Кафедра вычислительной техники



**ЛАБОРАТОРНАЯ РАБОТА №4**

**по дисциплине:** Базы данных

**на тему:** Перегружаемые функции, триггеры в PostgreSQL

|  |  |
| --- | --- |
| Выполнил: | Проверил: |
| Студент гр. *АВТ-819* | Бычков |
| *Ванин К.Е.* | Михаил Иванович |
| «\_\_\_» \_\_\_\_\_\_ 20\_\_г. | «\_\_\_» \_\_\_\_\_\_ 20\_\_г. |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| (подпись) | (подпись) |
|  |  |
|  |  |

Оглавление

[ЦЕЛЬ РАБОТЫ: 2](#_Toc65779369)

[ПОСТАНОВКА ЗАДАЧИ: 2](#_Toc65779370)

[ХОД РАБОТЫ 3](#_Toc65779371)

# ЦЕЛЬ РАБОТЫ:

Цель работы: Изучение перегружаемых функций и триггеров в PostgreSQL. Изучить синтаксис команд. Получение навыков работы с триггерами.

# ПОСТАНОВКА ЗАДАЧИ:

*Общая часть*: Ознакомиться с теоретическими сведениями о возможностях создания перегружаемых функций и триггеров в PostgreSQL. Согласно варианту из лабораторной работы №4 создать перегружаемые функции и триггеры. Продемонстрировать работу триггеров на примерах вставки и удаления записей из таблицы. Если в базе нет подходящих данных, то изменить условия, либо добавить подходящие данные. Удалить триггер. По заданию преподавателя создать триггер и проверить его работоспособность. Просмотреть и проанализировать полученную в результате выполнения операций информацию.

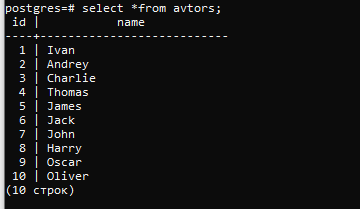
Вариант 4: Триггер   выполняется перед удалением записи из таблицы водителя.  Триггер проверяет наличие в другой таблице записей, относящихся к удаляемому водителю, и, если такие записи есть, удаляет их.

Триггер   выполняется перед вставкой новой записи в таблицу водителя.  Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

* eсли  не указано имя автора – оно генерируется по схеме – Avtor + уникальный номер  из последовательности;
* eсли  не указан номер автора – он генерируется по схеме – S+ уникальный номер  из последовательности;
* eсли  не указан город автора – ставится значение по  умолчанию – “Samara” ;

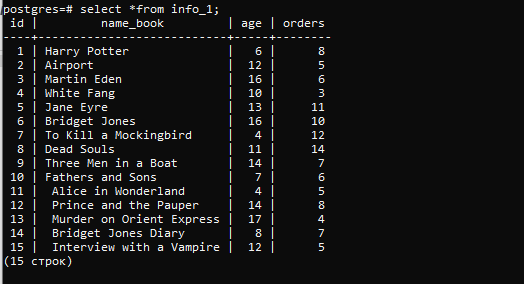
# ХОД РАБОТЫ

Таблица авторов avtors , из которой триггер будет удалять записи (Рис. 1.)



*Рис. 1 – Таблица* avtors *до удаления записей*

Таблица drivers связана с таблицей delivery.



*Рис. 2 – Таблица delivery до удаления записей*

Для создания триггеров используется команда CREATE TRIGGER со следующем синтаксисом: CREATE TRIGGER триггер BEFORE | AFTER } { событие [ OR событие ] } ON таблица FOR EACH { ROW | STATEMENT } EXECUTE PROCEDURE функция ( аргументы )

Триггер   выполняется перед удалением записи из таблицы водителя.  Триггер проверяет наличие в другой таблице записей, относящихся к удаляемому автору, и, если такие записи есть, удаляет их.

