ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8 |
| Триггеры. Обеспечение активной целостности данных базы данных |
| по дисциплине: ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4131 |  | 18.12.2023 |  | В.А.Алексеев |
|  |  |  | подпись, дата |  | инициалы, фамилия |

1. **Задание.**

Реализовать для своей базы данных триггеры для всех событий (insert,delete,update) до и после. Часть из которых будет обеспечивать ссылочную целостность, остальные могут иметь другое назначение.

а. Вершины, название/текст которых содержит слово «стоп», но не начинается с него

б. Вершины, у которых нет исходящих ребер

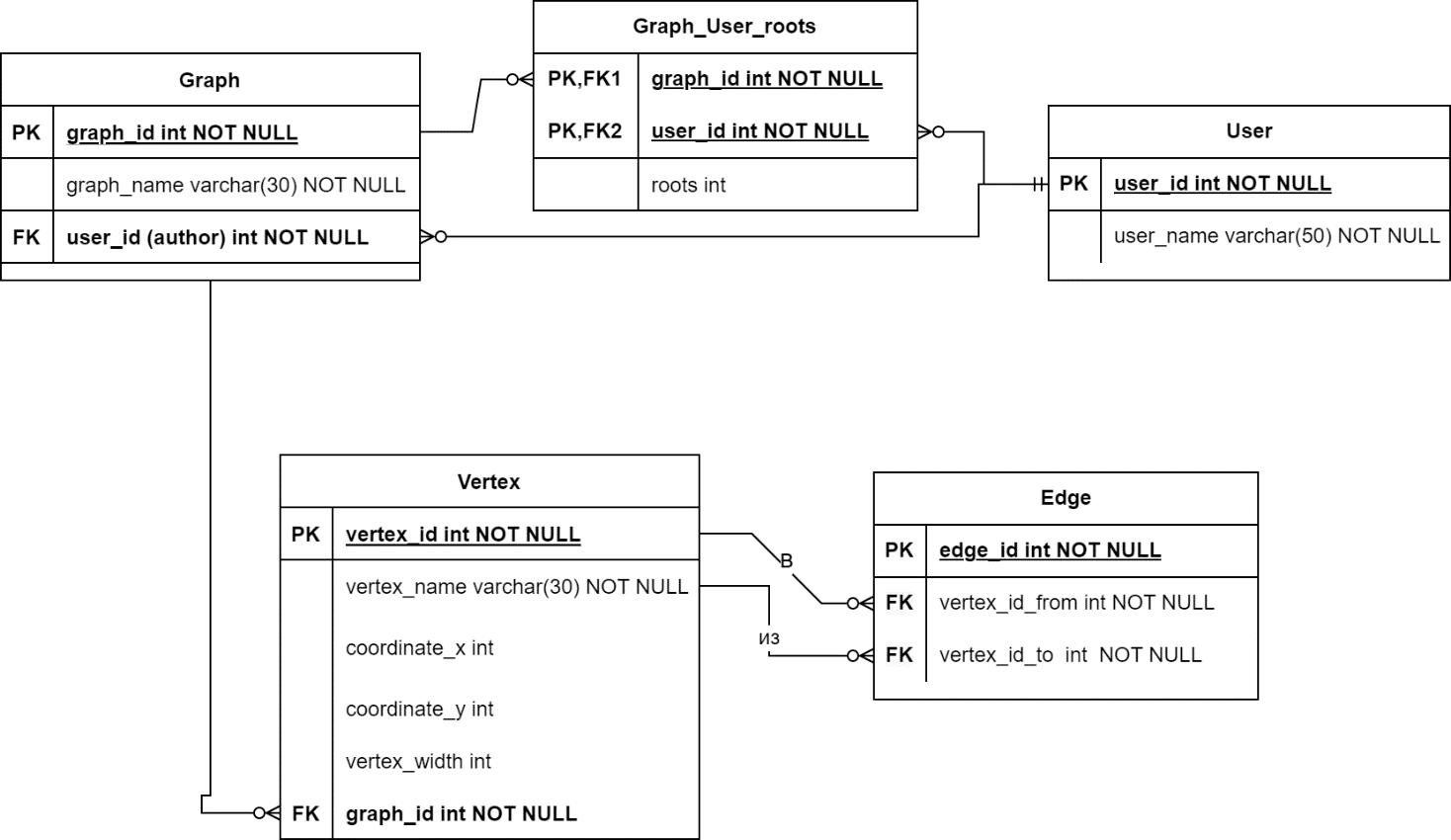
в. Графы, в которых есть вершина с петлей.

