ГУАП

КАФЕДРА № 43

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		Н.В Путилова
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №8

Обеспечение активной целостности данных базы данных

по дисциплине: Проектирование баз данных

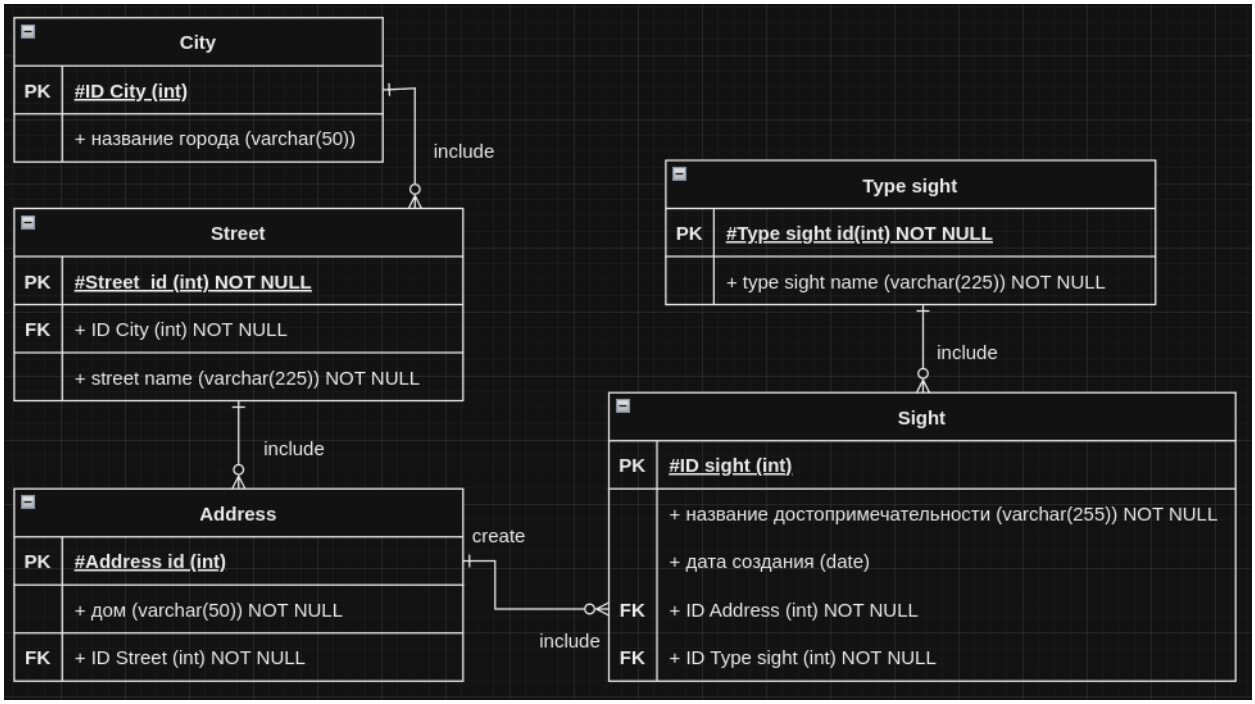
РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР.	4134к		Столяров Н.С.
		подпись, дата	инициалы, фамилия

Санкт-Петербург
2023

Задание

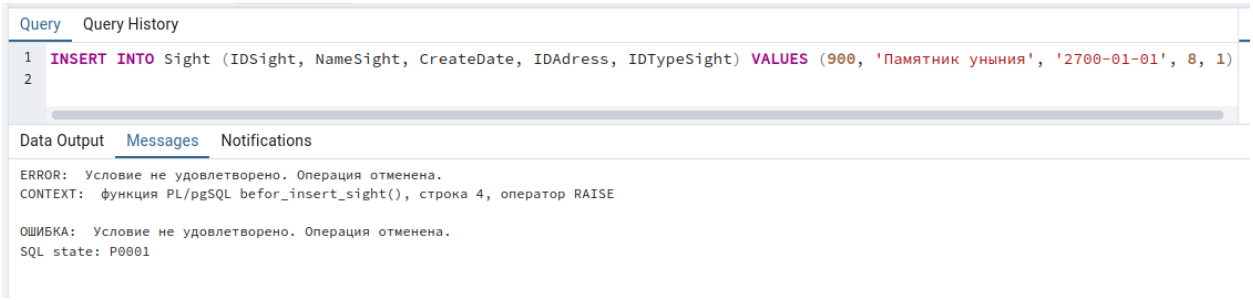
Реализовать для своей базы данных триггеры для всех событий (insert,delete, update) до и после.(6 триггеров) Часть из которых будет обеспечивать ссылочную целостность, остальные могут иметь другое назначение из других предложенных , но не менее 2 различных
(- Вычисление/поддержание в актуальном состоянии вычислимых (производных) атрибутов (полей);
- логирование (запись) изменений;
- обеспечения безопасности данных;
- логическое (мягкое) удаление данных
- проверка корректности проводимых действий.).
Вычислимые поля можно добавить при необходимости.



