Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО» Факультет инфокоммуникационных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 5

по теме: процедуры, функции, триггеры в PostgreSQL

Специальность:

09.03.03 Мобильные и сетевые технологии

Проверил:

Говорова М.М. _____

Выполнил:

студент группы K3240 Ковалев В.М.

Санкт-Петербург 2022

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).

Вариант 1

Практическое задание:

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.

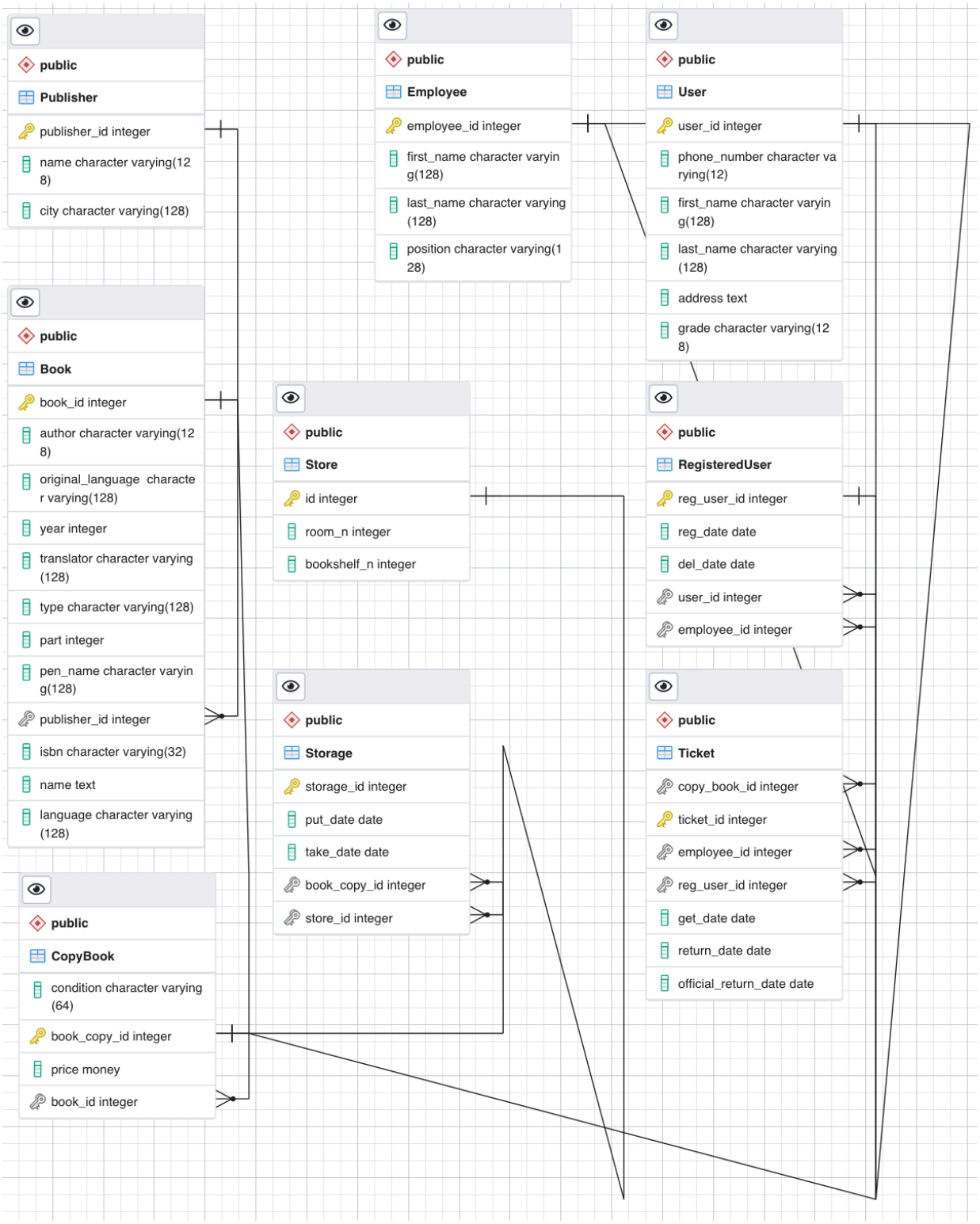
Вариант 3

Библиотека

Описание предметной области: Каждая книга может храниться в нескольких экземплярах. Для каждого экземпляра известно место его хранения (комната, стеллаж, полка). Читателю не может быть выдано более 3-х книг одновременно. Книги выдаются читателям на срок не более 10 дней. БД должна содержать следующий минимальный набор сведений: · Автор (фамилия и имя (инициалы) или псевдоним автора издания). · Название (заглавие) издания. · Номер тома (части, книги, выпуска). · Составитель (фамилия и имена (инициалы) каждого из составителей издания). · Язык, с которого выполнен перевод издания. · Вид издания (сборник, справочник, монография ...). · Область знания. · Переводчик (фамилия и инициалы переводчика). · Место издания (город). · Издательство (название издательства). · Год выпуска издания. · Библиотечный шифр (например, ББК 32.973). · Номер (инвентарный номер) экземпляра. · Номер комнаты (помещения для хранения экземпляров). · Номер стеллажа в комнате. · Номер полки на стеллаже. · Цена конкретного экземпляра. · Дата изъятия экземпляра с установленного места. · Номер читательского билета (формуляра). · Фамилия читателя. · Имя читателя. · Отчество читателя. · Адрес читателя. Телефон читателя.

Дополнить исходные данные информацией о читательском абонементе (выдаче книг).

Рисунок 1. ER-диаграмма



Задание 1.

1. *Для проверки наличия экземпляров заданной книги в библиотеке (процедура должна возвращать количество экземпляров книги).*

```
Library=# create or replace function count_of_copybooks_by_book_id(id integer)
    returns table
            (
                total bigint
            )
    language plpgsql
as
$$
begin
    return query (
      SELECT COUNT(*) FROM "CopyBook" AS t1 RIGHT JOIN "Storage" AS t2 ON t1.book_copy_id=t2.book_copy_id
WHERE t1.book_id=id AND put_date>take_date);
end;
$$;
CREATE FUNCTION
Library=# SELECT * FROM count_of_copybooks_by_book_id(1);
 total
-------
     2
(1 row)

Library=# SELECT * FROM count_of_copybooks_by_book_id(2);
 total
-------
     1
(1 row)

Library=# SELECT * FROM count_of_copybooks_by_book_id(5);
 total
-------
     0
(1 row)
```

2. Для ввода в базу данных новой книги.

```
