ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

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

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7 |
| Хранимые процедуры |
| по дисциплине: ПРОЕКТИРОВАНИЕ БАЗ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4131 |  | 29.11.2023 |  | В.А.Алексеев |
|  |  |  | подпись, дата |  | инициалы, фамилия |

1. **Задание.**

По аналогии с примерами, приведенными в п. 1, создать в БД ХП, реализующие:

— вставку с пополнением справочников;

— удаление с очисткой справочников – удаление всех зависимых данных;

— каскадное удаление;

— вычисление и возврат значения агрегатной функции;

— формирование статистики во временной таблице.

Программа для рисования графов: название вершины, координаты левой верхней точки отображаемой вершины, и еѐ размеры, автор графа, пользователи, которым можно просматривать или редактировать граф

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

б. Вершины, у которых нет исходящих ребер

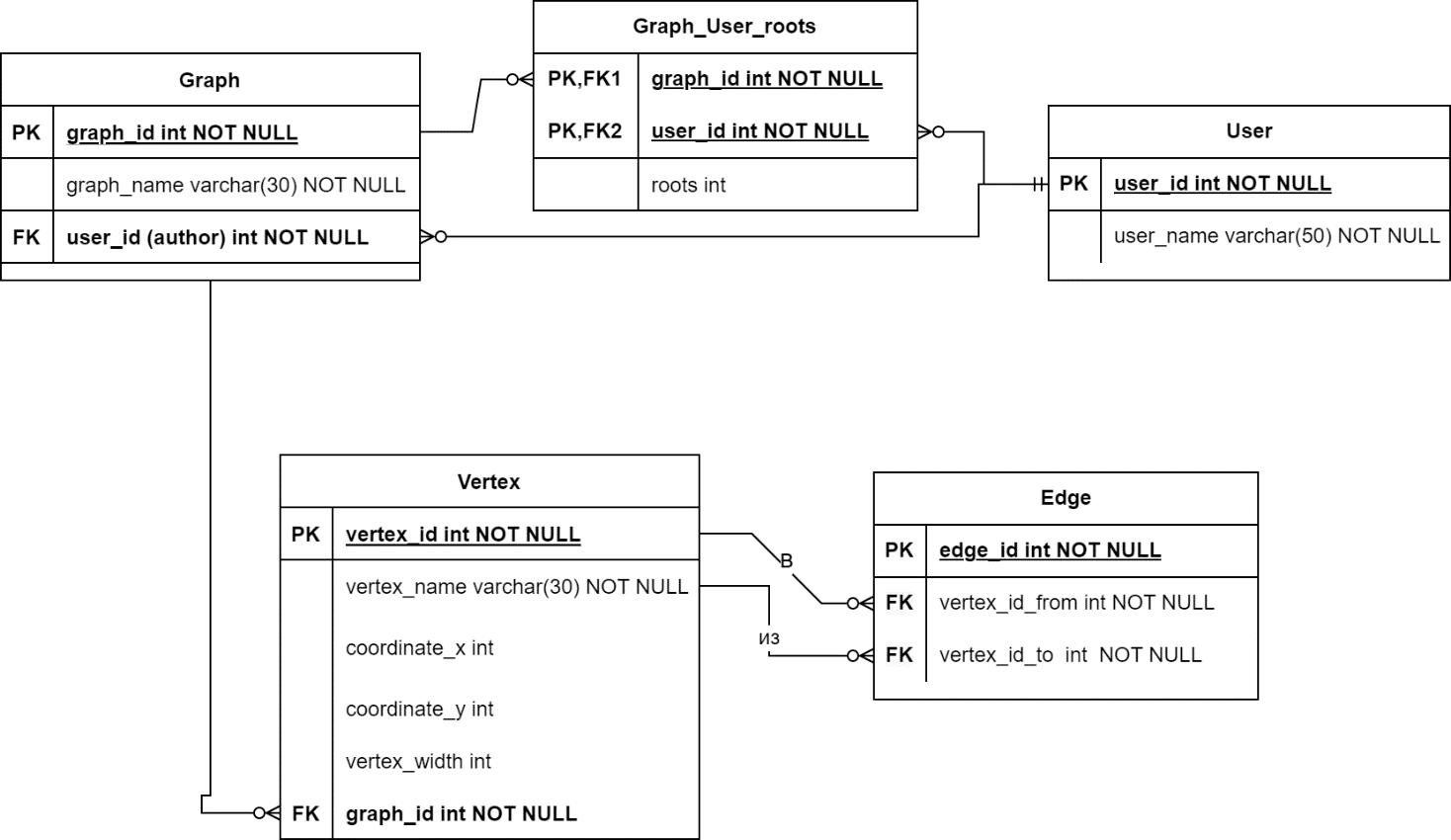
в. Графы, в которых есть вершина с петлей.

