Министерство науки и высшего образования Российской Федерации
федеральное государственное автономное образовательное учреждение высшего образования
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчет

по лабораторной работе №3 «процедуры, функции, триггеры в PostgreSQL»

по дисциплине «**Проектирование и реализация баз данных**»

Вариант 5

Автор: Таякин Даниил

Факультет: ИКТ

Группа: К32392

Преподаватель: Говорова М. М.

Дата: 30.05.2023

Санкт-Петербург 2023

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

Оборудование: компьютерный класс.

Программное обеспечение: СУБД PostgreSQL, SQL Shell (psql).

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.

Выполнение:

1. Название БД – “publishing_office”.
2. Схема логической модели базы данных, сгенерированная в Generate ERD.

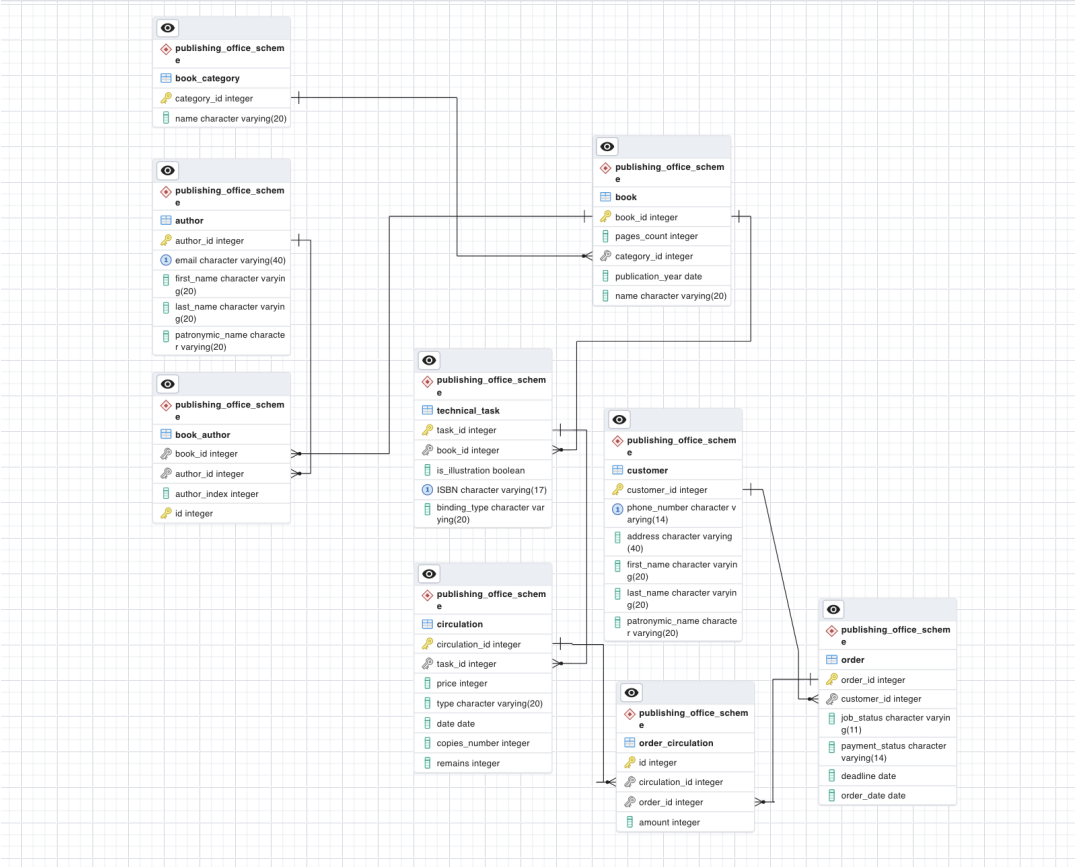
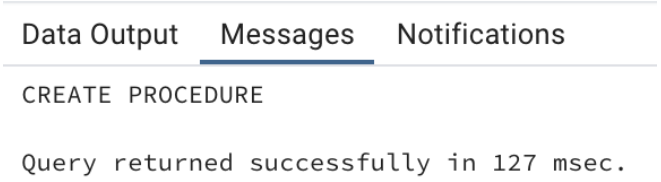


Рисунок 1 – Схема логической модели базы данных (ERD)

3. Создаем процедуры согласно индивидуальному заданию:

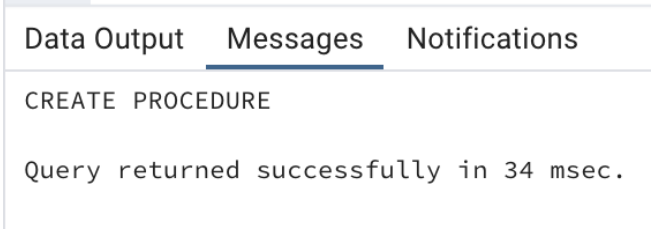
3.1. Для снижения цен на книги, которые находятся на базе в количестве, превышающем 1000 штук.

```
CREATE OR REPLACE PROCEDURE update_book_prices()
LANGUAGE plpgsql
AS $$
BEGIN
    UPDATE publishing_office_scheme.circulation
    SET price = price * 0.9 -- Уменьшаем цену на 10%
    WHERE remains > 1000;
END;
$$;
```



