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 К3240 Вали Насибулла.

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.

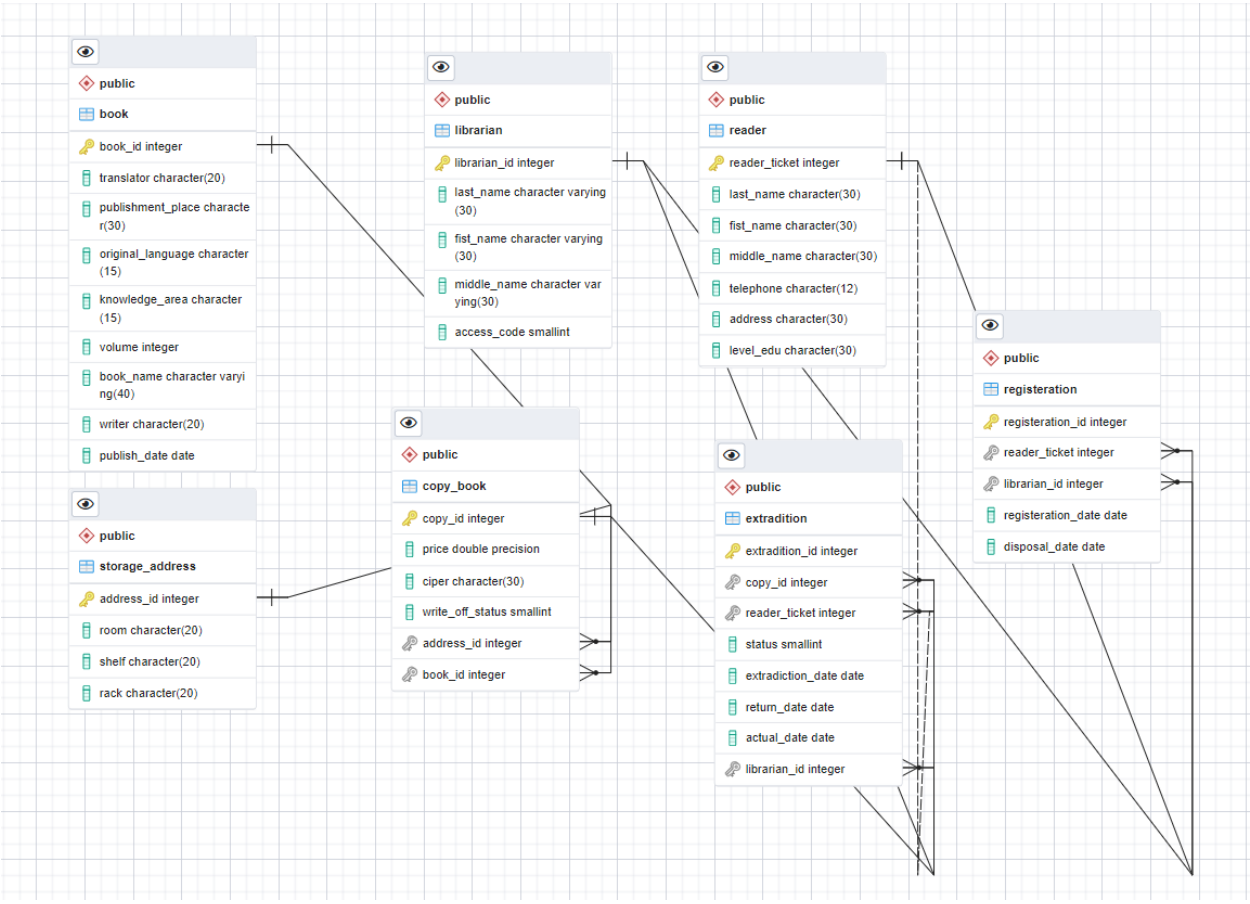
Вариант 3

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).*

```
create or replace function count_copiedBooks_by_BookId_try2(id integer)

returns table

(

total bigint

)

language plpgsql

as

$$

begin

        return query(SELECT COUNT (*) FROM copy_book AS t1 RIGHT JOIN extradition AS t2 ON

t1.copy_id=t2.copy_id Where t1.book_id = id AND t2.status=0);

end;

$$;
```

CREATE FUNCTION;

```
Lab_2.2=# SELECT * FROM count_copiedBooks_by_BookId_try2(2);
 total
-------
     1
(1 эки▮·Ё)


Lab_2.2=# SELECT * FROM count_copiedBooks_by_BookId_try2(1);
 total
-------
     1
(1 эки▮·Ё)


Lab_2.2=# SELECT * FROM count_copiedBooks_by_BookId_try2(4);
 total
-------
     0
(1 эки▮·Ё)
```

