1. Создать функцию Period(…, …) c двумя входными параметрами типа date, которая выберет строки из дочерней таблицы в диапазоне дат, указанных первым и вторым аргументами при вызове функции Period(…, …).

Create or replace function Period(

in a date,

in b date

)

returns TABLE(myDate date, concentration int)

as $$

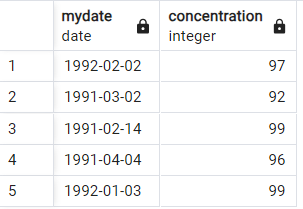
begin

return QUERY(select "Date", "Concentration" from "Acid\_production" where "Acid\_production"."Date" > a

and "Acid\_production"."Date" < b);

end; $$ language plpgsql;

SELECT \* FROM Period('1991-01-03', '1993-01-15')



2. Создать функцию Sum\_object(…)(с одним параметром), которая возвращает список имен объектов из родительской таблицы на основании данных дочерней таблицы. Список объектов определяется значением параметра, исходя из условия, что суммарное количество объектов должно быть больше, чем заданное значение в параметре.

Create or replace function Sum\_obj(

in a int

)

returns TABLE(City character, sum\_con int)

as $$

begin

return QUERY(select "City", (select CAST(sum("Concentration") as int) from "Acid\_production"

where "Cities"."ID\_city" = "Acid\_production"."City") as sum\_conc

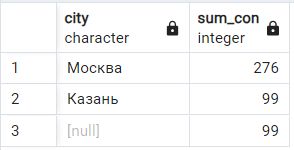
from "Cities"

where (select sum("Concentration") from "Acid\_production"

where "Cities"."ID\_city" = "Acid\_production"."City") > a);

end; $$ language plpgsql;

Select \* From sum\_obj(97)



3. Создать функцию row\_count(…), которая подсчитывает количество строк дочерней таблицы, даты которых находятся между параметрами date\_from и date\_to

Create or replace function my\_row\_count(

in a date,

in b date

)

returns TABLE(data\_count int)

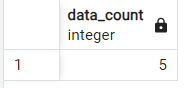
as $$

begin

return QUERY(select COUNT(\*) FROM period(a, b));

end; $$ language plpgsql;

select \* FROM my\_row\_count('1991-01-03', '1993-01-15')



4. Создать хранимую процедуру object\_stat(…), которая подсчитывает минимальное, максимальное и среднее значение объектов в дочерней таблице, входным параметром является Id объекта.

Create or replace function all\_obj\_stat()

returns TABLE(max\_con int, min\_con int, avg\_con int)

as $$

begin

return QUERY(select cast(max("Concentration") as int), cast(min("Concentration") as int), cast(avg("Concentration") as int)

from "Acid\_production");

end; $$ language plpgsql;

SELECT \* FROM all\_obj\_stat()



5. Создать хранимую процедуру objects\_stat(…), которая подсчитывает минимальное, максимальное и среднее значение каждого объекта в дочерней таблице и выводит имя объекта, входным параметром является имя объекта из родительской таблицы.

Create or replace function that\_obj\_stat(

in that\_city character

)

returns TABLE(max\_con int, min\_con int, avg\_con int)

as $$

begin

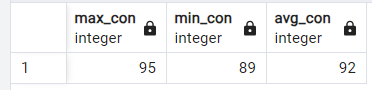
return QUERY(select cast(max("Concentration") as int), cast(min("Concentration") as int), cast(avg("Concentration") as int)

from "Acid\_production", "Cities" where "Cities"."ID\_city" = "Acid\_production"."City"

and "Cities"."City" = that\_city);

end; $$ language plpgsql;

SELECT \* FROM that\_obj\_stat('Москва')



6. Создать хранимую процедуру Itog(…) с одним входным параметром, которая выводит наименование объекта по суммарному количеству объектов: а) Оценка «Незначительный объект», если число объектов меньше 2 б) Оценка «Обычный объект», если число объектов больше 2 и меньше или равно 3 с) Оценка «Значительный объект», если число объектов больше 3.

Create or replace function city\_itog(

in that\_city character

)

returns character

as $$

begin

if(CAST((select count(\*)

from "Acid\_production", "Cities" where "Cities"."ID\_city" = "Acid\_production"."City"

and "Cities"."City" = that\_city) as int) <= 2)

then return 'minor city';

end if;

if(CAST((select count(\*)

from "Acid\_production", "Cities" where "Cities"."ID\_city" = "Acid\_production"."City"

and "Cities"."City" = that\_city) as int) = 3)

then return 'normal city';

end if;

if(CAST((select count(\*)

from "Acid\_production", "Cities" where "Cities"."ID\_city" = "Acid\_production"."City"

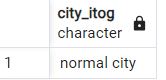
and "Cities"."City" = that\_city) as int) > 3)

then return 'significant city';

end if;

end; $$ language plpgsql;

SELECT \* FROM city\_itog('Москва')



2.1 Создать триггер After\_Delete, который при удалении записи из родительской таблицы удалял бы все связанные записи из дочерней таблицы. Показать результат работы триггера.