3.2. Для ввода новой книги.

```
CREATE OR REPLACE PROCEDURE insert_book(
    IN pages_count INTEGER,
    IN category_id INTEGER,
    IN publication_year DATE,
    IN book_title VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO publishing_office_scheme.book (pages_count, category_id, publication_year, name)
    VALUES (pages_count, category_id, publication_year, book_title);
END;
$$;
```



3.3. Для ввода нового заказа.

```
CREATE OR REPLACE PROCEDURE insert_order(
    IN customer_id INTEGER,
    IN deadline DATE
)
LANGUAGE plpgsql
AS $$
BEGIN
    INSERT INTO publishing_office_scheme.order (customer_id, job_status, payment_status, deadline, order_date)
    VALUES (customer_id, 'в обработке', 'ожидает оплаты', deadline, NOW());
```

```
END;
$$;
```

Data Output | Messages | Notifications

```
CREATE PROCEDURE

Query returned successfully in 38 msec.
```

4. Создаем триггер для логирования событий вставки, удаления, редактирования данных в базе данных PostgreSQL (согласно индивидуальному заданию, часть 5).

Query | Query History

```
1  CREATE TABLE IF NOT EXISTS publishing_office_scheme.log_table (
2      id SERIAL PRIMARY KEY,
3      event_type VARCHAR(20) NOT NULL,
4      table_name VARCHAR(60) NOT NULL,
5      timestamp TIMESTAMPTZ DEFAULT NOW()
6  );
7
8  CREATE OR REPLACE FUNCTION log_changes()
9  RETURNS TRIGGER
10 LANGUAGE plpgsql
11 AS $$
12 BEGIN
13     IF TG_OP = 'INSERT' THEN
14         INSERT INTO publishing_office_scheme.log_table (event_type, table_name, timestamp)
15         VALUES ('INSERT', TG_TABLE_NAME, now());
16     ELSIF TG_OP = 'UPDATE' THEN
17         INSERT INTO publishing_office_scheme.log_table (event_type, table_name, timestamp)
18         VALUES ('UPDATE', TG_TABLE_NAME, now());
19     ELSIF TG_OP = 'DELETE' THEN
20         INSERT INTO publishing_office_scheme.log_table (event_type, table_name, timestamp)
21         VALUES ('DELETE', TG_TABLE_NAME, now());
22     END IF;
23
24     RETURN NEW;
25 END;
26 $$;
27
28 CREATE TRIGGER log_changes_trigger
29 AFTER INSERT OR UPDATE OR DELETE
30 ON publishing_office_scheme.book
31 FOR EACH ROW
32 EXECUTE FUNCTION log_changes();
```

Проверяем процедуру *insert_book* и триггер на соответствующую вставку:

Query | Query History

```
1  CALL insert_book(239, 2, '2023-06-07', 'Book Ed');
2
3  SELECT * FROM publishing_office_scheme.log_table;
```

Data Output | Messages | Notifications

	id [PK] integer	event_type character varying (20)	table_name character varying (60)	timestamp timestamp with time zone
1	1	INSERT	book	2023-05-31 16:25:46.863891+03
2	2	INSERT	book	2023-05-31 16:26:32.010938+03

Проверяем *update_book_prices*:

	circulation_id [PK] integer	task_id integer	price integer	type character varying (20)	date date	copies_number integer	remains integer
1	1	2	1990	книга	2023-05-01	[null]	[null]
2	4	7	4491	книга	2023-05-02	10000	9000
3	5	6	3591	книга	2023-05-10	10000	10000
4	6	5	233	книга	2023-02-12	100000	100000
5	7	8	449	книга	2023-03-20	10000	10000
6	8	3	134	книга	2022-03-04	200000	200000
7	9	10	791	книга	2021-05-02	10000	10000

Query Query History

```
1  CALL update_book_prices();
2
3  SELECT * FROM publishing_office_scheme.circulation;
```

Data Output Messages Notifications

	circulation_id [PK] integer	task_id integer	price integer	type character varying (20)	date date	copies_number integer	remains integer
1	1	2	1990	книга	2023-05-01	[null]	[null]
2	4	7	4042	книга	2023-05-02	10000	9000
3	5	6	3232	книга	2023-05-10	10000	10000
4	6	5	210	книга	2023-02-12	100000	100000
5	7	8	404	книга	2023-03-20	10000	10000
6	8	3	121	книга	2022-03-04	200000	200000
7	9	10	712	книга	2021-05-02	10000	10000

Вывод: В результате выполнения индивидуального задания были созданы процедуры/функции и триггеры для логирования событий вставки, удаления и редактирования данных в базе данных PostgreSQL.