г. Графы, ширина которых не превышает 300 пикселей.Ширина графа в пикселях (от максимальной сумма координаты по горизонтали с шириной отнять минимальную левую координату)

д. авторы графов, графы которых может редактировать наибольшее число пользователей

е. Вершины, для которых есть исходящие ребра, ведущие ко всем остальным вершинам еѐ графа ж. Вершины, у которых нет входящих ребер от вершины со словом «выход»

1. **Физическая модель БД.**



1. **Назначение и тексты триггеров**

**А)** Триггер перед insert

Предназначен для проверки уже существующих вершин с таким id, и есть ли граф с вводимым id.

CREATE OR REPLACE FUNCTION before\_insert\_vertex()

RETURNS TRIGGER AS $$

BEGIN

if not exists(select \* from graph where graph.graph\_id=NEW.graph\_id) then

RAISE EXCEPTION 'Граф с id % не существует', NEW.graph\_id;

elseif exists(select \* from vertex where vertex.vertex\_id=NEW.vertex\_id)then

RAISE EXCEPTION 'вершина с id % уже существует', NEW.vertex\_id;

end if;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE or replace TRIGGER before\_insert\_vertex\_trigger

BEFORE INSERT ON vertex

FOR EACH ROW

EXECUTE FUNCTION before\_insert\_vertex();

**Б)** Триггер после insert

Предназначен для записи информации о вставке в лог.

CREATE TABLE log\_vertex (

log\_id SERIAL PRIMARY KEY,

event\_type VARCHAR(10) NOT NULL,

vertex\_id INT NOT NULL,

old\_vertex\_name VARCHAR(50),

new\_vertex\_name VARCHAR(50),

event\_timestamp TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

CREATE OR REPLACE FUNCTION after\_insert\_vertex()

RETURNS TRIGGER AS $$

BEGIN

insert into log\_vertex(event\_type, vertex\_id, new\_vertex\_name, event\_timestamp)

values('INSERT', NEW.vertex\_id, NEW.vertex\_name, NOW());

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE or replace TRIGGER after\_insert\_vertex\_trigger

AFTER INSERT ON vertex

FOR EACH ROW

EXECUTE FUNCTION after\_insert\_vertex();

**В)** Триггер перед update  
Предназначен для проверки корректности нового значения ширины вершины. Также проверяется существование обновленного id.

CREATE OR REPLACE FUNCTION before\_update\_vertex()

RETURNS TRIGGER AS $$

BEGIN

IF NEW.vertex\_width < 0 THEN

RAISE EXCEPTION 'Некорректные значения для ширины вершины.';

END IF;

if exists(select \* from vertex where vertex.vertex\_id=NEW.vertex\_id and NEW.vertex\_id<>OLD.vertex\_id)then

RAISE EXCEPTION 'вершина с id % уже существует', NEW.vertex\_id;

end if;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE or replace TRIGGER before\_update\_vertex\_trigger

BEFORE UPDATE ON vertex

FOR EACH ROW

EXECUTE FUNCTION before\_update\_vertex();

**Г)** Триггер после update

Предназначен для записи информации об обновлении в лог.

CREATE OR REPLACE FUNCTION after\_update\_vertex()

RETURNS TRIGGER AS $$

BEGIN

insert into log\_vertex(event\_type, vertex\_id, old\_vertex\_name,new\_vertex\_name, event\_timestamp)

values('UPDATE', NEW.vertex\_id, OLD.vertex\_name, NEW.vertex\_name, NOW());

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE or replace TRIGGER after\_update\_vertex\_trigger

AFTER UPDATE ON vertex

FOR EACH ROW

EXECUTE FUNCTION after\_update\_vertex();

**Д)** Триггер перед delete.

Предназначен для удаления связанных с вершиной ребер.

CREATE OR REPLACE FUNCTION before\_delete\_vertex()

RETURNS TRIGGER AS $$

BEGIN

IF EXISTS (SELECT \* FROM vertex

join edge on edge.vertex\_id\_from=vertex.vertex\_id

OR edge.vertex\_id\_to=vertex.vertex\_id

WHERE vertex.vertex\_id = OLD.vertex\_id) THEN

delete from edge where edge.vertex\_id\_from = OLD.vertex\_id

OR edge.vertex\_id\_to = OLD.vertex\_id;

RETURN OLD;

END IF;

RETURN NULL;

END;

$$ LANGUAGE plpgsql;

CREATE or replace TRIGGER before\_delete\_vertex\_trigger

BEFORE DELETE ON vertex

FOR EACH ROW

EXECUTE FUNCTION before\_delete\_vertex();

**Е)** Триггер после delete

Предназначен для записи информации об удалении в лог.