Library=# create or replace function add_a_new_book(INOUT author VARCHAR(128), INOUT original_language VARCHAR(128),INOUT yea
r integer,INOUT translator VARCHAR(128),INOUT type VARCHAR(128),INOUT part integer,INOUT pen_name VARCHAR(128),INOUT publishe
r_id integer,INOUT isbn VARCHAR(32),INOUT name text,INOUT language VARCHAR(128))
Library-# as
Library-# $$
Library$# begin
Library$#     INSERT INTO "Book"("author", "original_language ", "year", "translator", "type", "part", "pen_name", "publisher
_id", "isbn", "name", "language")
Library$# VALUES(author, original_language, year, translator, type, part, pen_name, publisher_id, isbn, name, language);
Library$# RAISE NOTICE 'THE BOOK HAS BEEN ADDED';
Library$# end;
Library$# $$ language plpgsql VOLATILE;
CREATE FUNCTION
Library=#
Library=#
Library=#
Library=#
Library=#
Library=#
Library=# SELECT add_a_new_book('Yuri', 'RU', 2005, 'Bulkin', 'Guide', 1, 'Stranger', 1, '3423553', 'TESTBOOK', 'EN');
NOTICE:  THE BOOK HAS BEEN ADDED
                       add_a_new_book
-------------------------------------------------------------
 (Yuri,RU,2005,Bulkin,Guide,1,Stranger,1,3423553,TESTBOOK,EN)
(1 row)

Library=#
```

3. *Для ввода нового читателя (необходимо проверить наличие читателя в картотеке, чтобы не назначить ему номер вторично)*

```
Library=# create or replace function add_new_user(_phone_number text, _first_name text, _last_name text, _address text, _grade text) RETURNS VOID
Library-# as
Library-# $$
Library$# declare
Library$# _count integer;
Library$# _user_id integer;
Library$# begin
Library$# SELECT COUNT(*) INTO _count  FROM "User" WHERE phone_number=_phone_number AND first_name=_first_name AND last_name=_last_name AND address=
_address AND grade=_grade GROUP BY user_id;
Library$# IF _count = 1 THEN
Library$#     SELECT user_id INTO _user_id  FROM "User" WHERE phone_number=_phone_number AND first_name=_first_name AND last_name=_last_name AND add
ress=_address AND grade=_grade GROUP BY user_id;
Library$#     RAISE NOTICE 'THE USER HAS ALREADY BEEN ADDED WITH ID %', _user_id;
Library$# ELSE INSERT INTO "User"(phone_number, first_name, last_name, address, grade) VALUES (_phone_number, _first_name, _last_name, _address, _gr
ade);
Library$#     RAISE NOTICE 'THE USER % % HAS BEEN ADDED', _first_name, _last_name;
Library$#     END IF;
Library$# end;
Library$# $$ language plpgsql VOLATILE;
CREATE FUNCTION
Library=#
Library=#
Library=#
Library=#
Library=# SELECT add_new_user('3545345', 'Petya', 'Pupkin', 'Lomonosova, 9', 'Student');
NOTICE:  THE USER Petya Pupkin HAS BEEN ADDED
 add_new_user
--------------

(1 row)

Library=# SELECT add_new_user('3545345', 'Petya', 'Pupkin', 'Lomonosova, 9', 'Student');
NOTICE:  THE USER HAS ALREADY BEEN ADDED WITH ID 202
 add_new_user
--------------

(1 row)
```