Before insert

```
CREATE OR REPLACE FUNCTION befor_insert_sight()

RETURNS TRIGGER AS $$

BEGIN

    if new.createdate > current_date
```

```
 then RAISE EXCEPTION 'Условие не удовлетворено. Операция отменена.';
 end if;
   RETURN new;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER befor_insert_sight
before insert ON sight
FOR EACH ROW
EXECUTE FUNCTION befor_insert_sight();
```

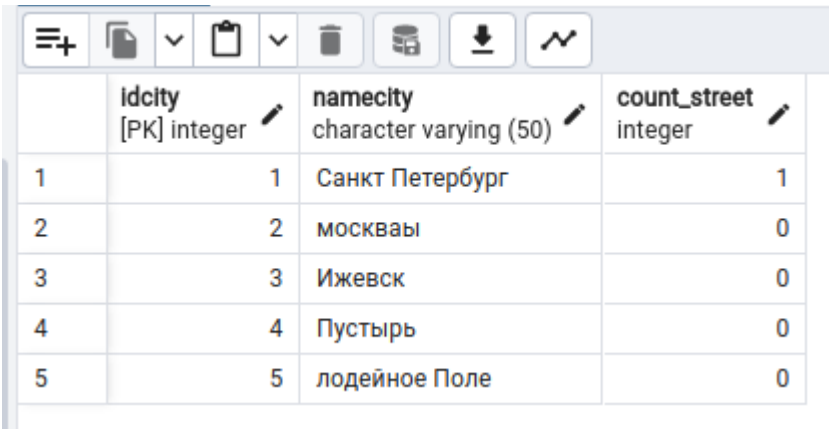


After delete

```
CREATE OR REPLACE FUNCTION after_del_street()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE city
   SET count_street = count_street - 1
   where idcity = old.idcity;
   RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER after_del_street

after delete ON street

FOR EACH ROW

EXECUTE FUNCTION after_del_street();
```

Для примера удалил улицу из «лодейное поле»



	idcity [PK] integer	namecity character varying (50)	count_street integer
1	1	Санкт Петербург	1
2	2	москваы	0
3	3	Ижевск	0
4	4	Пустырь	0
5	5	лодейное Поле	0

Before delete

```
-- удаление доспремечательностей если удаляем их тип

CREATE OR REPLACE FUNCTION befor_del_typesight()

RETURNS TRIGGER AS $$

BEGIN

    delete from sight

  where idtypesight = old.idtypesight;

    RETURN NULL;

END;

$$ LANGUAGE plpgsql;


CREATE TRIGGER befor_del_typesight

before delete ON typesight

FOR EACH ROW
```

```
EXECUTE FUNCTION befor_del_typesight();
```

Для примера удлил тип id 3 (памятник)

	idsight [PK] integer	namesight character varying (255)	createdate date	idadress integer	idtypesight integer
1	1	Музей Фаберже	2013-01-01	2	1
2	2	Музей-заповедник "Ораниенбаум"	1711-01-01	3	1
3	3	Мариинский театр	1860-01-01	1	2
4	4	Царицыно Музей-Заповедник	1984-01-01	4	1
5	5	Московский Государственный Объединенный Музей-Заповедник "Коломенск...	2005-01-01	5	1
6	6	Большой театр	1776-01-01	6	2
7	7	Музейно-выставочный комплекс стрелкового оружия им. М.Т. Калашникова	2004-01-01	7	1
8	9	Национальный Музей Удмуртской Республики	1920-01-01	9	1
9	10	Музей веселья	2077-01-01	8	1
10	12	какой-то дворец	2033-01-01	2	1
11	13	Архитектурный комплекс	1389-01-01	1	2
12	14	Музей страданий	2077-01-01	8	1
13	100	Памятник уныния	2000-01-01	8	1
14	110	Памятник уныния	2001-01-01	8	1