CREATE OR REPLACE FUNCTION after\_delete\_vertex()

RETURNS TRIGGER AS $$

BEGIN

insert into log\_vertex(event\_type, vertex\_id, old\_vertex\_name, event\_timestamp)

values('DELETE', OLD.vertex\_id, OLD.vertex\_name, NOW());

RETURN OLD;

END;

$$ LANGUAGE plpgsql;

CREATE or replace TRIGGER after\_delete\_vertex\_trigger

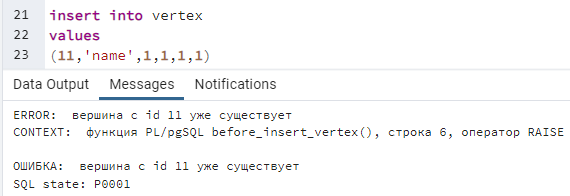
AFTER DELETE ON vertex

FOR EACH ROW

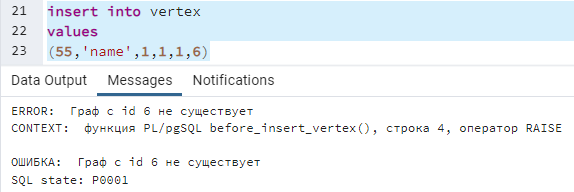
EXECUTE FUNCTION after\_delete\_vertex();

1. **SQL операторы и наборы данных, иллюстрирующие работу триггеров.**

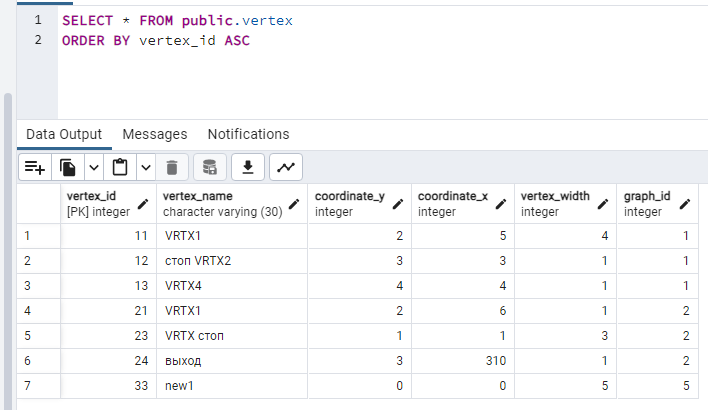
**А)** Добавление вершины с существующим id

****

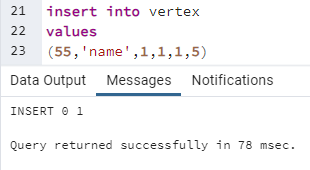
Добавление вершины с несуществующим id графа

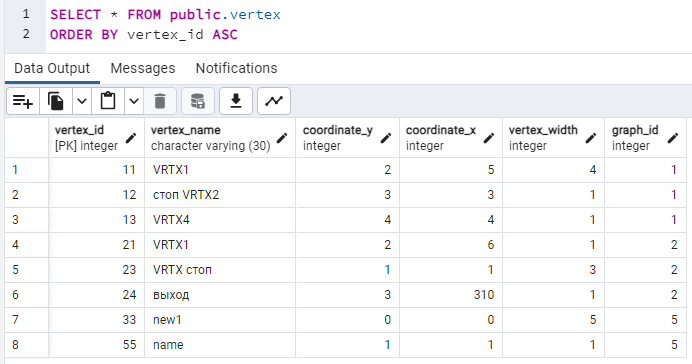
****

Вершины до вставки

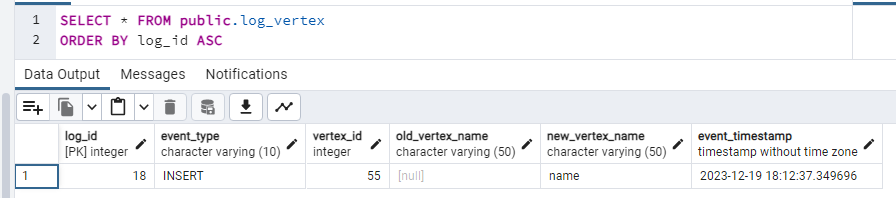


Вершины после вставки



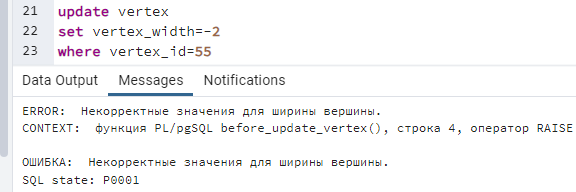


**Б)** Запись в логе после добавления

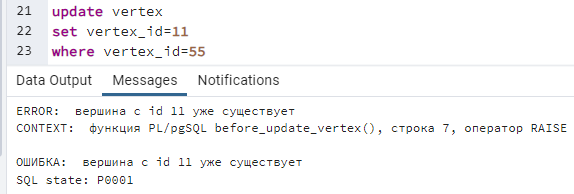


**В)** Обновление

Попытка обновления с отрицательной шириной



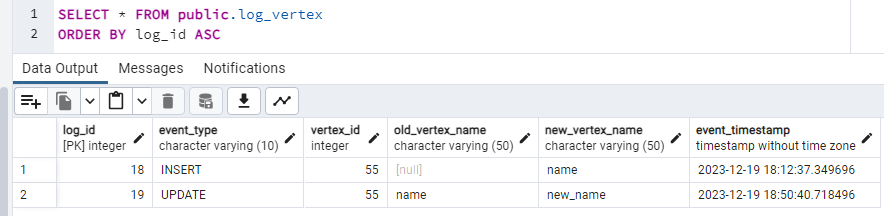
Попытка обновления с другим существующим id



Обновление названия

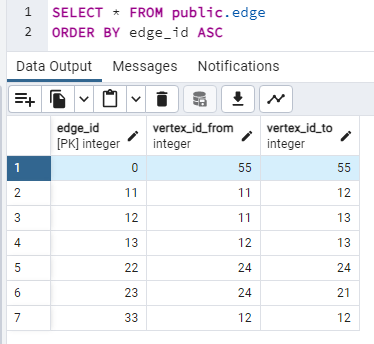


**Г)** Запись в логе после обновления

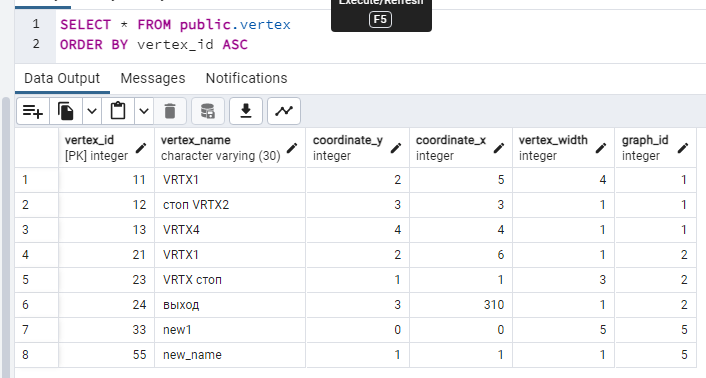


**Д)** Каскадное удаление ребер

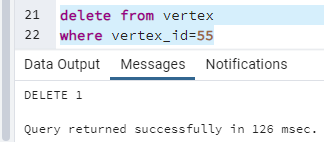
Ребра:



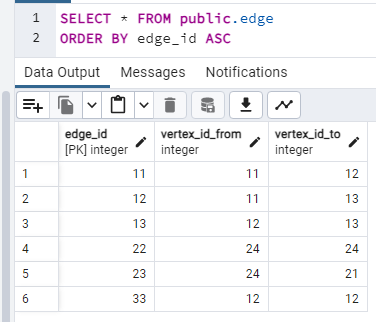
Вершины:



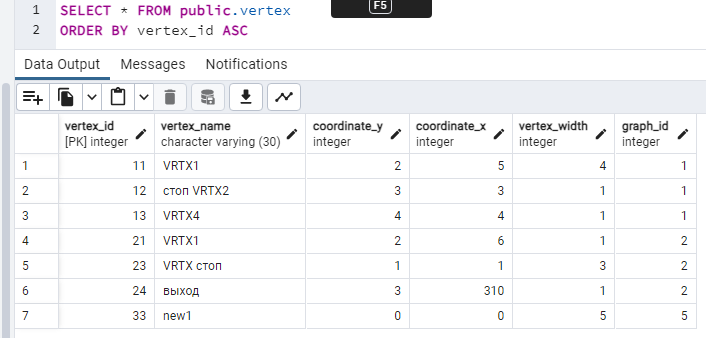
Удаление:



Ребра после удаления:



Вершины после удаления:



Е) Запись в логе после удаления