Задание 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.

```
Library=# CREATE OR REPLACE FUNCTION add_to_log() RETURNS TRIGGER AS $$
Library$# DECLARE
Library$#     mstr varchar(30);
Library$#     astr varchar(100);
Library$#     retstr varchar(254);
Library$# BEGIN
Library$#     IF    TG_OP = 'INSERT' THEN
Library$#         astr = NEW;
Library$#         mstr := 'Add new data ';
Library$#         retstr := mstr||astr;
Library$#         INSERT INTO logs(text,added,table_name) values (retstr,NOW(), TG_TABLE_NAME);
Library$#         RETURN NEW;
Library$#     ELSIF TG_OP = 'UPDATE' THEN
Library$#         astr = NEW;
Library$#         mstr := 'Update data ';
Library$#         retstr := mstr||astr;
Library$#         INSERT INTO logs(text,added,table_name) values (retstr,NOW(), TG_TABLE_NAME);
Library$#         RETURN NEW;
Library$#     ELSIF TG_OP = 'DELETE' THEN
Library$#         astr = OLD;
Library$#         mstr := 'Remove data ';
Library$#         retstr := mstr || astr;
Library$#         INSERT INTO logs(text,added,table_name) values (retstr,NOW(), TG_TABLE_NAME);
Library$#         RETURN OLD;
Library$#     END IF;
Library$# END;
Library$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
Library=#
Library=#
Library=#
Library=# CREATE TRIGGER t_user AFTER INSERT OR UPDATE OR DELETE ON "User" FOR EACH ROW EXECUTE PROCEDURE add_to_log ();
CREATE TRIGGER
Library=# CREATE TRIGGER t_employee AFTER INSERT OR UPDATE OR DELETE ON "Employee" FOR EACH ROW EXECUTE PROCEDURE add_to_log ();
CREATE TRIGGER
Library=#
```

Проверка:

```
Library=# INSERT INTO "User"(phone_number, first_name, last_name, address, grade) VALUES ('53425', 'Roma', 'Bulkin', 'Lomonosova, 8', 'Studet');
INSERT 0 1
Library=# UPDATE "User" SET grade='Master' WHERE first_name='Roma' AND last_name='Bulkin' AND phone_number='53425';
UPDATE 1
Library=# DELETE FROM "User" WHERE first_name='Roma' AND last_name='Bulkin' AND phone_number='53425';
DELETE 1
Library=# INSERT INTO "Employee"(first_name, last_name, position) VALUES('Nadegda', 'Vadimova', 'Cleaner');
INSERT 0 1
Library=# UPDATE "Employee" SET position='Plumber' WHERE first_name='Nadegda' AND last_name='Vadimova';
UPDATE 1
Library=# DELETE FROM "Employee" WHERE first_name='Nadegda' AND last_name='Vadimova';
DELETE 1
Library=#
```

```
Library=# SELECT * FROM logs;
                            text                             |           added            | table_name
-------------------------------------------------------------+----------------------------+------------
 Add new data (208,53425,Roma,Bulkin,"Lomonosova, 8",Studet) | 2022-04-27 12:56:15.379237 | User
 Update data (208,53425,Roma,Bulkin,"Lomonosova, 8",Master)  | 2022-04-27 12:56:28.372446 | User
 Remove data (208,53425,Roma,Bulkin,"Lomonosova, 8",Master)  | 2022-04-27 12:57:36.128549 | User
 Add new data (5,Nadegda,Vadimova,Cleaner)                   | 2022-04-27 13:07:42.917271 | Employee
 Update data (5,Nadegda,Vadimova,Plumber)                    | 2022-04-27 13:09:22.383458 | Employee
 Remove data (5,Nadegda,Vadimova,Plumber)                    | 2022-04-27 13:09:42.457768 | Employee
(6 rows)

Library=#
```

Вывод:

В ходе выполнения лабораторной работы я создал хранимые процедуры и триггер для логирования изменения данных, который универсален для всех таблиц базы данных.