Before update

```
-- для бэкапа
CREATE TABLE City_backup (
    id_City_backup serial,
    IDCity INT,
    NameCity VARCHAR(50),
    count_street int
);


-- бэкап
CREATE OR REPLACE FUNCTION befor_update_City()
RETURNS TRIGGER AS $$
BEGIN
    INSERT INTO City_backup (IDCity,NameCity,count_street)
    values (old.IDCity,old.NameCity, old.count_street);
    RETURN new;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER befor_update_City
before update ON City
FOR EACH ROW
EXECUTE FUNCTION befor_update_City();
```

	id_city_backup integer	idcity integer	namecity character varying (50)	count_street integer
1	1	2	Москва	0
2	2	1	Санкт Петербург	0
3	3	1	Санкт Петербург	1
4	4	5	Поле	0
5	5	5	лодейное Поле	0
6	6	2	москваы	0
7	7	5	лодейное Поле	1
8	8	2	москваы	1
9	9	5	лодейное Поле	0
10	10	1	Санкт Петербург	2

After update

```
-- обновляет счётсчик (если улица перемистится в другой город)
CREATE OR REPLACE FUNCTION after_update_street()
RETURNS TRIGGER AS $$
BEGIN
    UPDATE city
    SET count_street = count_street + 1
    where idcity = new.idcity;

  UPDATE city
    SET count_street = count_street - 1
    where idcity = old.idcity;
```

```
   RETURN NULL;
END;
$$ LANGUAGE plpgsql;


CREATE TRIGGER after_update_street
after update ON street
FOR EACH ROW
EXECUTE FUNCTION after_update_street();
```

Для примера перенесём улицу из «лодейное поле» в Санкт Петербург

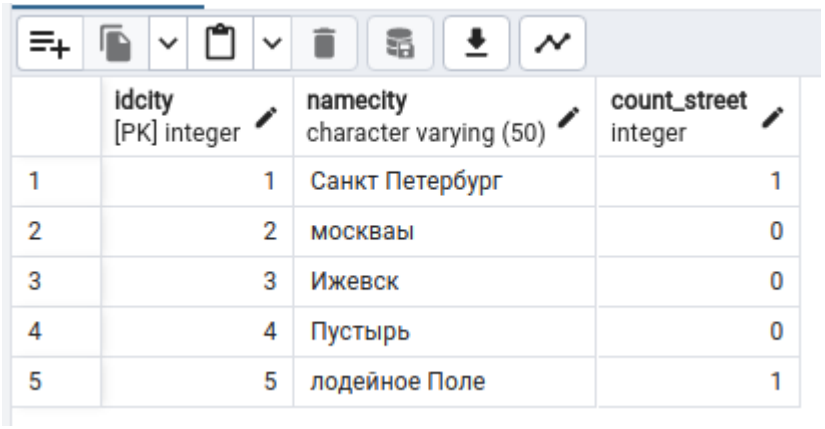
	idcity [PK] integer	namecity character varying (50)	count_street integer
1	1	Санкт Петербург	2
2	2	москваы	0
3	3	Ижевск	0
4	4	Пустырь	0
5	5	лодейное Поле	0

After insert

```
-- прибавляет
CREATE OR REPLACE FUNCTION after_insert_street()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE city
   SET count_street = count_street + 1
   where idcity = new.idcity;
   RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER after_insert_street

after insert ON street

FOR EACH ROW

EXECUTE FUNCTION after_insert_street();
```

Для примера добавил улицу в лодейное поле



	idcity [PK] integer	namecity character varying (50)	count_street integer
1	1	Санкт Петербург	1
2	2	москваы	0
3	3	Ижевск	0
4	4	Пустырь	0
5	5	лодейное Поле	1