CREATE FUNCTION alter\_del() RETURNS trigger AS '

BEGIN

delete from "Acid\_production" where "Acid\_production"."City" = OLD."ID\_city";

return OLD;

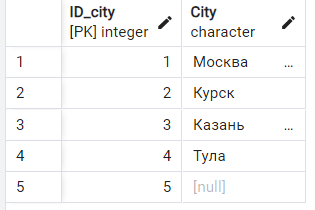
END;

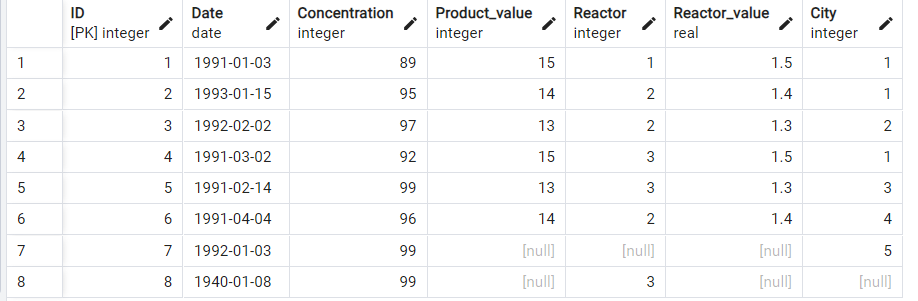
' LANGUAGE plpgsql;

CREATE TRIGGER alterdel\_tr

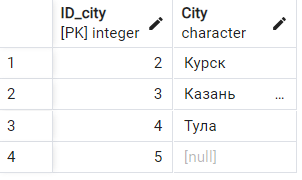
BEFORE DELETE ON "Cities" FOR EACH ROW

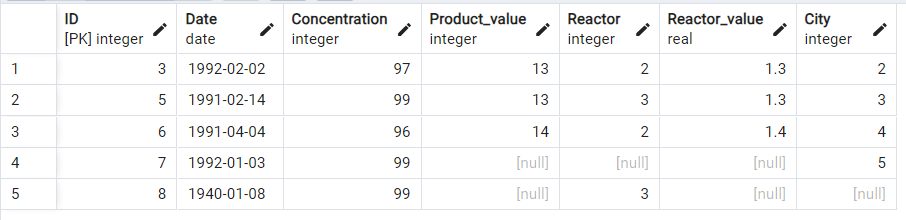
EXECUTE PROCEDURE alter\_del();





delete from "Cities" where "City" = 'Москва'





2.2 Создать триггер Before\_Delete, который при удалении записи из дочерней таблицы выводил бы имя объекта родительской таблицы. Показать результат работы триггера.

CREATE FUNCTION before\_del () RETURNS trigger AS '

declare parent\_name text;

BEGIN

select "City" into parent\_name from "Cities" where "Cities"."ID\_city" = OLD."City";

raise notice 'Deleting %', parent\_name;

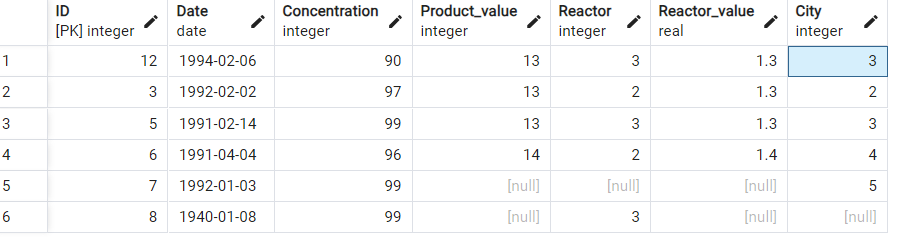
return OLD;

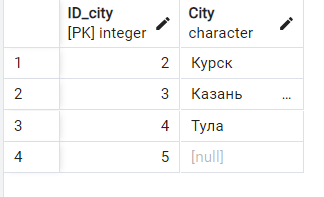
END;' LANGUAGE plpgsql;

CREATE TRIGGER befordel\_tr

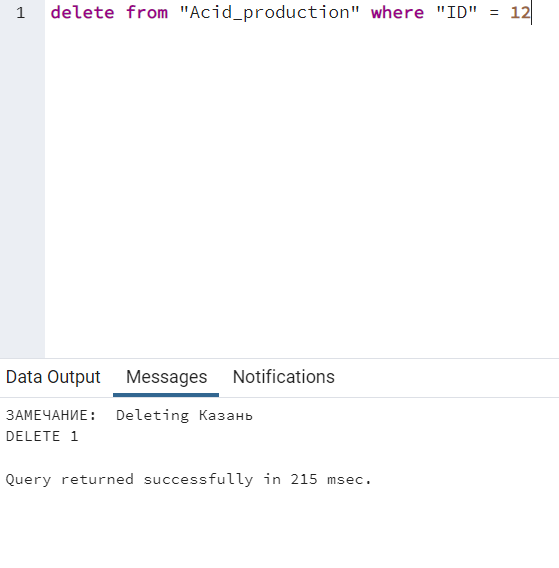
BEFORE DELETE ON "Acid\_production" FOR EACH ROW

