**Министерство науки и высшего образования Российской Федерации**

**Федеральное государственное автономное образовательное**

**учреждение высшего образования**

**«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»**

**Факультет инфокоммуникационных технологий**

**ОТЧЕТ**

О ЛАБОРАТОРНОЙ РАБОТЕ **№ 3**

Проектирование и реализация баз данных

процедуры, функции, триггеры в PostgreSQL

**Специальность:**

09.03.03 Мобильные и сетевые технологии

**Проверил:**

Говорова М. М.

**Выполнил:**

студент группы K3240 Мамедов Тогрул

**Дата:** «03» 06 2022г.

**Оценка** \_\_\_\_\_\_\_\_\_\_\_\_\_\_

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).

**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

Изображение выглядит как текст, карта, внутренний, несколько

Автоматически созданное описание

**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;

Изображение выглядит как текст

Автоматически созданное описание

**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;

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

**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;

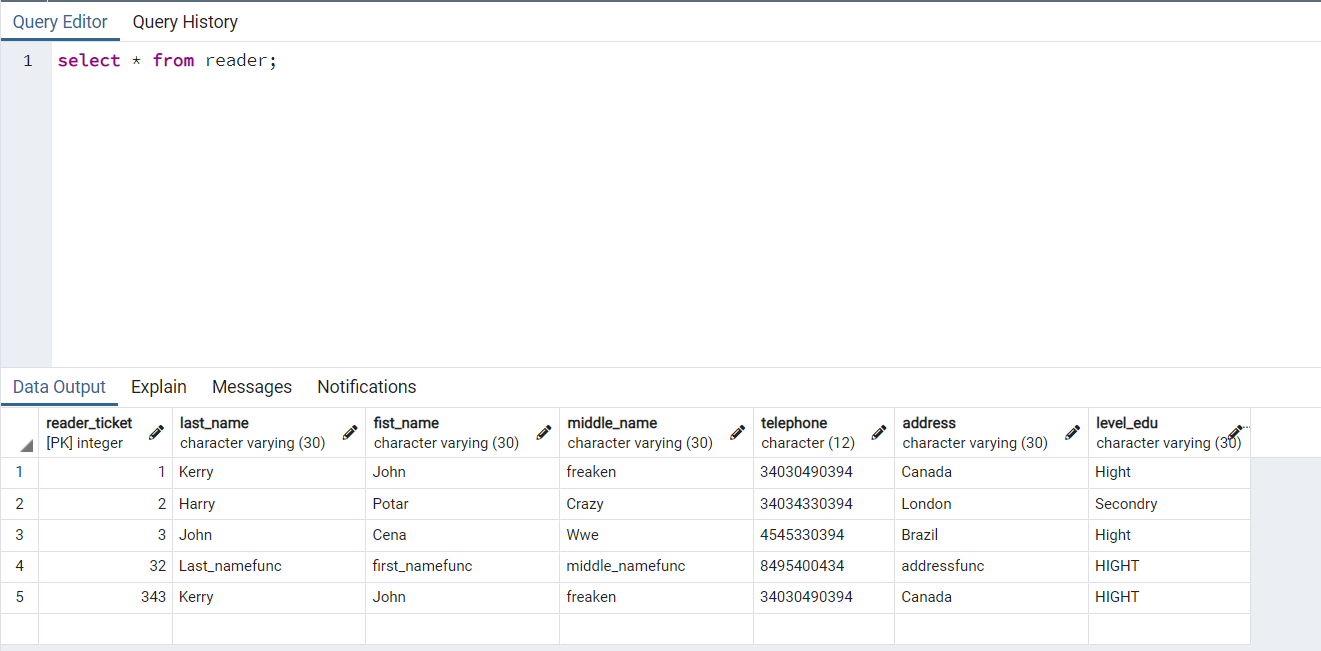
END IF; end; $$ language plpgsql VOLATILE;

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание



**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;

Изображение выглядит как текст

Автоматически созданное описание

Create TRIGGER reader\_teg AFTER INSERT OR UPDATE OR DELETE ON "reader"

FOR EACH ROW EXECUTE PROCEDURE my\_logs();

Изображение выглядит как текст

Автоматически созданное описание

…

Изображение выглядит как текст

Автоматически созданное описание

…

Изображение выглядит как текст

Автоматически созданное описание

**Вывод:**

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