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: Логачев Даниил
Факультет: ИКТ
Группа: К32402
Преподаватель: Говорова М. М.
Дата: 25.09.2023

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.

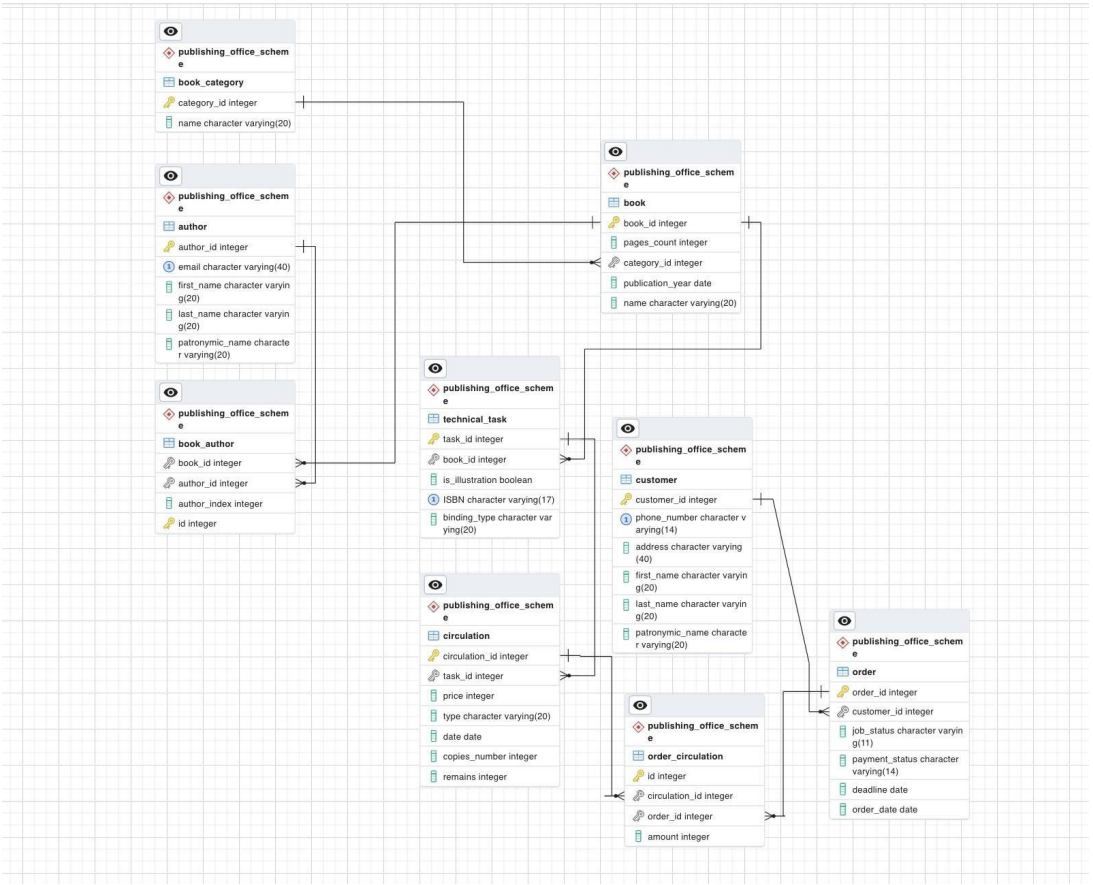
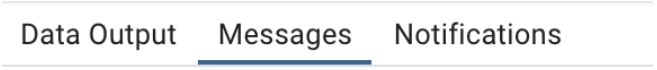


Рисунок 1 – Схема логической модели базы данных (ERD)

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 $$
BEGIN
   UPDATE publishing_office_scheme.circulation
   SET price = price * 0.9 -- Уменьшаем цену на
   10%
   WHERE remains > 1000;
END;
$$;
```

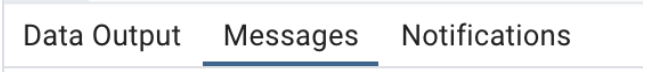



```
CREATE PROCEDURE

Query returned successfully in 127 msec.
```

3.2. Для ввода новой книги.