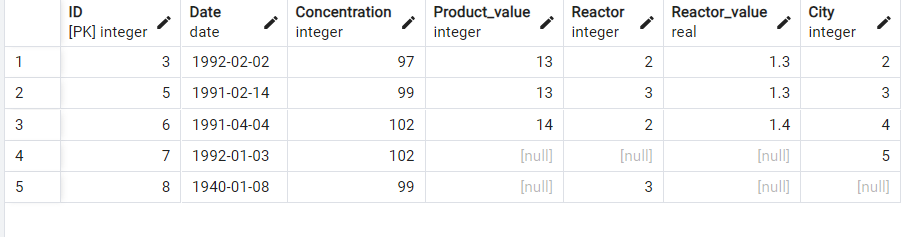
EXECUTE PROCEDURE before\_del();





delete from "Acid\_production" where "ID" = 12





2.3 Создать триггер ins\_sum, который связывает триггер с таблицей для инструкций INSERT. Это действует как сумматор, чтобы суммировать значения, вставленные в один из столбцов дочерней таблицы. Триггер должен активироваться перед каждой строкой, вставленной в таблицу. Показать результат работы триггера.

Вывод суммы всех записей + одна новая.

CREATE FUNCTION before\_del () RETURNS trigger AS '

declare parent\_name int;

BEGIN

raise notice 'inserted %', COUNT(\*) + 1 From "Acid\_production";

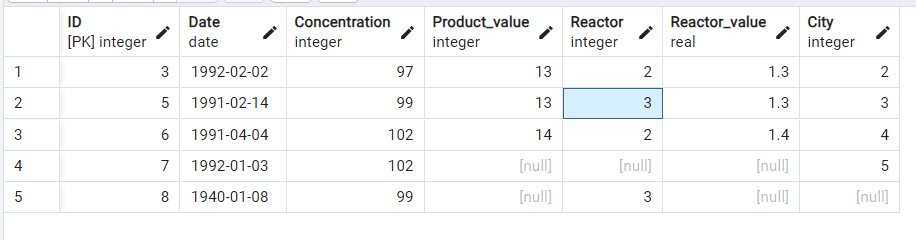
return NEW;

END;' LANGUAGE plpgsql;

CREATE TRIGGER int\_sum\_tr

BEFORE INSERT ON "Acid\_production" FOR EACH ROW

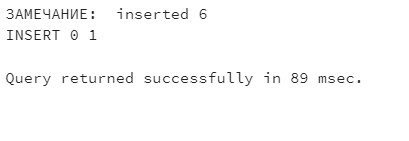
EXECUTE PROCEDURE int\_sum();

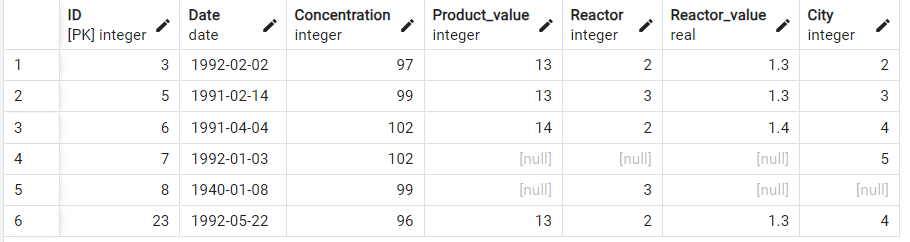


insert into

"Acid\_production"("ID", "Date", "Concentration", "Product\_value", "Reactor", "Reactor\_value", "City") VALUES

(default, '1992-05-22', 96, 13, 2, 1.3, 4)





2.4 Создать триггер Before\_Update\_Value на событие UPDATE, который увеличивает значение числового поля дочерней таблицы на 10%. Показать результат работы триггера.

CREATE FUNCTION before\_update\_val() RETURNS trigger AS '

BEGIN

NEW."Concentration" = Cast(NEW."Concentration"\*(1.1) as int);

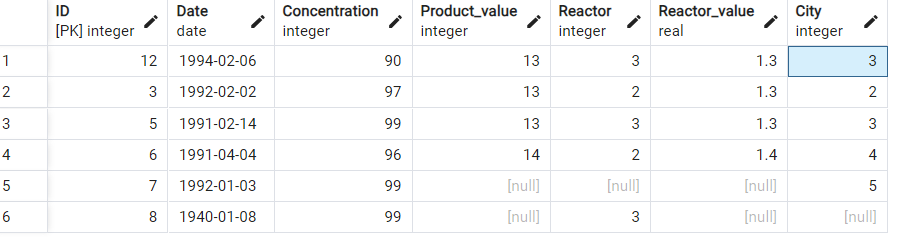
END;

' LANGUAGE plpgsql;

CREATE TRIGGER beforeupdate\_tr

BEFORE UPDATE ON "Acid\_production" FOR EACH ROW

EXECUTE PROCEDURE before\_update\_val();



UPDATE "Acid\_production" SET "Concentration" = 93 WHERE "City" = 4;

UPDATE "Acid\_production" SET "Concentration" = 93 WHERE "City" = 5;