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, тексты ХП и их вызовов**

**А)** вставка с пополнением справочников

Хранимая процедура реализует вставку нового графа. Если в таблице авторов не существует автора для вставляемого графа, то происходит добавление нового пользователя.

CREATE OR REPLACE PROCEDURE insert\_graph\_and\_user(IN

graph\_id\_param INT,

graph\_name\_param varchar(30),

user\_id\_param int)

LANGUAGE plpgsql

AS $$

DECLARE

new\_user\_id integer;

BEGIN

if not exists(select from "user" where user\_id = user\_id\_param)

then

begin

new\_user\_id :=(select COALESCE (max( user\_id)+1, 0) from "user");

insert into "user" values (new\_user\_id,'New USER\_'||new\_user\_id::varchar);

end;

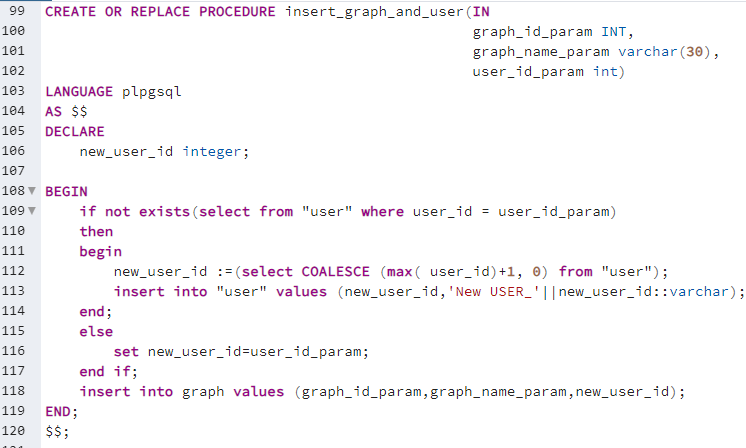
else

set new\_user\_id=user\_id\_param;

end if;

insert into graph values (graph\_id\_param,graph\_name\_param,new\_user\_id);

END;

$$; 

Вызов процедуры:

call insert\_graph\_and\_user(6,'New graph',4);

**Б)** Удаление с очисткой справочников

Процедура реализует удаление вершины. Если вершина была единственной в графе, то удаляется и граф, а также все права пользователей, связанных с этим графом.

CREATE OR REPLACE PROCEDURE del\_vertex\_and\_graph(IN vertex\_id\_param INT)

LANGUAGE plpgsql

AS $$

DECLARE

graph\_of\_vertex integer;

BEGIN

select vertex.graph\_id into graph\_of\_vertex from vertex

where vertex.vertex\_id=vertex\_id\_param;

delete from vertex

where vertex.vertex\_id=vertex\_id\_param;

if not exists (select \* from vertex where vertex.graph\_id=graph\_of\_vertex)

then

begin

delete from graph\_user\_roots where graph\_id=graph\_of\_vertex;

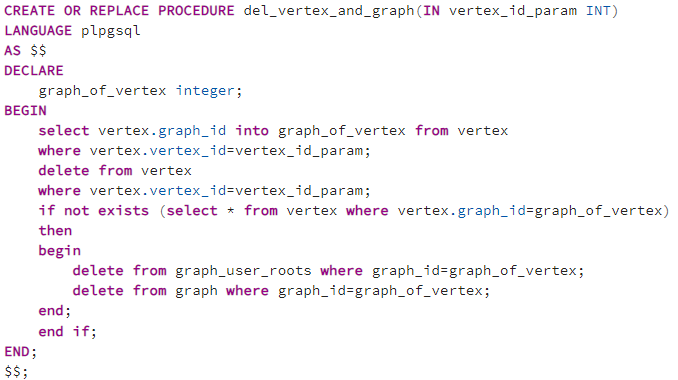
delete from graph where graph\_id=graph\_of\_vertex;

end;

end if;