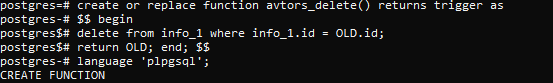
*create or replace function avtors\_delete() returns trigger as*

*$$ begin*

*delete from info\_1 where info\_1.id = OLD.id;*

*return OLD; end; $$*

*language* '*plpgsql*'*;*



*Рис. 3 – Создание триггера функции*

create trigger avtors\_delete before delete on avtors for each row execute procedure avtors\_delete();



*Рис. 4 – Создание триггера*

Удаляем запись о авторах:

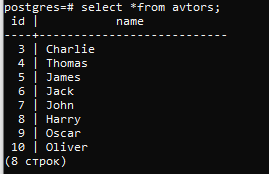
*Delete from avtors d where d.id = 1;*

*Delete from avtors d where d.id = 2;*

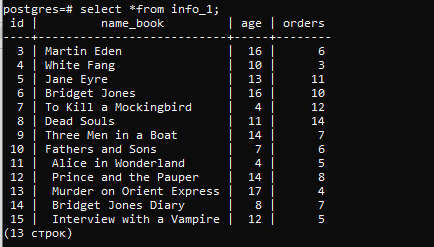


*Рис. 5 – Операция удаления записи водителей*

Таблицы avtors, info\_1 после удаления записей о авторах (Рис. 6, Рис. 7)



*Рис. 6 – Таблица* avtors *после удаления записей*



*Рис. 6 – Таблица* info\_1 *после удаления записей*

Триггер выполняется перед вставкой новой записи в таблицу автора.  Триггер проверяет значения, которые должна содержать новая запись и может их изменить:

* eсли  не указано имя автора – оно генерируется по схеме – Avtor + уникальный номер  из последовательности;
* eсли  не указан номер автора – он генерируется по схеме – S+ уникальный номер  из последовательности;
* eсли  не указан город автора – ставится значение по  умолчанию – “Samara” ;

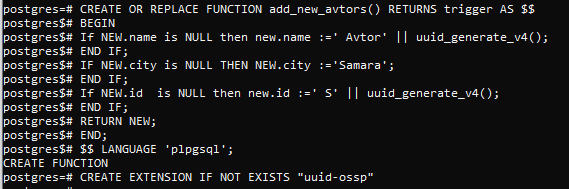
Триггерная функция:

*CREATE OR REPLACE FUNCTION add\_new\_avtors() RETURNS trigger AS $$  
BEGIN  
If*[*NEW.name*](https://vk.com/away.php?utf=1&to=http%3A%2F%2FNEW.name)*is NULL then*[*new.name*](https://vk.com/away.php?utf=1&to=http%3A%2F%2Fnew.name)*:='*Avtor*' || uuid\_generate\_v4();  
END IF;  
IF [NEW.city](https://vk.com/away.php?utf=1&to=http%3A%2F%2FNEW.city" \t "_blank) is NULL THEN [NEW.city](https://vk.com/away.php?utf=1&to=http%3A%2F%2FNEW.city" \t "_blank) :='*Samara*';  
END IF;*

*If NEW.id  is NULL then new.id :='*S*' || uuid\_generate\_v4();  
END IF;  
RETURN NEW;  
END;  
$$ LANGUAGE 'plpgsql';*

Пакет для генерации чисел:

CREATE EXTENSION IF NOT EXISTS *“*uuid-ossp*”*



*Рис. 7 – Создание функции add\_new\_avtors ();*

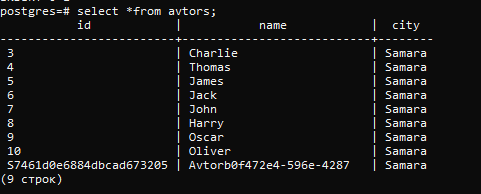
Создаём триггер вставки:

CREATE trigger add\_avtors before insert on info\_1 for each row execute procedure add\_new\_avtors ();



*Рис. 8 – Создание триггера add\_new\_avtors ();*

*Проверяем работоспособность триггера (Рис. 9):*



*Рис. 9 –Работа триггера вставки*

Вывод

В ходе лабораторной работы были изучены правила перегрузки функций и триггеров. Приобретены практические навыки по применению триггеров в среде PostgreSQL.