```
CREATE OR REPLACE PROCEDURE
   insert_book( IN pages_c INTEGER,
   IN cat_id
   INTEGER, IN
   pub_year DATE,
   IN book_title VARCHAR
)
LANGUAGE plpgsql
AS $$
BEGIN
   IF NOT EXISTS (
      SELECT 1 FROM
      publishing_office_scheme.book WHERE
      category_id = cat_id
      AND publication_year = pub_year
      AND name = book_title
   )
   THE
   N
      INSERT INTO publishing_office_scheme.book (pages_count,
category_id, publication_year, name)
      VALUES (pages_c, cat_id, pub_year,
   book_title); END IF;
END;
$$;
```




```
CREATE PROCEDURE

Query returned successfully in 34 msec.
```

3.3. Для ввода нового заказа.

```
CREATE OR REPLACE PROCEDURE
   insert_order( IN customer_id INTEGER,
   IN deadline DATE
)
LANGUAGE plpgsql
AS $$
BEGIN
   INSERT INTO publishing_office_scheme.order (customer_id,
job_status, payment_status, deadline, order_date)
   VALUES (customer_id, 'in process', 'waiting payment', deadline, NOW());
END;
$$;
```

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

```
CREATE TABLE IF NOT EXISTS publishing_office_scheme.log_table(
  id SERIAL PRIMARY KEY,
  event_type VARCHAR (20) NOT NULL,
  table_name VARCHAR (60) NOT NULL,
  timestamp TIMESTAMPTZ DEFAULT NOW(),
  old_value VARCHAR (400),
  new_value VARCHAR (400)
);

CREATE OR REPLACE FUNCTION log_changes()
RETURNS TRIGGER
LANGUAGE plpgsql
AS $$
BEGIN
  IF TG_OP = 'INSERT' THEN
    INSERT INTO publishing_office_scheme.log_table (event_type, table_name, old_value, new_value, timestamp)
    VALUES ('INSERT', TG_TABLE_NAME, NULL::VARCHAR, row_to_json(NEW) ::VARCHAR, now());
  ELSIF TG_OP = 'UPDATE' THEN
    INSERT INTO publishing_office_scheme.log_table (event_type, table_name, old_value, new_value, timestamp)
    VALUES ('UPDATE', TG_TABLE_NAME, row_to_json(OLD) ::VARCHAR, row_to_json(NEW):: VARCHAR, now());
  ELSIF TG_OP = 'DELETE' THEN
    INSERT INTO publishing_office_schene.log_table (event_type, table_name, old_value, new_value, timestamp)
    VALUES ('DELETE', TG_TABLE_NAME, row_to_json(OLD) ::VARCHAR, NULL::VARCHAR, now());
END IF;

RETURN NEW;
END;
$$;
CREATE TRIGGER log_changes_trigger AFTER INSERT OR UPDATE OR DELETE ON publishing_office_scheme.book FOR EACH ROW
EXECUTE FUNCTION log_changes();
```

Проверяем процедуру *insert_book* и триггер на соответствующую вставку:

```
1  CALL insert_book(245, 3, '2022-07-27', 'Mongo');
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	2023-09-27 02:24:40.417581+03
2	2	INSERT	book	2023-09-27 02:25:01.55538+03

Проверяем *update_book_prices*:

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

Data Output | Messages | Notifications

	circulation_id [PK] integer	task_id integer	book_id integer	ISBN character varying (17)	price integer	type character varying (20)	date date	remains integer
1	2	3	13	978-3-16-148410-0	699	journal	2023-09-02	1000
2	3	2	12	978-0-14-312656-0	999	book	2023-08-02	150
3	1	1	11	978-0-06-269601-9	1791	book	2023-09-01	2000
4	4	4	14	978-0-241-29752-8	809	book	2023-08-02	3000

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