'Удалено ';
publisher$#         retstr := mstr || astr;
publisher$#         INSERT INTO logs(text, added, table_name) VALUES (retstr, NOW(), TG_TABLE_NAME);
publisher$#         RETURN OLD;
publisher$#     END IF;
publisher$# END;
publisher$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
publisher=#

```

Рисунок 13. Создание функции add\_to\_log

1. CREATE TRIGGER book\_tr AFTER INSERT OR UPDATE OR DELETE ON it.book FOR EACH ROW EXECUTE PROCEDURE add\_to\_log();
2. CREATE TRIGGER books\_order\_tr AFTER INSERT OR UPDATE OR DELETE ON it.books\_order FOR EACH ROW EXECUTE PROCEDURE add\_to\_log();
3. CREATE TRIGGER order\_book\_tr AFTER INSERT OR UPDATE OR DELETE ON it.order\_book FOR EACH ROW EXECUTE PROCEDURE add\_to\_log();

```

SQL Shell (psql)
publisher=# CREATE TRIGGER book_tr AFTER INSERT OR UPDATE OR DELETE ON it.book FOR EACH ROW EXECUTE PROCEDURE add_to_log();
CREATE TRIGGER
publisher=# CREATE TRIGGER books_order_tr AFTER INSERT OR UPDATE OR DELETE ON it.books_order FOR EACH ROW EXECUTE PROCEDURE add_to_log();
CREATE TRIGGER
publisher=# CREATE TRIGGER order_book_tr AFTER INSERT OR UPDATE OR DELETE ON it.order_book FOR EACH ROW EXECUTE PROCEDURE add_to_log();
CREATE TRIGGER
publisher=#

```

Рисунок 14. Создание триггеров

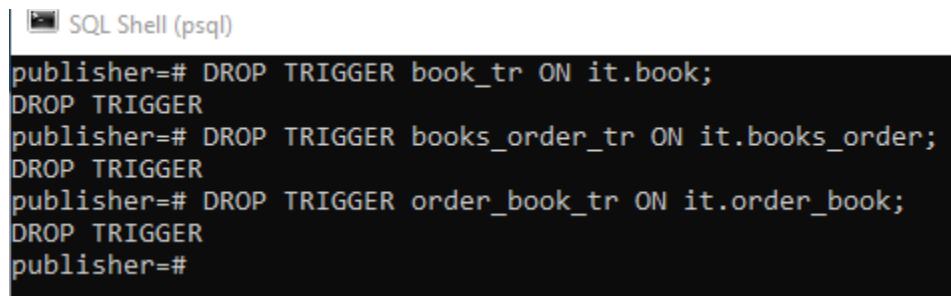
1. CALL add\_book('Проектирование и реализация систем управления базами данных', 466, 2);
2. CALL create\_order(1, 3, '2022-05-30', ARRAY[ROW(10,50),ROW(11,100)]::t\_order\_book\_qty[]);

```

SQL Shell (psql)
publisher=# CALL add_book('Проектирование и реализация систем управления базами данных', 466, 2);
CALL
publisher=# CALL create_order(1, 3, '2022-05-30', array[row(10,50),row(11,100)]::t_order_book_qty[]);
CALL
publisher=# SELECT * FROM it.logs;
 id |      added      | table_name | text
-----+-----+-----+-----
  1 | 23:29:59.158047 | book       | Добавлено (18,2,"Проектирование и реализация систем управления базами данных",,466,)
  2 | 23:30:00.972187 | books_order | Добавлено (204,1,3,2022-05-03,2022-05-30,"В обработке")
  3 | 23:30:00.972187 | order_book | Добавлено (204,10,50)
  4 | 23:30:00.972187 | order_book | Добавлено (204,11,100)
(4 строки)

```

Рисунок 15. Пример работы триггеров



```
SQL Shell (psql)
publisher=# DROP TRIGGER book_tr ON it.book;
DROP TRIGGER
publisher=# DROP TRIGGER books_order_tr ON it.books_order;
DROP TRIGGER
publisher=# DROP TRIGGER order_book_tr ON it.order_book;
DROP TRIGGER
publisher=#
```

*Рисунок 16. Удаление триггеров*

## **Выводы**

В ходе данной лабораторной работы я освоила основы процедурного языка PL/pgSQL, научилась создавать и применять функции и процедуры, а также использовать их при создании триггеров.