Санкт-Петербургский Политехнический Университет Петра Великого

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**ОТЧЕТ**

**по лабораторной работе**

**«Триггеры, вызовы процедур»**

По дисциплине «Базы данных»

Работу выполнил студент

Группа 3530901/60202

Курякин Д. А.

Работу принял преподаватель

Мяснов А.В.

Санкт-Петербург

2020

**Содержание**

[**1.** **Цель работы** 3](#_Toc38838999)

[**2.** **Программа работы** 3](#_Toc38839000)

[**3.** **Стандартные запросы** 3](#_Toc38839001)

[**3.1** **Триггер для автоматического заполнения ключевого поля** 3](#_Toc38839002)

[**3.2** **Триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице** 3](#_Toc38839003)

[**4.** **Выполнение заданий преподавателя** 5](#_Toc38839004)

[**4.1.**  **Задание 1** 5](#_Toc38839005)

[**4.2.**  **Задание 2** 6](#_Toc38839006)

[**Вывод** 7](#_Toc38839007)

**1. Цель работы**

Познакомиться с обработкой данных на стороне сервера с помощью хранимых процедур и тригеров.

**2. Программа работы**

1. Создание двух триггеров: один триггер для автоматического заполнения ключевого поля, второй триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице.

2. Создание триггера в соответствии с индивидуальным заданием, полученным у преподавателя.

3. Создание триггера в соответствии с индивидуальным заданием, вызывающего хранимую процедуру.

4. Выкладывание скриптов в GitLab.

5. Демонстрация результатов работы преподавателю.

**3. Стандартные запросы**

**3.1 Триггер для автоматического заполнения ключевого поля**

Создадим триггер, который при добавление данных в таблицу violation без id будет добавлять к максимальному id две единицы.

CREATE OR REPLACE FUNCTION *violation\_auto\_key*() RETURNS trigger AS $violation\_auto\_key$  
 BEGIN  
 new.id = (SELECT *max*(id) FROM violation) + 2;  
 RETURN new;  
 END;  
 $violation\_auto\_key$ LANGUAGE plpgsql;  
  
CREATE TRIGGER violation\_auto\_key\_trigger  
 BEFORE INSERT OR UPDATE ON violation  
 FOR EACH ROW  
 EXECUTE PROCEDURE *violation\_auto\_key*();

SELECT *max*(id) FROM violation;

  
  
insert into violation (title, punishment)  
VALUES (1.1,'Test1')

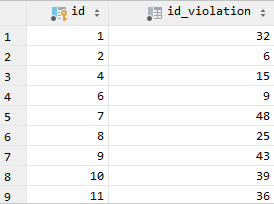
SELECT *max*(id) FROM violation;



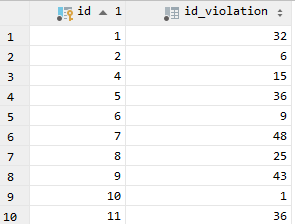
**3.2 Триггер для контроля целостности данных в подчиненной таблице при удалении/изменении записей в главной таблице**

Создадим тригер который при удаление violation все штрафы принимают другое violation.  
CREATE OR REPLACE FUNCTION *violation\_control*() RETURNS trigger AS $violation\_control$  
 DECLARE  
 i BIGINT;  
 BEGIN  
 i = 1;  
 IF (SELECT id FROM violation WHERE (SELECT *COUNT*(*\**) FROM violation) = 1) IS NOT NULL and (SELECT id FROM fine) IS NOT NULL THEN  
 RAISE EXCEPTION 'First remove all the fines';  
 end if;  
 FOR i IN (SELECT id FROM violation)  
 LOOP  
 IF(SELECT id FROM violation WHERE id = i and id != old.id) IS NOT NULL THEN  
 UPDATE fine SET id\_violation = i where id\_violation = old.id;  
 RETURN old;  
 ELSE  
 i = i + 1;  
 END IF;  
 end loop;  
 END;  
$violation\_control$ LANGUAGE plpgsql;  
  
CREATE TRIGGER violation\_control\_trigger  
 BEFORE DELETE ON violation  
 FOR EACH ROW  
 EXECUTE PROCEDURE *violation\_control*();

SELECT id, id\_violation FROM fine;

  
  
DELETE FROM violation where id = 39;

SELECT id, id\_violation FROM fine;



Из рисунков видно, что при удалении violation равному 39 id\_violation заменилось с 39 на 1.

**4. Выполнение заданий преподавателя**

**4.1. Задание 1**

При фиксации нарушения проверять наличие действующего водительского удостоверения на момент нарушения. Если нет - выбрасывать исключение и не добавлять.

Берем сегодняшнею дату и дату окончания водительского удостоверение, сравниваем, если водительское удостоверение просрочено, то выкидываем исключение, если не просрочено, то добавляем штраф.

CREATE OR REPLACE FUNCTION valid\_dl() RETURNS trigger AS $valid\_dl$  
 DECLARE  
 today\_date timestamp;  
 tmp timestamp;  
 BEGIN  
  
 today\_date = statement\_timestamp();  
 SELECT driver\_license.end\_date\_and\_time  
 INTO tmp  
 FROM driver\_license  
 WHERE NEW.driver\_license = driver\_license.id;  
  
 IF (today\_date < tmp) THEN  
 RETURN NEW;  
 ELSE  
 RAISE EXCEPTION 'Driver license expired.';

END IF;  
 END;  
 $valid\_dl$ LANGUAGE plpgsql;  
  
CREATE TRIGGER valid\_dl\_trigger  
 BEFORE INSERT OR UPDATE ON fine  
 FOR EACH ROW  
 EXECUTE PROCEDURE valid\_dl();

INSERT INTO fine (driver\_license, police\_certificate, data\_and\_time, id\_violation)  
VALUES (34, 5, '2020-10-5', 5);



INSERT INTO fine (driver\_license, police\_certificate, data\_and\_time, id\_violation)  
VALUES (35, 5, '2020-10-5', 5);

select fine.id, fine.data\_and\_time, fine.id\_violation,  
 driver\_license.id, driver\_license.data\_and\_time\_of\_issue, driver\_license.end\_date\_and\_time  
from fine  
join driver\_license on fine.driver\_license = driver\_license.id  
where fine.driver\_license = 35;



**4.2. Задание 2**

При включении категории в водительское удостоверение проверять наличие. Если уже есть - выбрасывать исключение и не добавлять дубль.CREATE OR REPLACE FUNCTION unique\_category() RETURNS trigger AS $unique\_category$  
 DECLARE  
 tmp integer;  
 BEGIN  
 SELECT categories.id\_categories  
 INTO tmp  
 FROM categories  
 where categories.categories = NEW.categories and categories.id\_categories = NEW.id\_categories;  
  
 IF (tmp IS NULL ) THEN  
 RETURN NEW;  
 ELSE  
 RAISE EXCEPTION 'Category not unique.';  
 END IF;  
 END;  
 $unique\_category$ LANGUAGE plpgsql;  
  
CREATE TRIGGER unique\_category\_trigger  
 BEFORE INSERT OR UPDATE ON categories  
 FOR EACH ROW  
 EXECUTE PROCEDURE unique\_category();

select driver\_license.id, driver\_license.categories, dc.id, dc.name  
from driver\_license  
join categories c on driver\_license.categories = c.categories  
join dir\_categories dc on c.id\_categories = dc.id  
where driver\_license.id = 35;

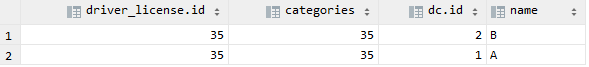


INSERT INTO categories (categories, id\_categories)  
VALUES (35,2);



INSERT INTO categories (categories, id\_categories)  
VALUES (35,1);

select driver\_license.id, driver\_license.categories, dc.id, dc.name  
from driver\_license  
join categories c on driver\_license.categories = c.categories  
join dir\_categories dc on c.id\_categories = dc.id  
where driver\_license.id = 35;



**Вывод**

В процессе выполнения данной лабораторной работы был получен опыт работы с триггерами и был получен дополнительны опыт работы с процедурами. Также был получен дополнительный опыт работы с SQL запросами для тестирования написанных триггеров.