END;

$$;



Вызов

call del\_vertex\_and\_graph(44);

**В)** Каскадное удаление  
Процедура реализует удаление вершины, а также с ней связанные ребра.

CREATE OR REPLACE PROCEDURE del\_vertex(IN vertex\_id\_param INT)

LANGUAGE plpgsql

AS $$

BEGIN

delete from edge

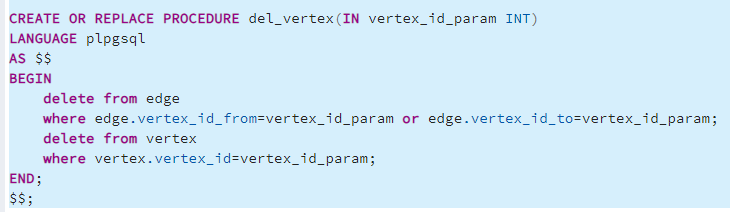
where edge.vertex\_id\_from=vertex\_id\_param or edge.vertex\_id\_to=vertex\_id\_param;

delete from vertex

where vertex.vertex\_id=vertex\_id\_param;

END;

$$;

  
Вызов

call del\_vertex(44);

**Г)** Вычисление и возврат значения агрегатной функции

Процедура и функция возврата количества вершин в заданном графе.

CREATE OR REPLACE PROCEDURE get\_count\_vertex\_in\_graph(IN graph\_id\_param INT, OUT count\_vertex INTEGER)

LANGUAGE plpgsql

AS $$

BEGIN

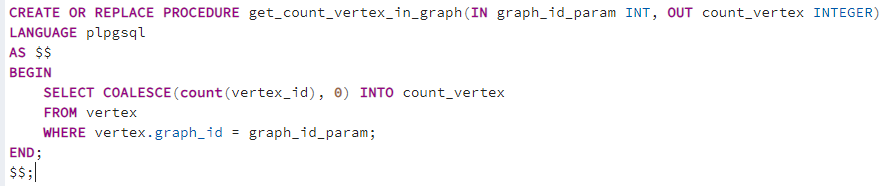
SELECT COALESCE(count(vertex\_id), 0) INTO count\_vertex

FROM vertex

WHERE vertex.graph\_id = graph\_id\_param;

END;

$$;



Вызов

call get\_count\_vertex\_in\_graph(5, NULL);

CREATE OR REPLACE FUNCTION get\_count\_vertex\_in\_graph1(graph\_id\_param INT)

RETURNS INTEGER

LANGUAGE plpgsql

AS $$

DECLARE

count\_v INTEGER;

BEGIN

SELECT COALESCE(count(vertex\_id), 0) INTO count\_v

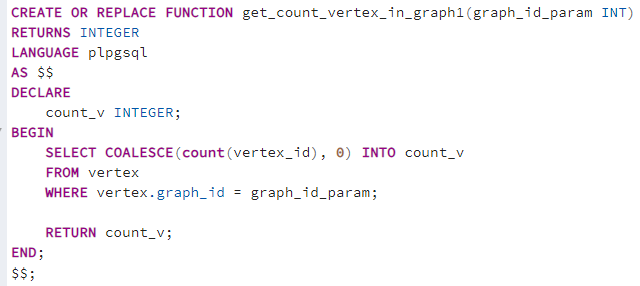
FROM vertex

WHERE vertex.graph\_id = graph\_id\_param;

RETURN count\_v;

END;

$$;



Вызов

select get\_count\_vertex\_in\_graph1(2);

**Д)** Формирование статистики во временной таблице.

Функция определяет и выводит количество вершин в каждом графе, среднее арифметическое количества вершин по всем графам, отклонение в большую или меньшую сторону от среднего значения.