2-Для ввода в базу данных новой книги.

```
create or replace function new_book(
INOUT book_id integer,
INOUT translator character(20),
INOUT publishment_place character (30),
INOUT original_language character(15),
INOUT knowledge_area character (15) ,
INOUT volume integer,
INOUT book_name character (40),
INOUT writer character(20),
INOUT publish_date date)
as
$$
begin
INSERT INTO book (book_id, translator, publishment_place, original_language, knowledge_area, volume, book_name, writer, publish_date)
VALUES(book_id,translator,publishment_place,original_language,knowledge_area,volume,book_name, writer,publish_date);
RAISE NOTICE 'ADDED';
end;
$$ language plpgsql VOLATILE;
```

Query Editor | Query History

```
create or replace function new_book(INOUT book_id integer, INOUT translator character(20),
                                    INOUT publishment_place character (30), INOUT original_language character(15),
                                    INOUT knowledge_area character (15) , INOUT volume integer, INOUT book_name character (40),
                                    INOUT writer character(20),INOUT publish_date date)
as
$$
begin
INSERT INTO book (book_id,translator,publishment_place,original_language,knowledge_area,volume,book_name,writer,publish_date)
VALUES(book_id,translator,publishment_place,original_language,knowledge_area,volume,book_name,writer,publish_date);
RAISE NOTICE 'ADDED';
end;
$$ language plpgsql VOLATILE;

```

Data Output | Explain | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 40 msec.
```

Query Editor | Query History

```
SELECT new_book(9,'translator_7','publish_place_7','Pashto','literature',1,'Pashto literature','Writer_7','2012-02-09');
```

Data Output | Explain | Messages | Notifications

	new_book record
1	(9,translator_7,publish_place_7,Pashto,literature,1,"Pashto literature",Writer_7,2012-02-09)

Query Editor | Query History

```
1  select * from book;
```

Data Output | Explain | Messages | Notifications

	book_id [PK] integer	translator character (20)	publishment_place character (30)	original_language character (15)	knowledge_area character (15)	volume integer	book_name character varying (40)	writer character (20)	publish_date date
1	1	Sky	New York	English	Science	1	SkyBook	SkyWriter	2008-11-11
2	2	Earth	New York	English	Earth	1	EarthBook	EarthWriter	2010-11-11
3	3	Sun	New York	Italian	Sun	1	SunBook	SunWriter	2020-11-11
4	34	translator34	place34	Franch	sth	2	Skills Deve	writer34	1900-01-02
5	322	translator322	address322	English	Presentation	1	Moonbook	writer322	1955-09-09
6	9	translator_7	publish_place_7	Pashto	literature	1	Pashto literature	Writer_7	2012-02-09

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

```
create or replace function add_new_reader(_reader_ticket integer,_last_name
character(30),_fist_name character(30),  _middle_name character(30),_telephone
character(12),_address character(30),_level_edu character(30))

RETURNS VOID  as

$$ declare

_count integer;

begin

SELECT COUNT(*) INTO _count FROM reader WHERE reader_ticket=_reader_ticket

AND last_name=_last_name AND fist_name = _fist_name AND middle_name= _middle_name

AND telephone= _telephone AND address= _address AND level_edu= _level_edu GROUP BY
reader_ticket;

IF _count =1 THEN

        SELECT reader_ticket INTO _reader_ticket FROM reader WHERE reader_ticket=_reader_ticket
AND last_name=_last_name

        AND fist_name = _fist_name AND middle_name= _middle_name AND telephone= _telephone

        AND address= _address AND level_edu= _level_edu GROUP BY reader_ticket;

RAISE NOTICE 'THE READER HAS ALREADY BEEN ADDED WITH TICKET %', _reader_ticket;

ELSE

INSERT INTO reader (reader_ticket,last_name,fist_name, middle_name, telephone, address, level_edu)

 VALUES (_reader_ticket,_last_name,_fist_name, _middle_name,_telephone,_address,_level_edu);
```

RAISE NOTICE 'THE READER % % HAS BEEN ADDED', _fist_name, _last_name;

END IF; end; $$ language plpgsql VOLATILE;

Query Editor Query History

```
1  create or replace function add_new_reader(_reader_ticket integer,_last_name character(30),_fist_name character(30),
2                          _middle_name character(30),_telephone character(12),_address character(30),_level_edu character(30))
3  RETURNS VOID
4  as
5  $$
6  declare
7  _count integer;
8  begin
9  SELECT COUNT(*) INTO _count FROM reader WHERE reader_ticket=_reader_ticket
10 AND last_name=_last_name AND fist_name = _fist_name AND middle_name= _middle_name
11 AND telephone= _telephone AND address= _address AND level_edu= _level_edu GROUP BY reader_ticket;
12 IF _count =1 THEN
13     SELECT reader_ticket INTO _reader_ticket FROM reader WHERE reader_ticket=_reader_ticket AND last_name=_last_name
14     AND fist_name = _fist_name AND middle_name= _middle_name AND telephone= _telephone
15     AND address= _address AND level_edu= _level_edu GROUP BY reader_ticket;
16 RAISE NOTICE 'THE READER HAS ALREADY BEEN ADDED WITH TICKET %', _reader_ticket;
17 ELSE
18 INSERT INTO reader (reader_ticket,last_name,fist_name, middle_name, telephone, address, level_edu)
19  VALUES (_reader_ticket,_last_name,_fist_name, _middle_name,_telephone,_address,_level_edu);
20 RAISE NOTICE 'THE READER % % HAS BEEN ADDED', _fist_name, _last_name;
21 END IF;   end;   $$ language plpgsql VOLATILE;
```

Data Output Explain Messages Notifications

```
CREATE FUNCTION

Query returned successfully in 41 msec.
```

Query Editor Query History

```
1  SELECT add_new_reader(32,'Last_namefunc','first_namefunc','middle_namefunc','8495400434','addressfunc','HIGHT');
```

Data Output Explain Messages Notifications

	add_new_reader void
1	

Query Editor | Query History

```
1  select * from reader;
```

Data Output | Explain | Messages | Notifications

	reader_ticket [PK] integer	last_name character varying (30)	fist_name character varying (30)	middle_name character varying (30)	telephone character (12)	address character varying (30)	level_edu character varying (30)
1	1	Kerry	John	freaken	34030490394	Canada	Hight
2	2	Harry	Potar	Crazy	34034330394	London	Secondry
3	3	John	Cena	Wwe	4545330394	Brazil	Hight
4	32	Last_namefunc	first_namefunc	middle_namefunc	8495400434	addressfunc	HIGHT
5	343	Kerry	John	freaken	34030490394	Canada	HIGHT

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.

```
create or replace FUNCTION my_logs()
RETURNS TRIGGER AS $$
DECLARE
        word1 varchar(30);
        word2 varchar(500);
        word3 varchar(300);
BEGIN
    IF TG_OP = 'INSERT' THEN
        word2 = NEW;
        word1 := 'Add new data';
        word3 := word1 || word2;
        INSERT INTO logs(inform,added,name_table) values(word3,NOW(),TG_TABLE_NAME);
        RETURN NEW;
    ELSIF TG_OP = 'UPDATE' THEN
        word2 = NEW;
        word1 := 'update data ';
        word3 := word1 || word2;
```

```
        INSERT INTO logs(inform,added,name_table) values(word3,NOW(),TG_TABLE_NAME);
        RETURN NEW;
    ELSIF TG_OP = 'DELETE' THEN
        word2 = OLD;
        word1 := 'Delete data ';
        word3 := word1 || word2;
        INSERT INTO logs(inform,added,name_table) values(word3,NOW(),TG_TABLE_NAME);
        RETURN OLD;
    END IF;
END;
$$ LANGUAGE plpgsql;
```

Query Editor | Query History

```
1  create or replace FUNCTION my_logs()
2  RETURNS TRIGGER AS $$
3  DECLARE
4          word1 varchar(30);
5          word2 varchar(500);
6          word3 varchar(300);
7  BEGIN
8      IF TG_OP = 'INSERT' THEN
9          word2 = NEW;
10         word1 := 'Add new data';
11         word3 := word1 || word2;
12         INSERT INTO logs(inform,added,name_table) values(word3,NOW(),TG_TABLE_NAME);
13         RETURN NEW;
14     ELSIF TG_OP = 'UPDATE' THEN
15         word2 = NEW;
16         word1 := 'update data ';
17         word3 := word1 || word2;
18         INSERT INTO logs(inform,added,name_table) values(word3,NOW(),TG_TABLE_NAME);
19         RETURN NEW;
20     ELSIF TG_OP = 'DELETE' THEN
21         word2 = OLD;
22         word1 := 'Delete data ';
23         word3 := word1 || word2;
24         INSERT INTO logs(inform,added,name_table) values(word3,NOW(),TG_TABLE_NAME);
25         RETURN OLD;
26     END IF;
```

Data Output | Explain | Messages | Notifications

```
CREATE FUNCTION

Query returned successfully in 41 msec.
```

Create TRIGGER reader_teg AFTER INSERT OR UPDATE OR DELETE ON "reader"

FOR EACH ROW EXECUTE PROCEDURE my_logs();

Query Editor | Query History

```
1  Create TRIGGER reader_teg AFTER INSERT OR UPDATE OR DELETE ON "reader"
2  FOR EACH ROW EXECUTE PROCEDURE my_logs();
```

Data Output | Explain | Messages | Notifications

```
CREATE TRIGGER

Query returned successfully in 35 msec.
```

...

Query Editor | Query History

```
1  INSERT INTO public.reader(
2      reader_ticket, last_name, fist_name, middle_name, telephone, address, level_edu)
3      VALUES (104, 'test2', 'test2', 'test2', '7434398439', 'test2', 'test2');
```

Data Output | Explain | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 36 msec.
```

...

Query Editor | Query History

```
1  select * from logs;
```

Data Output | Explain | Messages | Notifications

	inform text	added text	name_table text
1	Add new data(104,"test2 ","test2 ","tes...	2022-05-09 16:40:00.371597+03	reader

Вывод:

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