CREATE OR REPLACE function graphs\_statistic3()

RETURNS table(stat\_id int,

graph\_ids INT,

count\_vertexs INT,

avg\_count\_vertexs DOUBLE PRECISION,

diff\_count\_vrtxs DOUBLE PRECISION )

LANGUAGE plpgsql

AS $$

BEGIN

CREATE TEMPORARY TABLE stat\_table (

stat\_id SERIAL PRIMARY KEY,

graph\_id int,

count\_vertexes int,

avg\_count\_vertexes double PRECISION default 0,

diff\_count\_vrtx double PRECISION default 0

);

insert into stat\_table(graph\_id,count\_vertexes)

select graph.graph\_id,coalesce(count(vertex\_id),0) as count\_vertex

from vertex full join graph on graph.graph\_id=vertex.graph\_id

group by graph.graph\_id;

update stat\_table set avg\_count\_vertexes =

(select avg(count\_vertexes) from stat\_table);

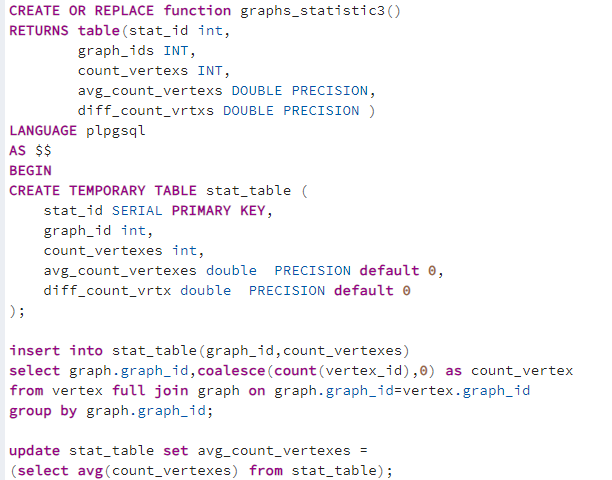
update stat\_table set diff\_count\_vrtx=count\_vertexes-avg\_count\_vertexes;

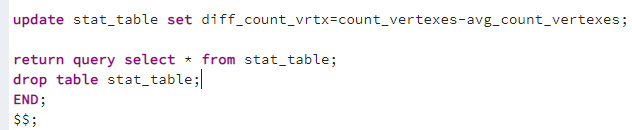
return query select \* from stat\_table;

drop table stat\_table;

END;

$$;



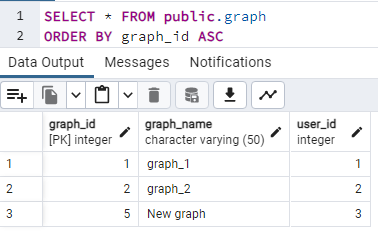


Вызов  
 select \* from graphs\_statistic3();

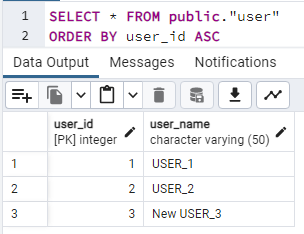
1. **Наборы данных, возвращаемые ХП.**

**А)**

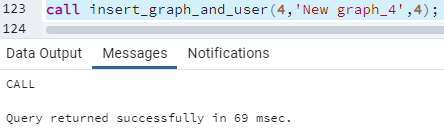
Графы до добавления:



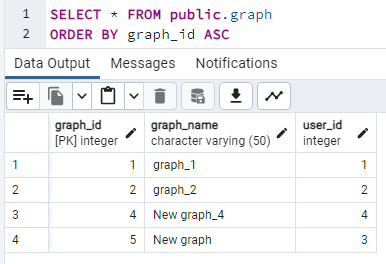
Авторы до добавления:



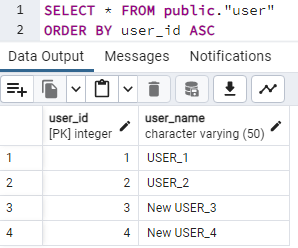
Добавляем новый граф с несуществующим автором:



Графы после добавления:

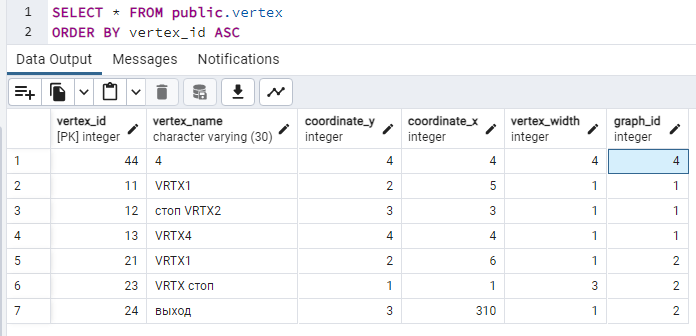


Авторы после добавления:

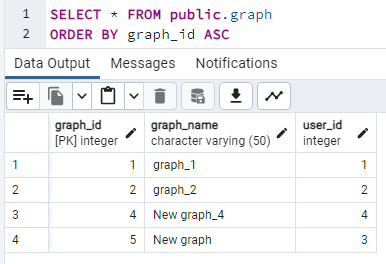


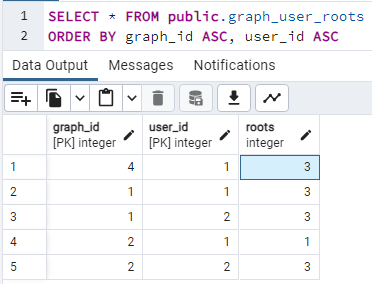
**Б)** Удаление с очисткой справочников

Вершины до удаления:

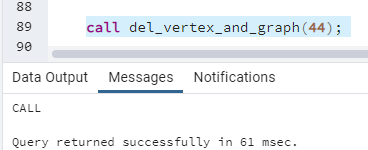


Графы до удаления:

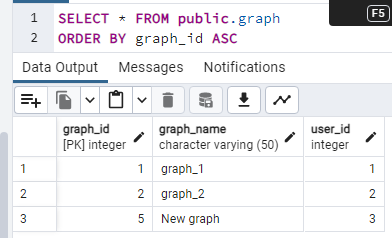


Права пользователей до удаления:  


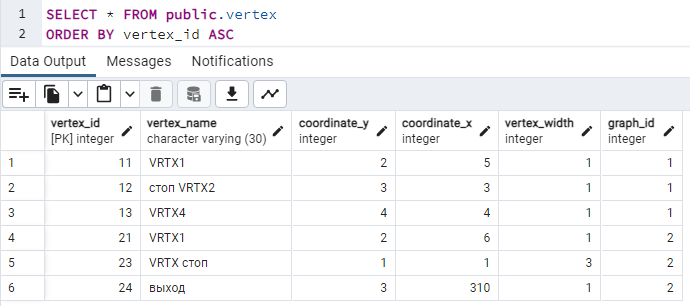
Удалим единственную вершину в графе 4:



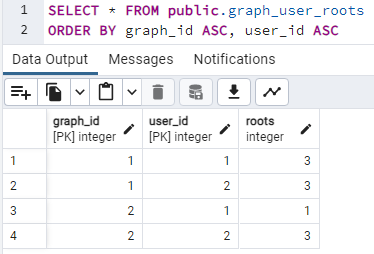
Графы после удаления:



Вершины после удаления:

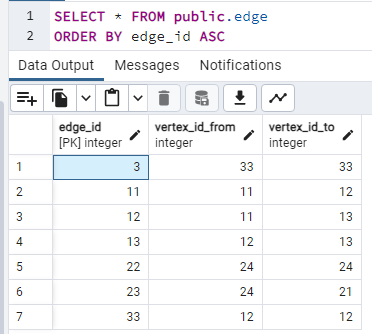


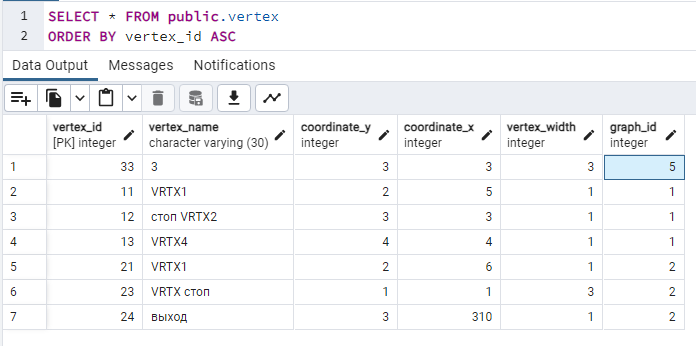
Права пользователей после удаления:



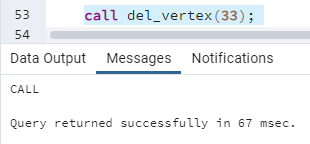
**В)** Каскадное удаление

Ребра до удаления:

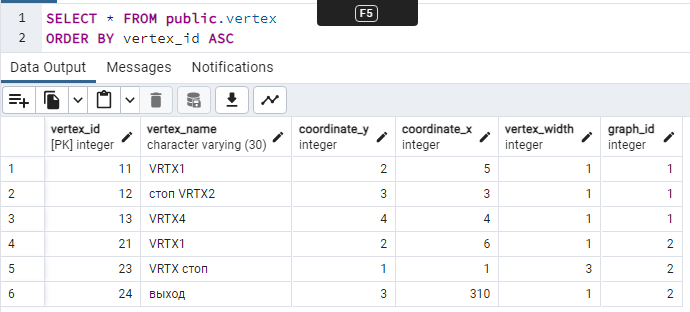


Вершины до удаления:  


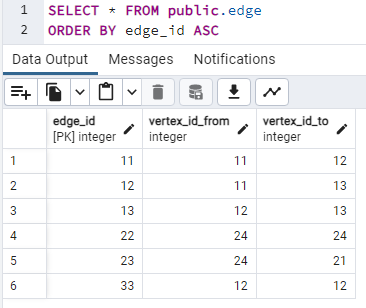
Удаление 33 вершины:



Вершины после удаления:

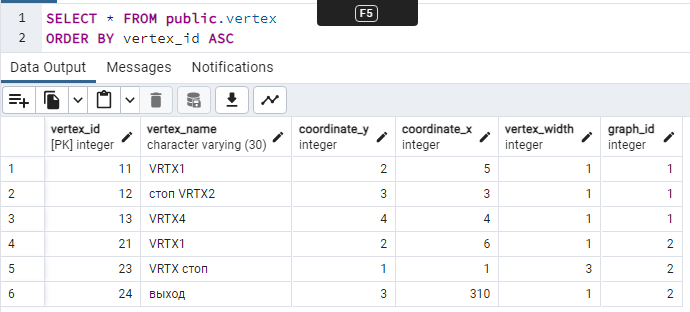


Ребра после удаления:

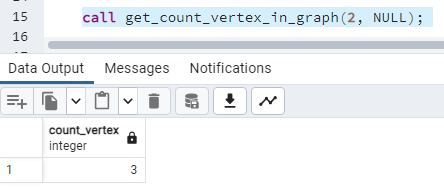


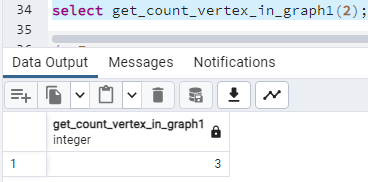
**Г)** Вычисление и возврат значения агрегатной функции

Вершины:



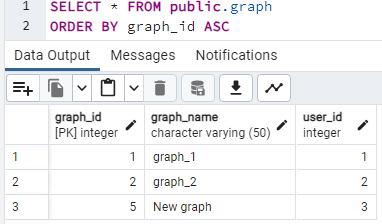
Узнаем сколько вершин в графе с id=2:



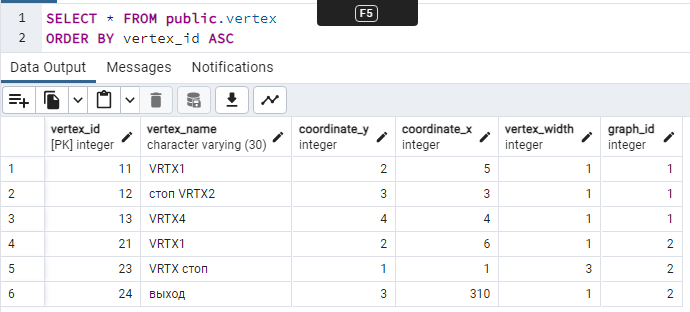


**Д)** Формирование статистики во временной таблице.

Графы:



Вершины:



Проведем статистику по графам:

