

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені Ігоря Сікорського»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №1**

з дисципліни **Бази даних і засоби управління**

*на тему: “Проектування бази даних та ознайомлення з базовими операціями СУБД PostgreSQL”*

Виконала:

студентка ІII курсу

групи КВ-92

Кузьмич А. А.

Перевірив:

Павловский В. И.

Київ – 2021

*Метою роботи* є здобуття вмінь проектування бази даних та практичних навичок створення реляційних баз даних за допомогою PostgreSQL.

*Завдання* роботи полягає у наступному:

1. Розробити модель «сутність-зв’язок» предметної галузі, обраної студентом самостійно, відповідно до пункту «Вимоги до ER-моделі».
2. Перетворити розроблену модель у схему бази даних (таблиці) PostgreSQL.
3. Виконати нормалізацію схеми бази даних до третьої нормальної форми (3НФ).
4. Ознайомитись із інструментарієм PostgreSQL та pgAdmin 4 та внести декілька рядків даних у кожну з таблиць засобами pgAdmin 4.

**Модель «сутність-зв’язок» галузі лікарня**

Обрана предметна галузь передбачає отримання і обробку потрібної інформації щодо лікарні. Згідно цієї області для побудови бази даних було виділено наступні сутності:

**Залишились зауваження по опису сутностей**

Лікарня з атрибутами : назва лікарні, телефон приймального покою, адреса лікарні. Призначення: збереження даних, що стосуються тих лікарень, які цікавлять користувача.

Лікар з атрибутами: код лікаря НЗУ, лікарня, де працює лікар, код спеціальності лікаря, ПІБ лікаря та телефон. Призначення: збереження даних щодо лікарів.

Пацієнт з атрибутами: номер медичної картки, ПІБ. Призначення: збереження даних щодо пацієнтів.

Направлення з атрибутами: номер направлення, код лікаря що видав направлення, код спеціальності, номер медичної картки пацієнта, дані обстеження. Призначення: збереження інформації стосовно обстежень пацієнтів та роботи лікарів.

**Абзаци**

Спеціалізація з атрибутами: код спеціальності, кабінет, що закріплений за спеціальністю, спеціальність. Призначення: : збереження інформації стосовно спеціальностей лікарів, що представлені в лікарні і їх зв'язок з кабінетами.

**Опис зв’язків**

Лікар може працювати в декількох лікарнях, а в одній лікарні обов’язково має бути декілька лікарів. При чому лікар може працювати і тільки в одній лікарні, а може бути звільненим, але ще знаходитись в базі даних. Тому між сутностями Лікар і Лікарня існує зв’язок М:N.

Лікар обов’язково має Спеціалізацію, а от Лікарів однієї Спеціалізації може бути декілька. Якщо спеціалізація є в базі даних, то хоча б один лікар має бути за нею закріплений. . Тому між сутностями Спеціалізація і Лікар існує зв’язок 1:N.

Лікар може видати декілька направлень, а от направлення видається тільки одним лікарем. . Тому між сутностями Лікар і Направлення існує зв’язок 1:N.

Направлення містить тільки одну Спеціалізацію, тобто направлення тільки до одного спеціаліста, а от Спеціалізація може мати декілька направлень, тобто до одного спеціаліста направляють декілька Направлень. Тому між сутностями Спеціалізація і Направлення існує зв’язок 1:N.

Направлення видається тільки на одного Пацієнта, а от у Пацієнта може бути декілька направлень. Тому між сутностями Пацієнт і Направлення існує зв’язок 1:N.

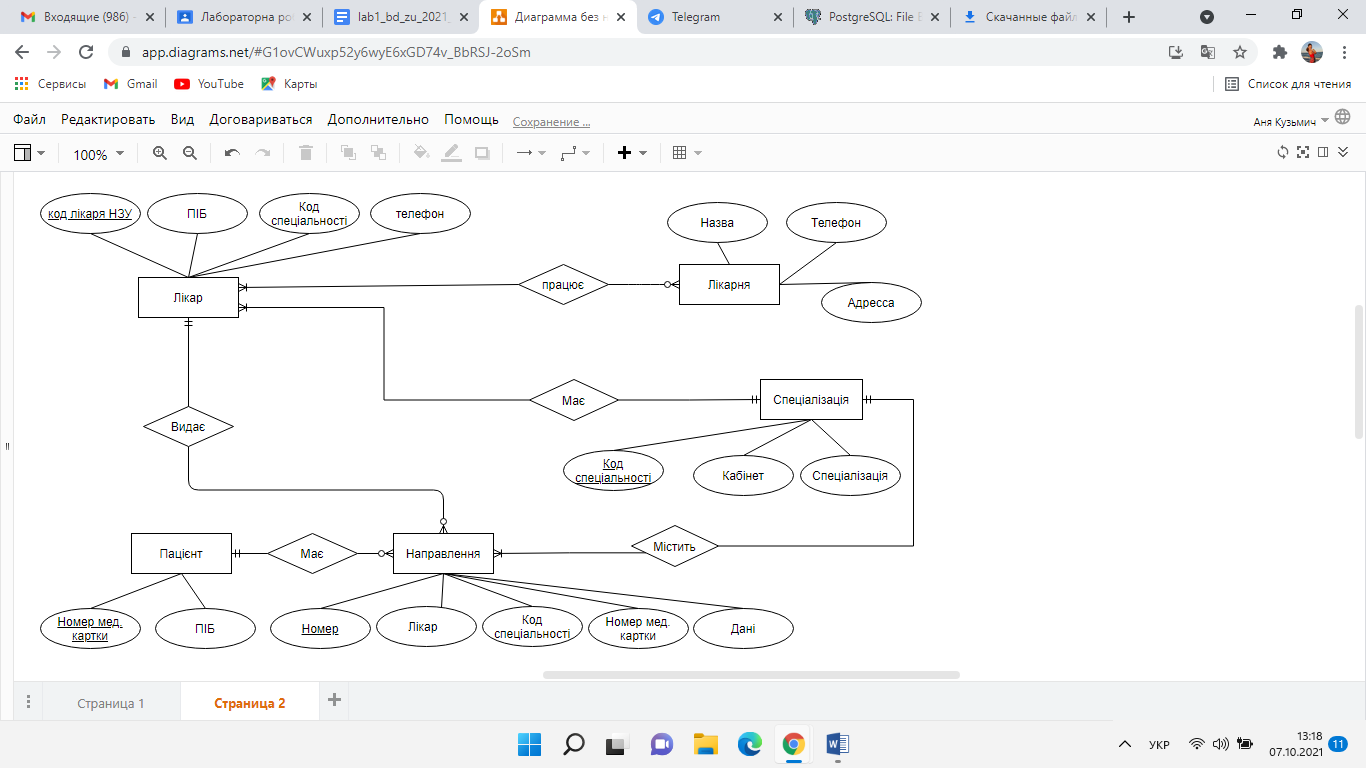


Рисунок 1. ER-діаграма, побудована за нотацією Чена

**Перетворення моделі у схему бази даних**

В схемі бази даних неможливий зв'язок М:N, а він у нас існує між сутностями Лікар і Лікарня. Щоб це уникнути було введено таблицю Лікар\_ Лікарня.

Також назви було перекладено англійською, для зручності наведено таблицю нижче.

|  |  |  |
| --- | --- | --- |
| Сутність | Атрибут | Тип атрибуту |
| **Hospital** - *містить дані про лікарню* | **id** *- унікальний ідентифікатор лікарні*  **name** *- назва лікарні*  **phone** *– телефонний номер лікарні*  **address** *- фізична адреса лікарні* | **integer** *(числовий)*  **character varying** *(рядок)*  **integer** *(числовий)*  **character varying** *(рядок)* |
| **Doctor** - *містить у собі всі дані лікарів* | **id\_doctor** *- унікальний ідентифікатор лікаря*  **name** *– ПІБ лікаря*  **id\_specialist** *- ідентифікатор спеціальності*  **phone** *- телефонний номер лікаря* | **integer** *(числовий)*  **character varying** *(рядок)*  **integer** *(числовий)*  **integer** *(числовий)* |
| **Patient** - *зберігає дані про пацієнтів* | **Number\_med** *- унікальний номер медичної картки*  **Name\_p** *– ПІБ пацієнта* | **integer** *(числовий)*  **character varying** *(рядок)* |
| **Direction** - *містить інформацію про обстеження пацієнтів.* | **number** *- унікальний ідентифікатор направлення*  **Number\_med** *- унікальний номер медичної картки*  **id\_doctor** *- унікальний ідентифікатор лікаря*  **id\_specialist -** *унікальний ідентифікатор спеціальності*  **data** *– дані обстеження* | **integer** *(числовий)*  **integer** *(числовий)*  **integer** *(числовий)*  **integer** *(числовий)*  **character varying** *(рядок* |
| **Specialist** - *містить інформацію про спеціальності лікарів* | **id\_specialist -** *унікальний ідентифікатор спеціальності*  **cabinet –** *номер кабінету*  **specialization –** *назва спеціальності* | **integer** *(числовий)*  **integer** *(числовий)*  **character varying** *(рядок* |
| **Hospital\_ Doctor**  *містить інформацію про відповідність лікаря і лікарні, де він працює.* | **id\_tab - -** *унікальний ідентифікатор таблиці відповідності*  **id\_doctor***- унікальний ідентифікатор лікаря*  **id***- унікальний ідентифікатор лікарні* | **integer** *(числовий)*  **integer** *(числовий)*  **integer** *(числовий)* |

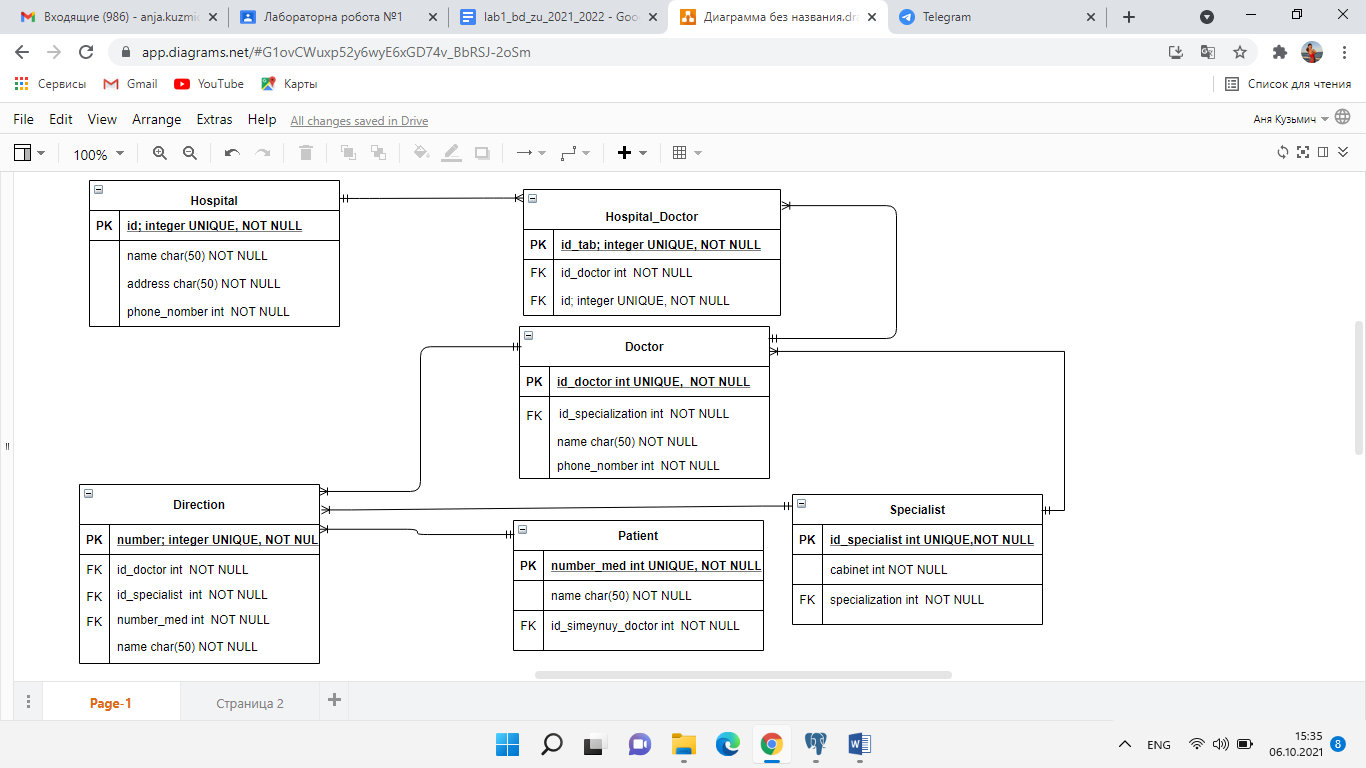


Рисунок 2. Схема бази даних

**Функціональні залежності**

**Hospital**:

**id** *- унікальний ідентифікатор лікарні*

**name** *- назва лікарні*

**phone** *– телефонний номер лікарні*

**address** *- фізична адреса лікарні*

**id** → name, address, phone

id → name (назва конкретної лікарні залежить від її коду)

id → address (адреса млікарні зв’язана з її унікальним кодом)

id → phone (номер лікарні пов'язаний з її кодом)

address → name

name → address

id → address → name (транзитивне відношення, але воно включає в себе ключовий атрибут)

id → name → address (транзитивне відношення, але воно включає в себе ключовий атрибут)

**Doctor**:

**id\_doctor** *- унікальний ідентифікатор лікаря*

**name** *– ПІБ лікаря*

**id\_specialist** *- ідентифікатор спеціальності*

**phone** *- телефонний номер лікаря*

id\_ doctor → name, id\_specialist, phone

id\_ doctor → name (Ім’я залежить від коду лікаря)

id\_ doctor → phone (номер залежить від коду лікаря)

Є декілька транзитивних відношень, але вони включають в себе ключовий атрибут.

**Patient**:

**Number\_med** *- унікальний номер медичної картки*

**Name\_p** *– ПІБ пацієнта*

id\_order → Number\_med, Name\_p

Number\_med → Name\_p (Ім’я пацієнта пов’язано з його унікальним номером медичної картки)

**Specialist:**

**id\_specialist -** *унікальний ідентифікатор спеціальності*

**cabinet –** *номер кабінету*

**specialization –** *назва спеціальності*

id\_specialist → cabinet, specialization

id\_specialist → cabinet (Кабінет спеціаліста залежить від коду спеціализації)

id\_specialist → specialization (спеціалізація залежить від коду)

Є декілька транзитивних відношень, але вони включають в себе ключовий атрибут.

**Direction**:

**number** *- унікальний ідентифікатор направлення*

**data** – *дані обстеження*

number → data **(**дані в направленні залежать від номера направлення)

Схема відповідає 1НФ,

1. В таблиці немає дубльованих рядків.

2. В кожній комірці зберігається атомарне (скалярне) значення.

3. В кожному стовпці зберігаються дані одного типу.

Схема відповідає 2НФ, тому що:

1. Вона відповідає 1НФ.

2. Має первинний ключ, а всі не ключові стовпці таблиці залежать від первинного ключа.

Схема відповідає 3НФ, тому що:

1. Вона відповідає 2НФ.

2. Всі не ключові атрибути таблиці залежать винятково від усього первинного ключа, а не його частини. Тобто кожен неключових атрибут нетранзитивно залежить від первинного ключа.

Що наведено у функціональних залежностях вище.

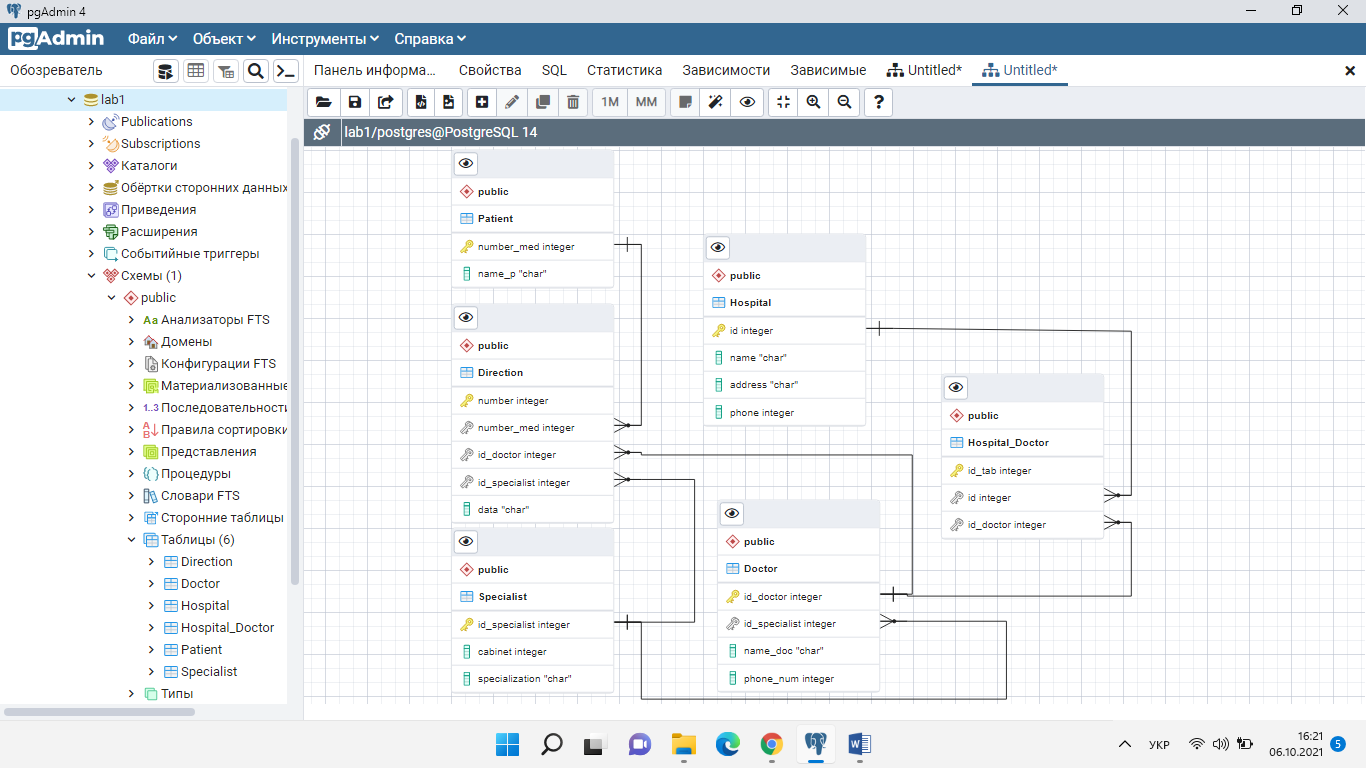
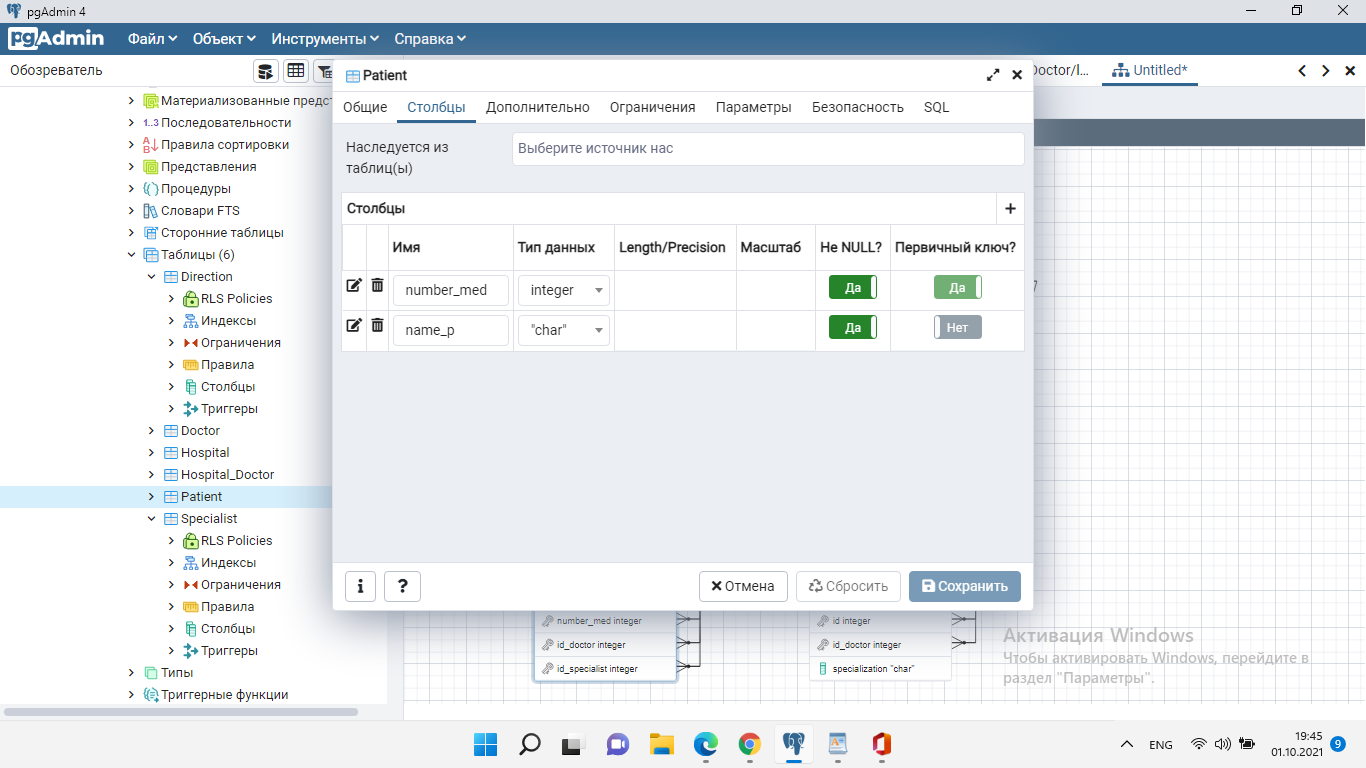
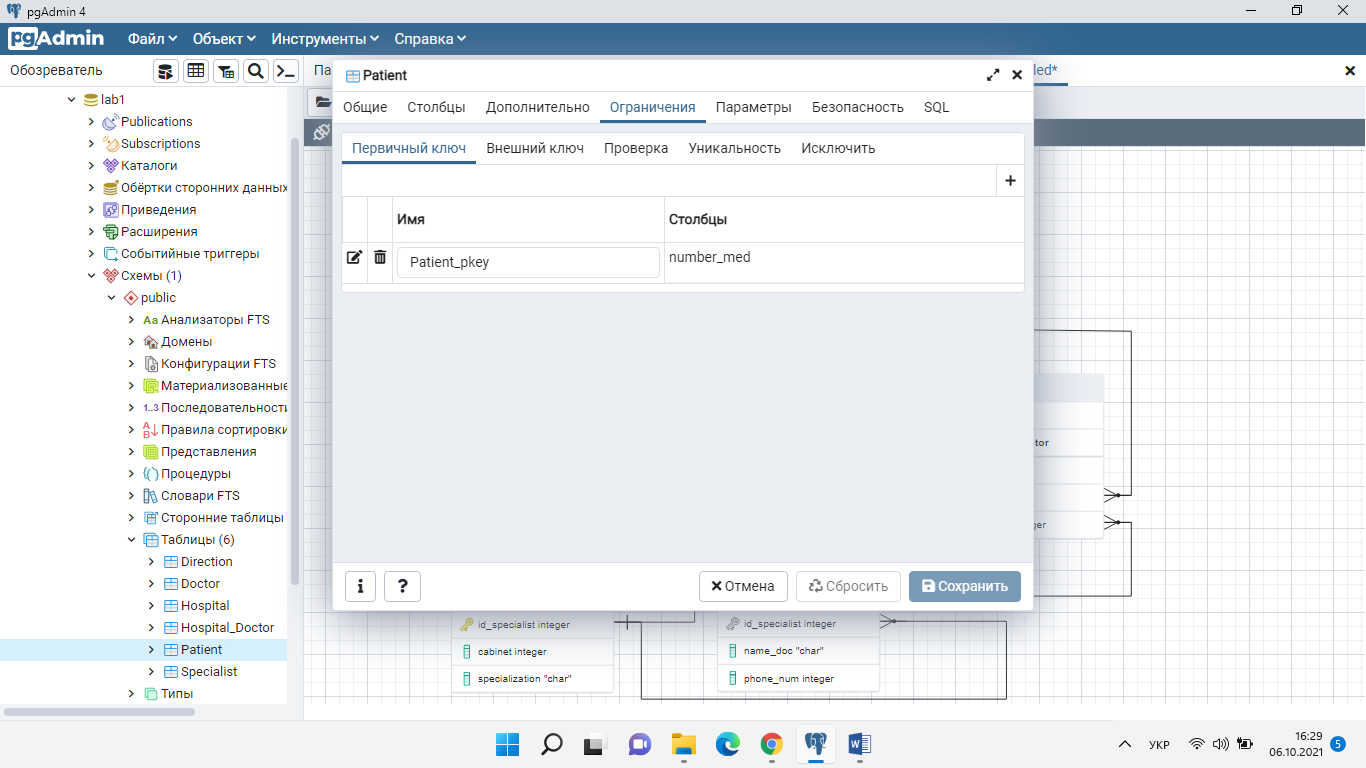


Рисунок 3. Схема бази даних у pgAdmin 4

**Таблиці бази даних у pgAdmin4**

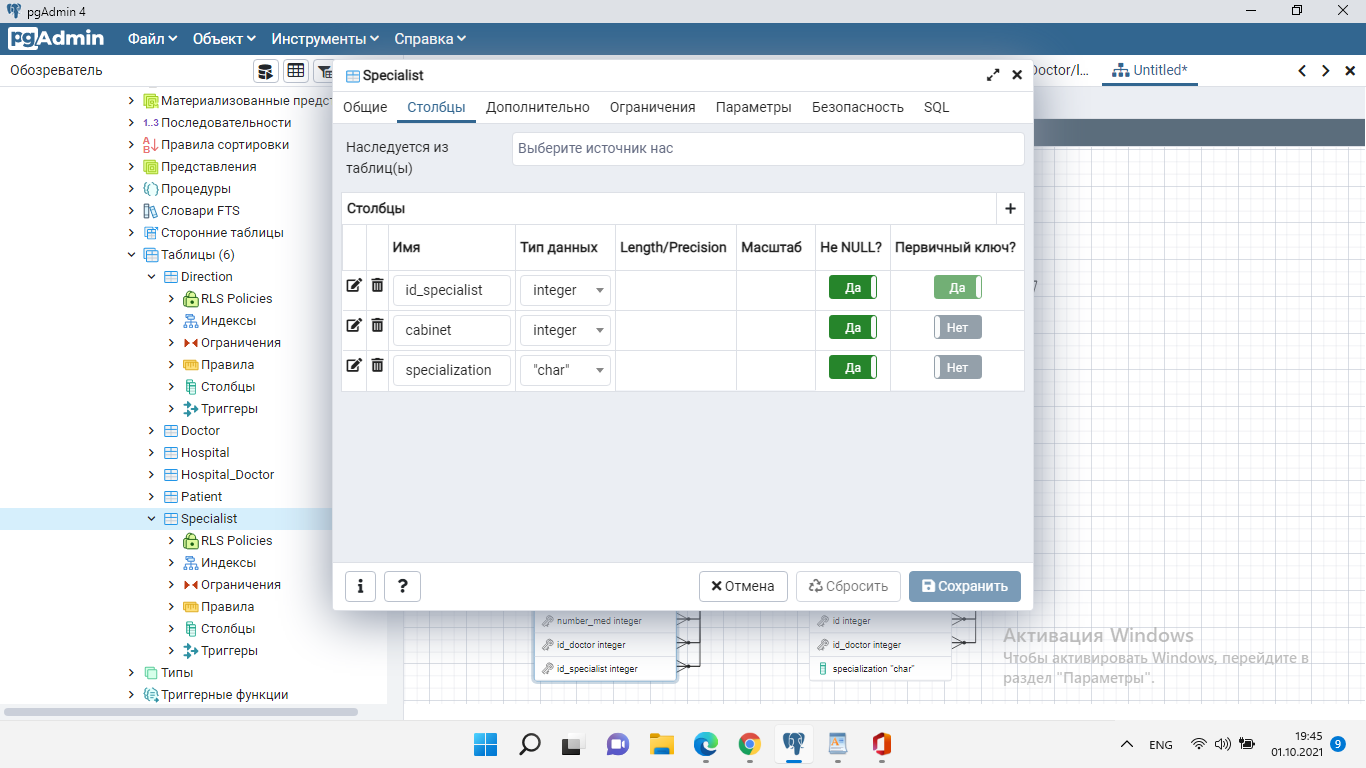
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Pacient\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

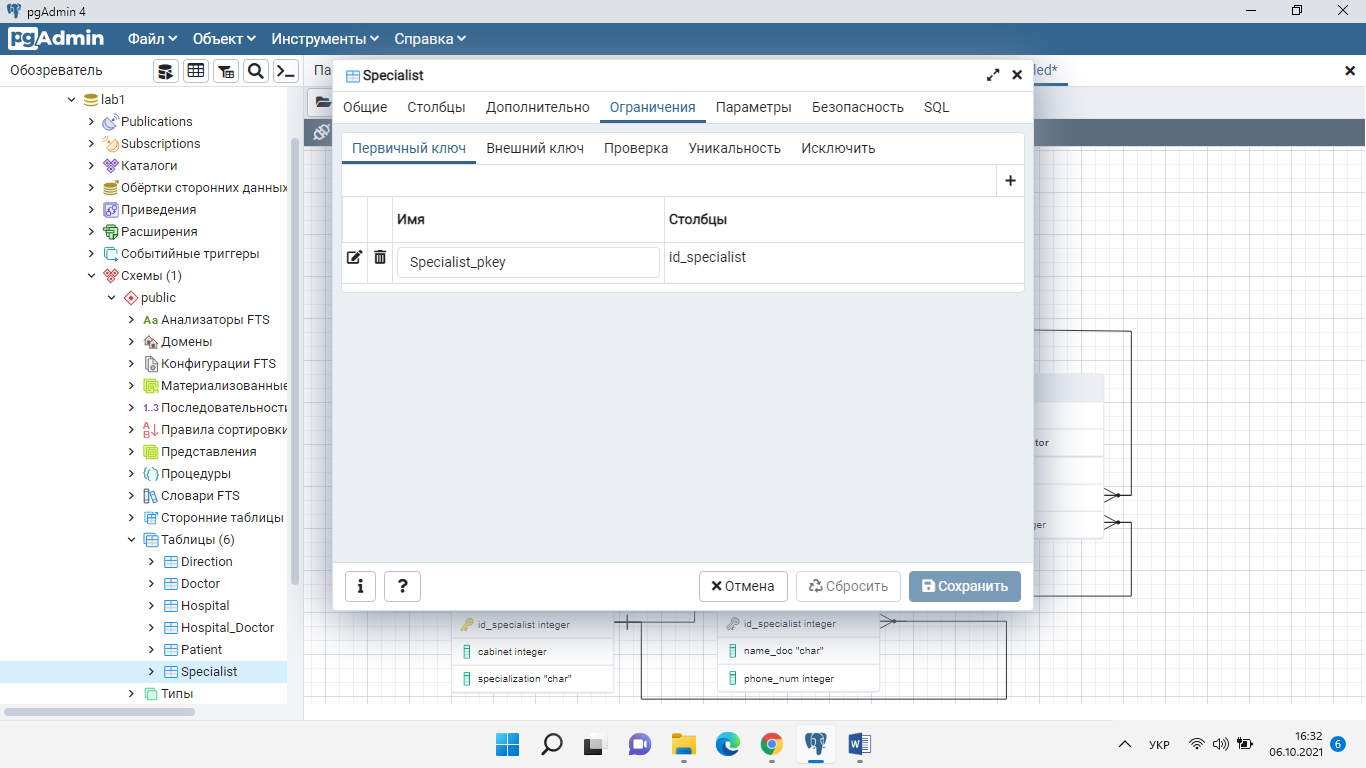




У цієї таблиці немає зовнішніх ключів (FK)

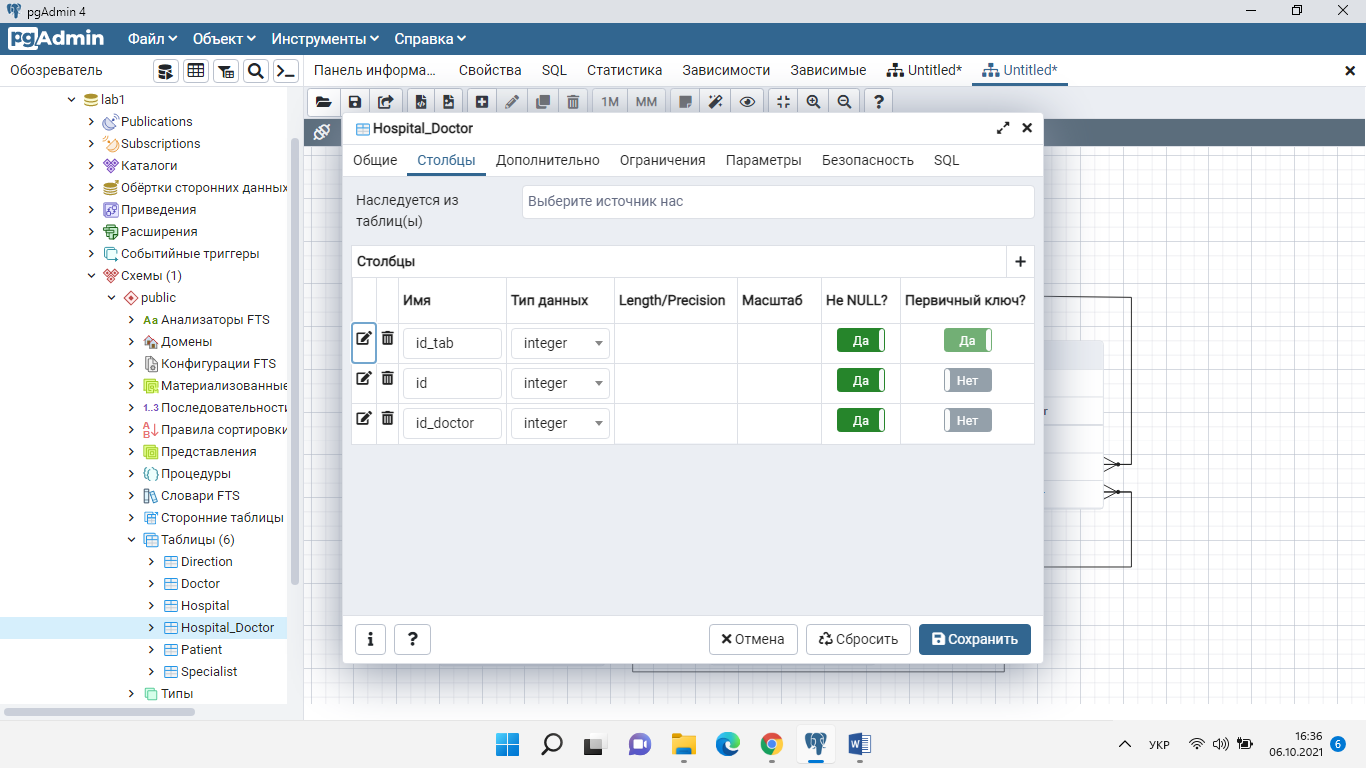
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Specialist\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

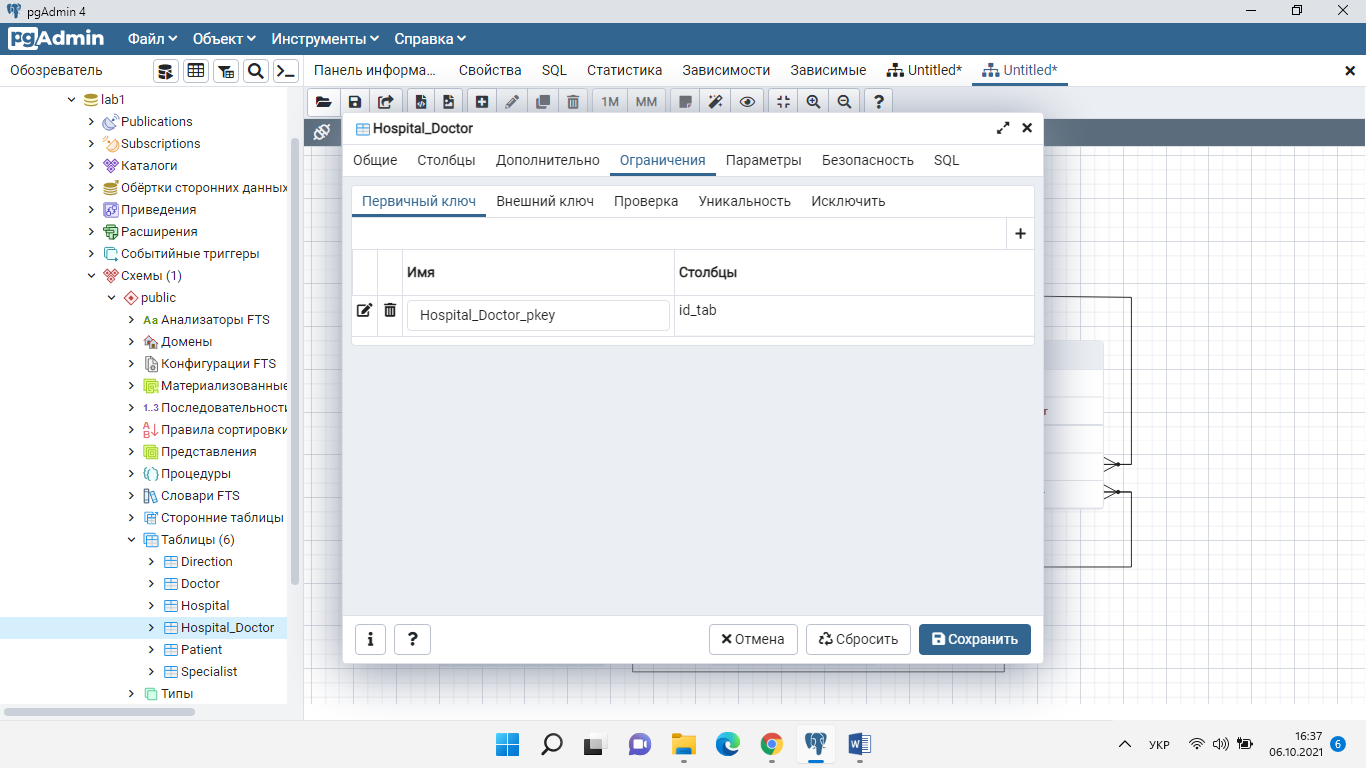


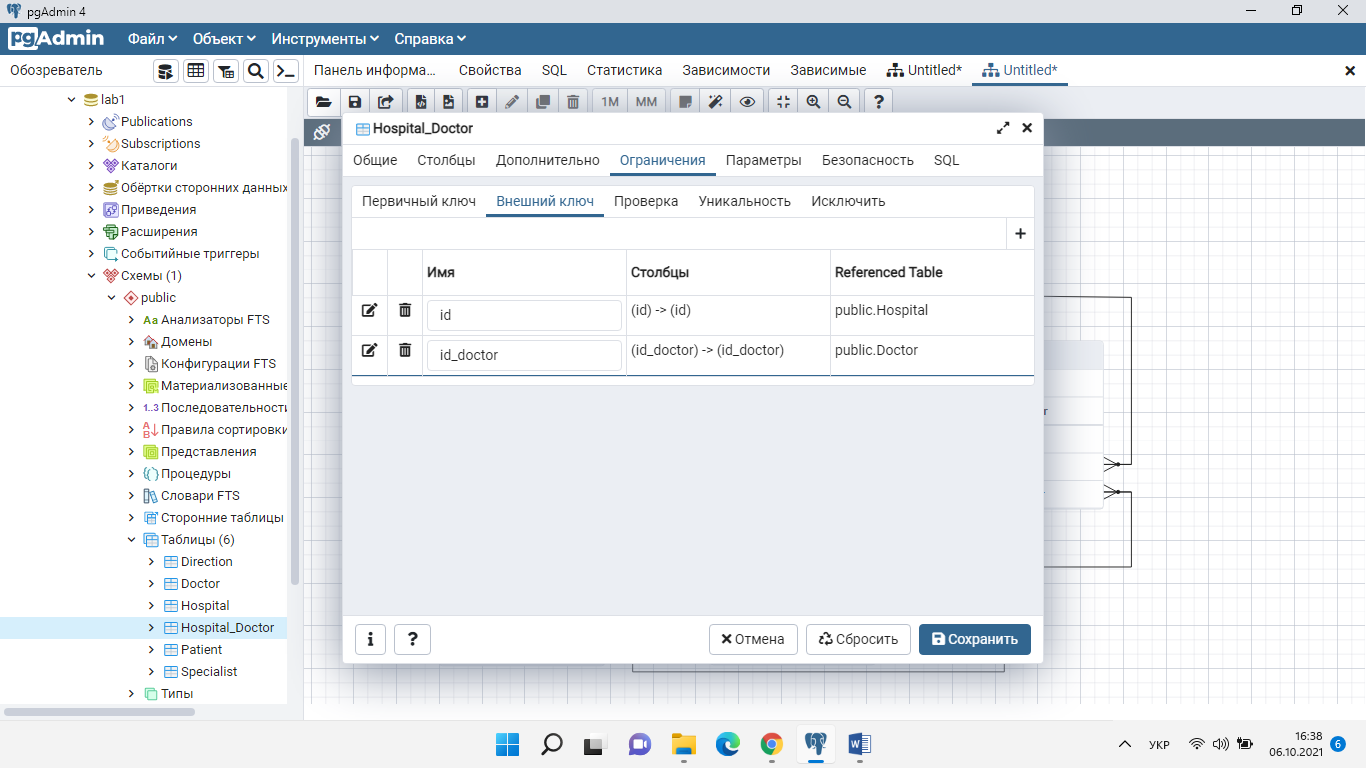


У цієї таблиці немає зовнішніх ключів (FK)

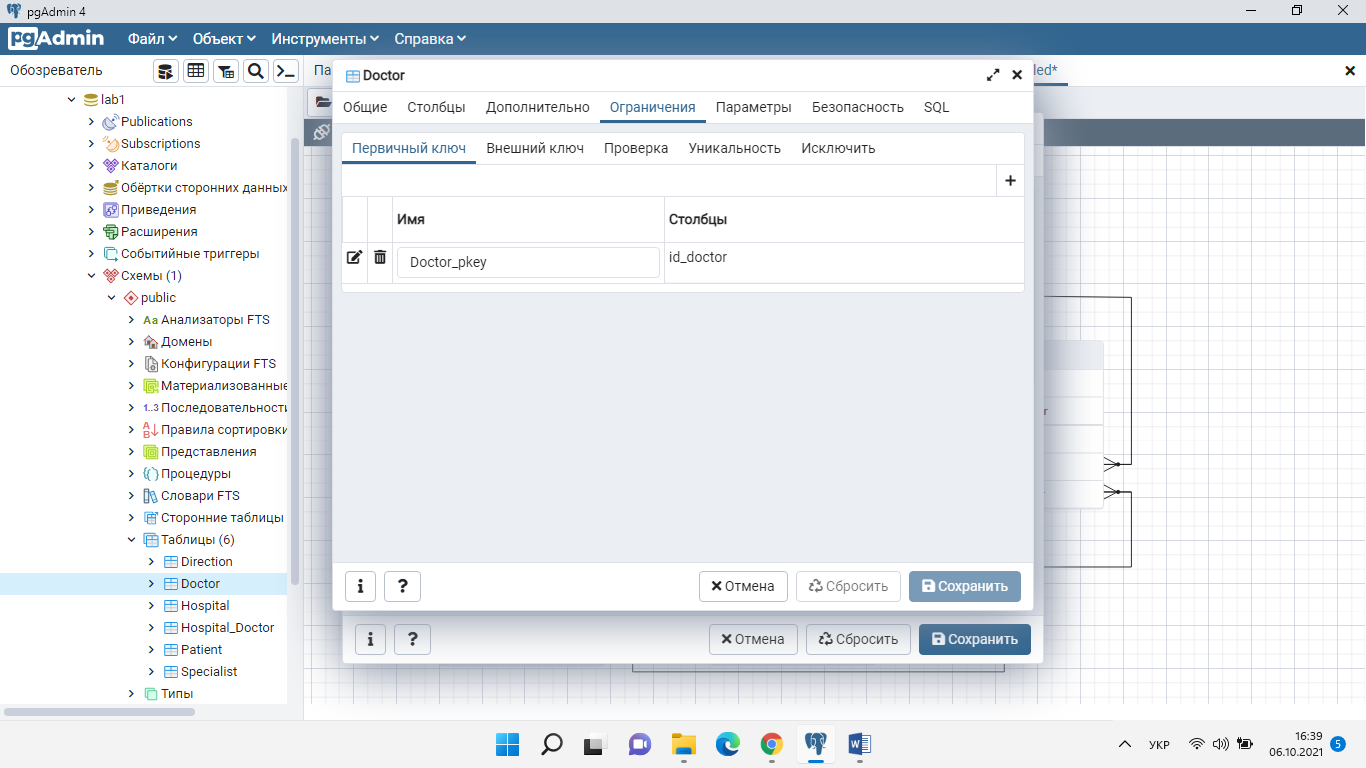
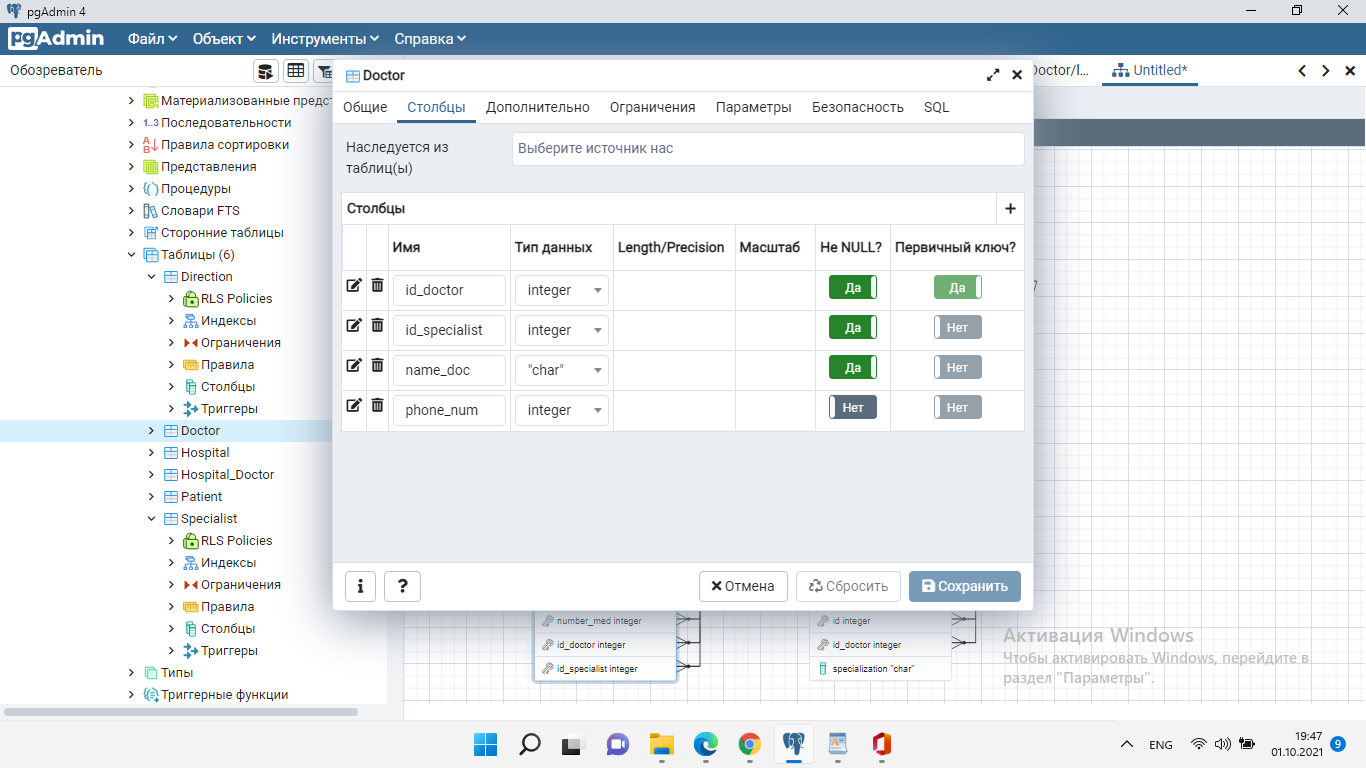
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Hospital\_Doctor\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

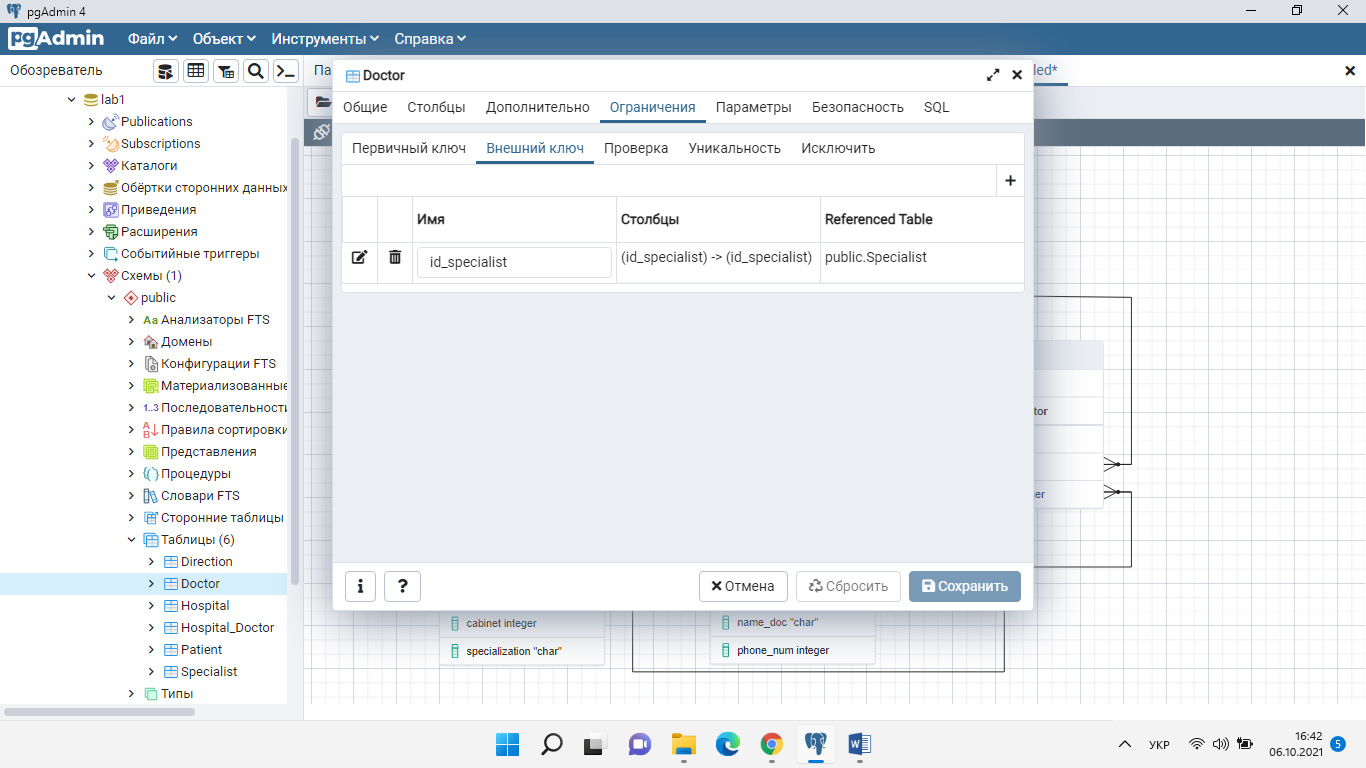




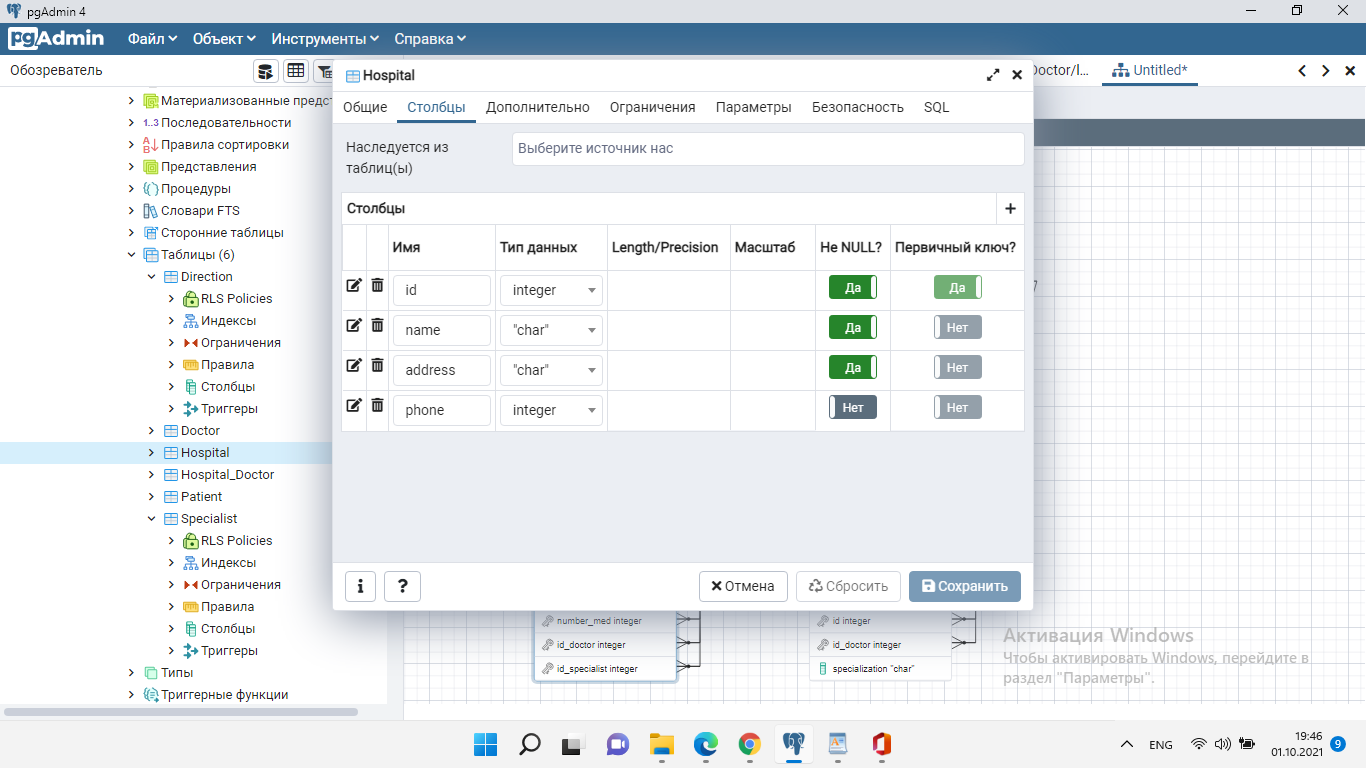


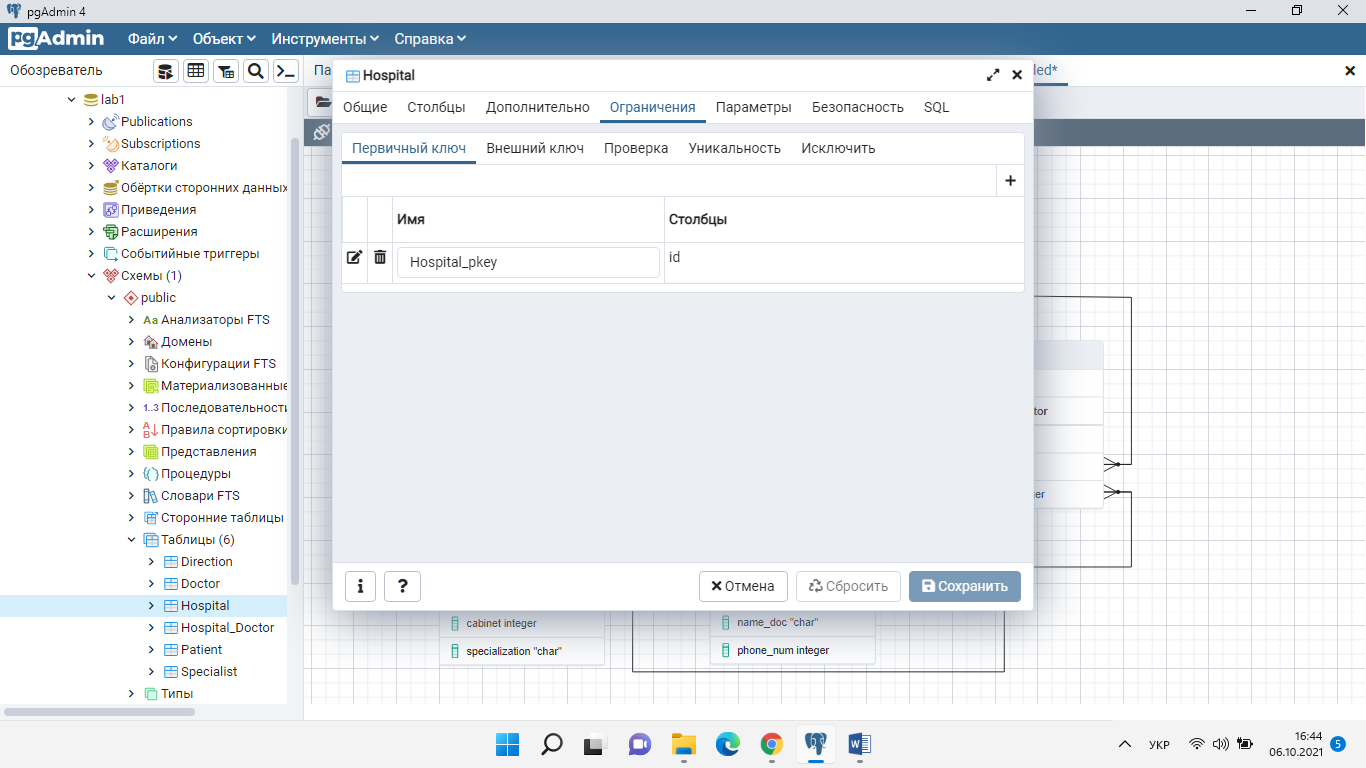
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Doctor\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_





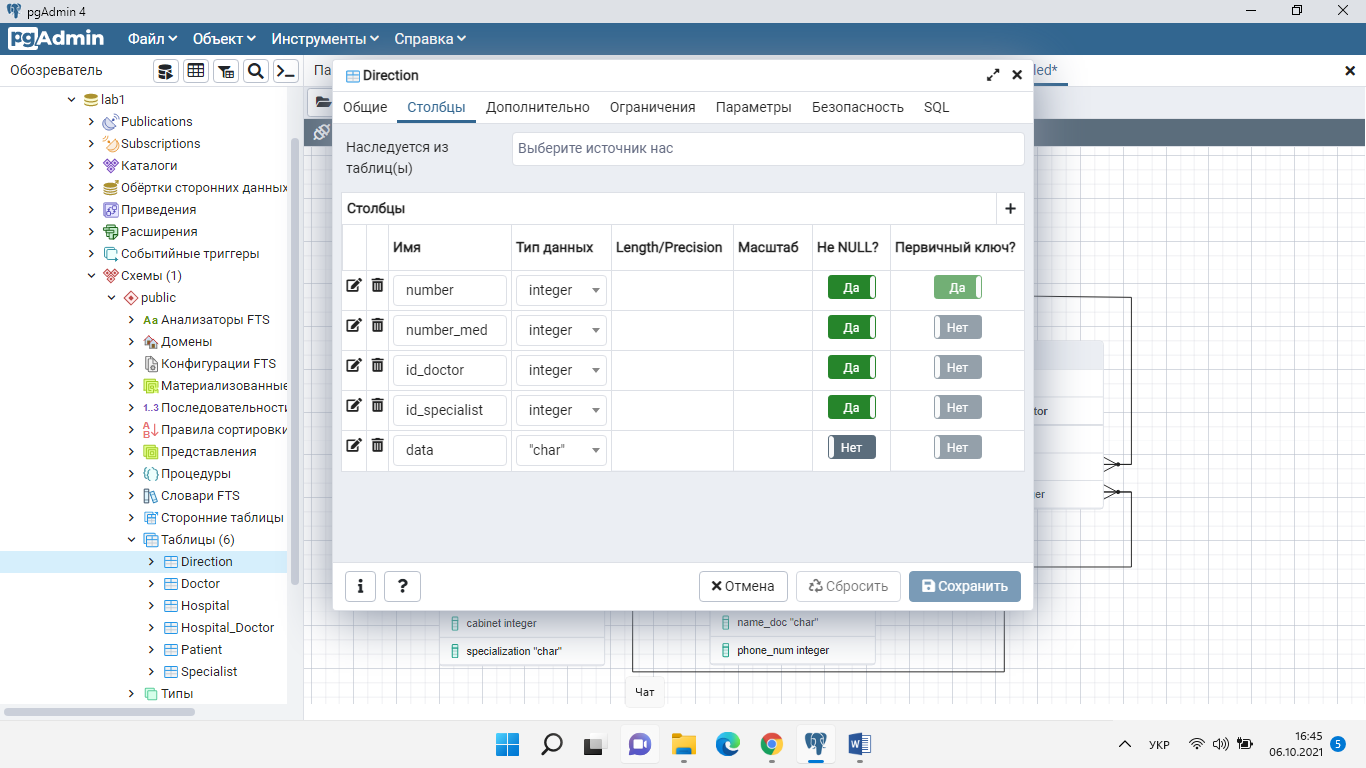
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Hospital\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

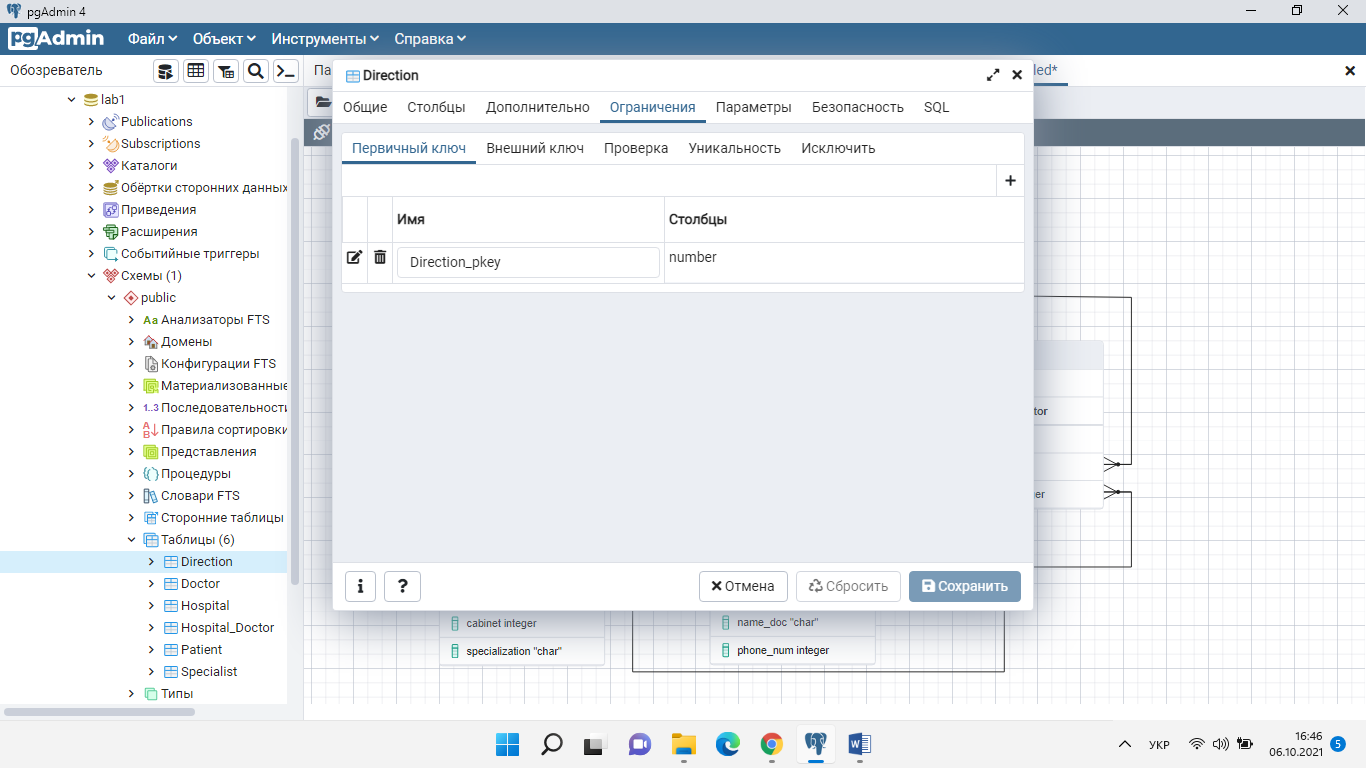


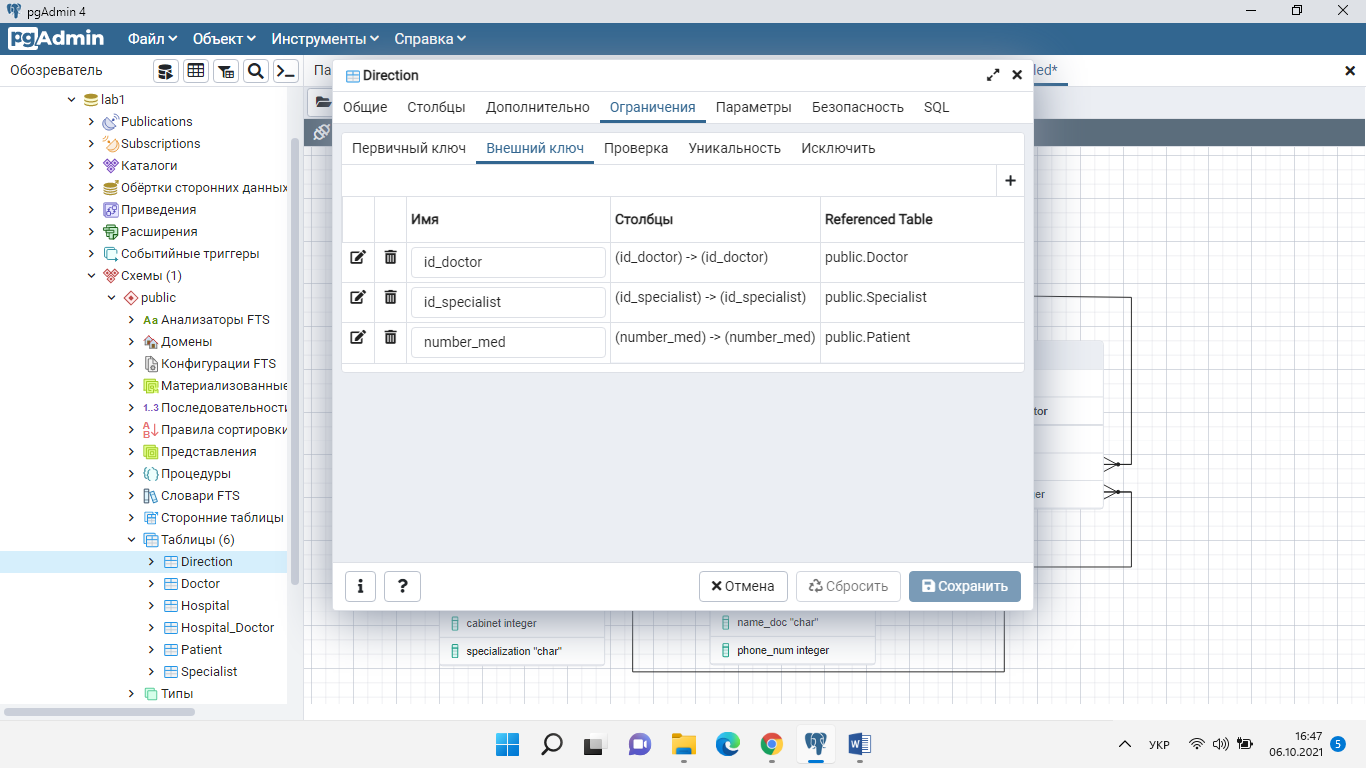


У цієї таблиці немає зовнішніх ключів (FK)

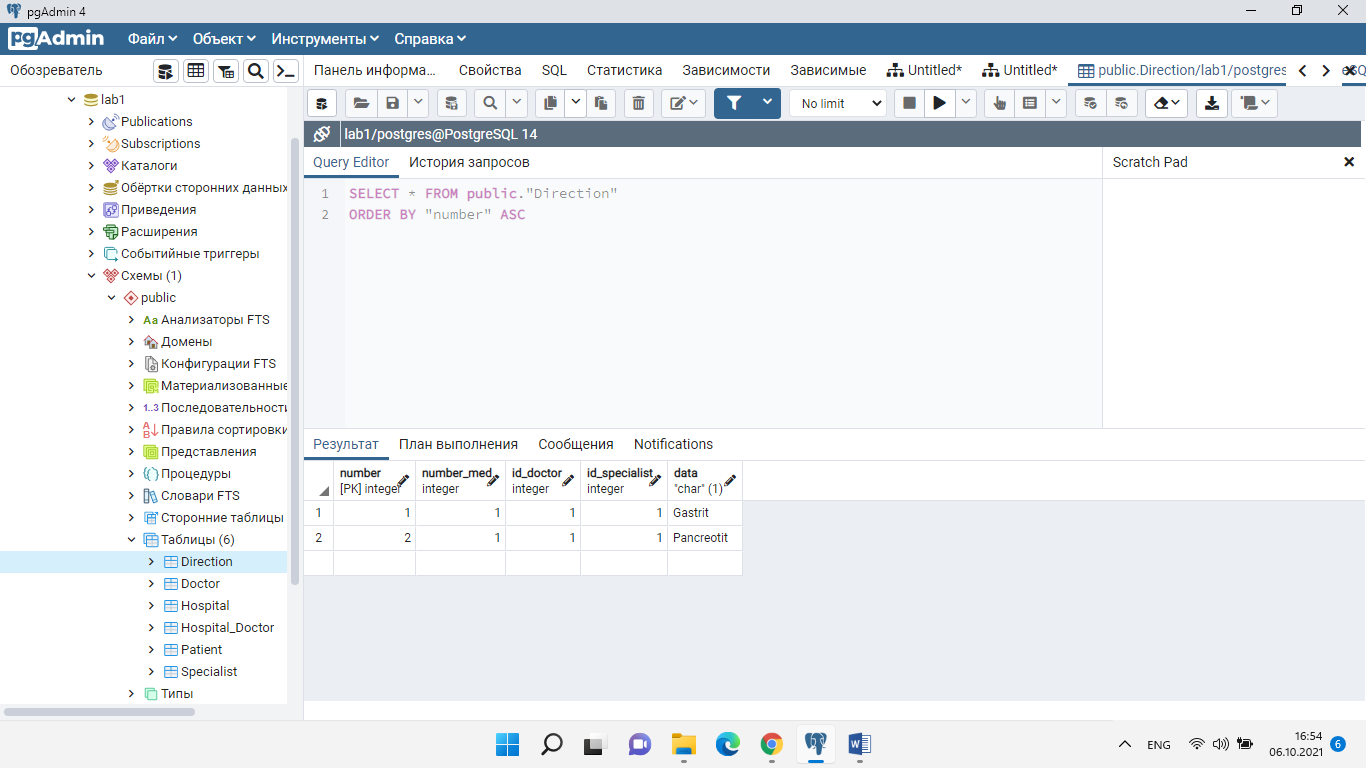
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Direction\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

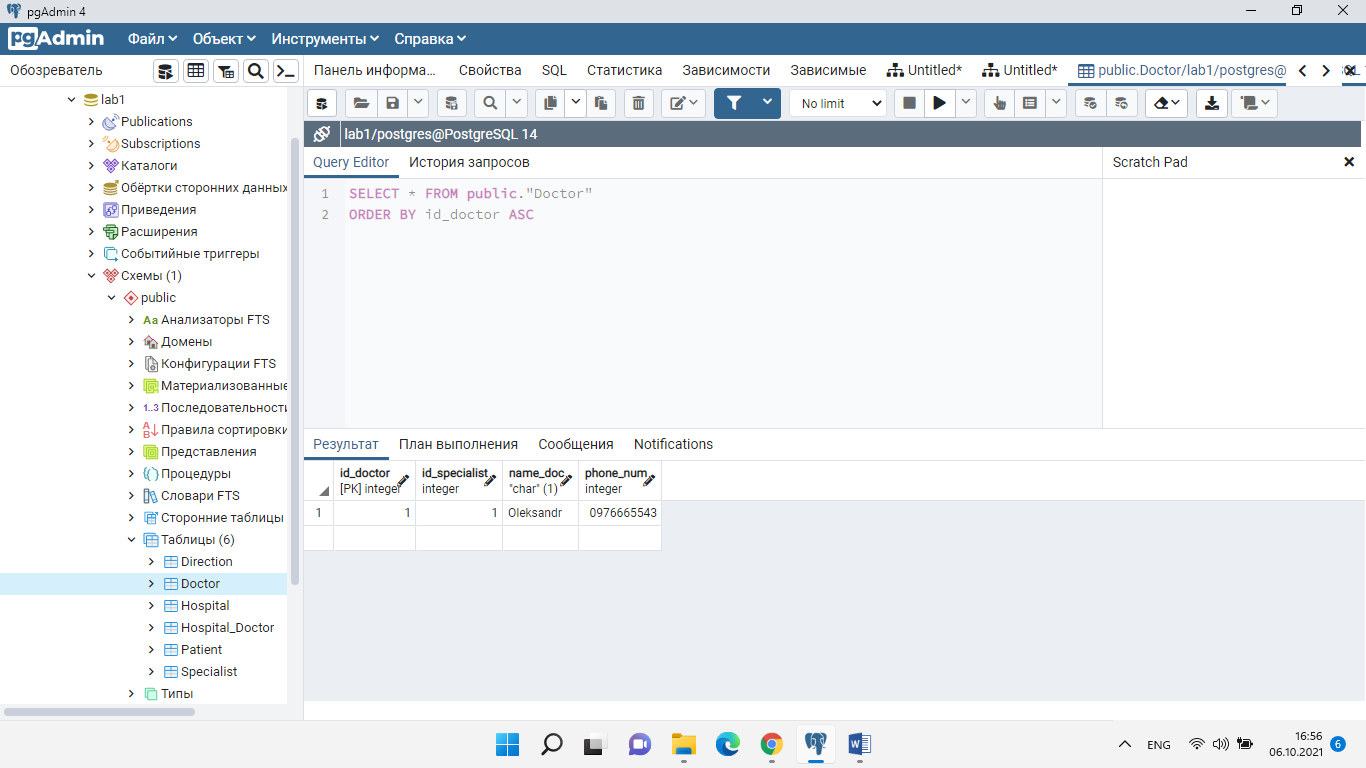


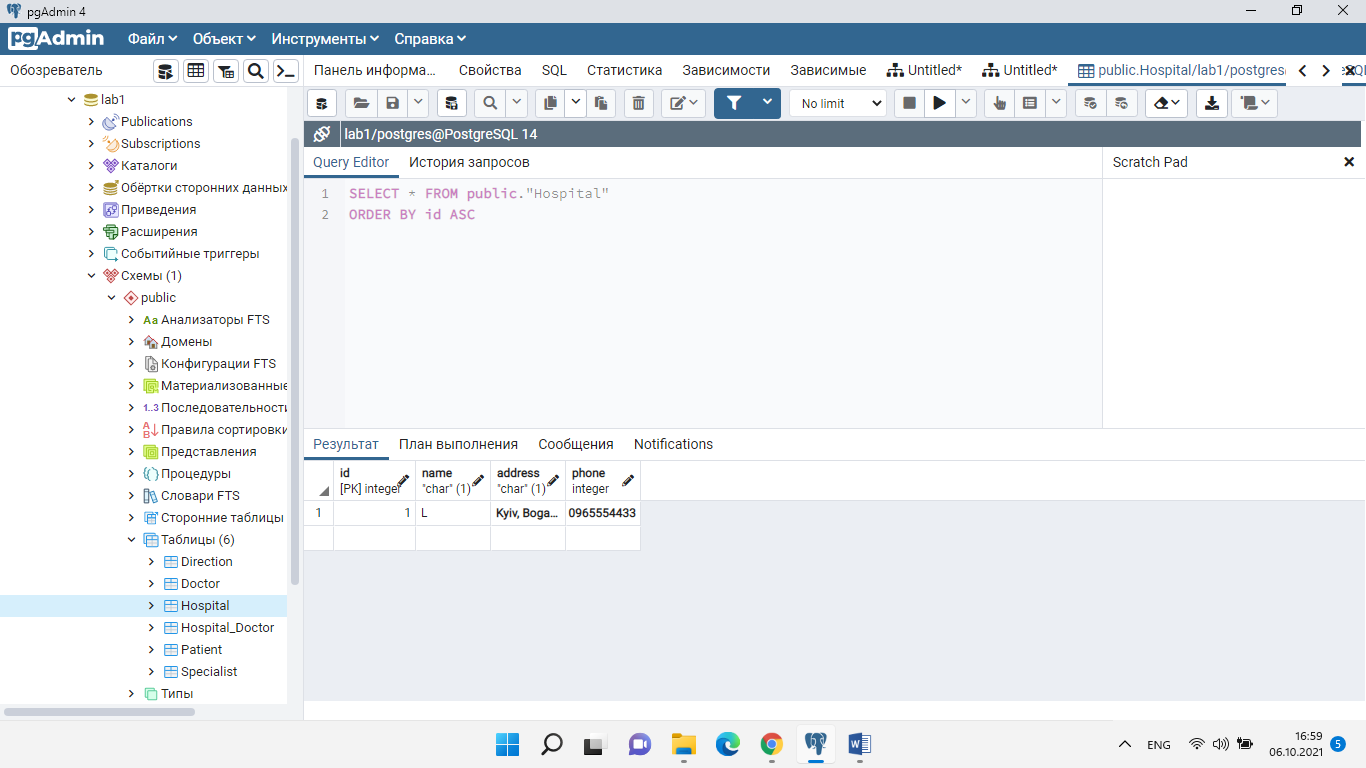


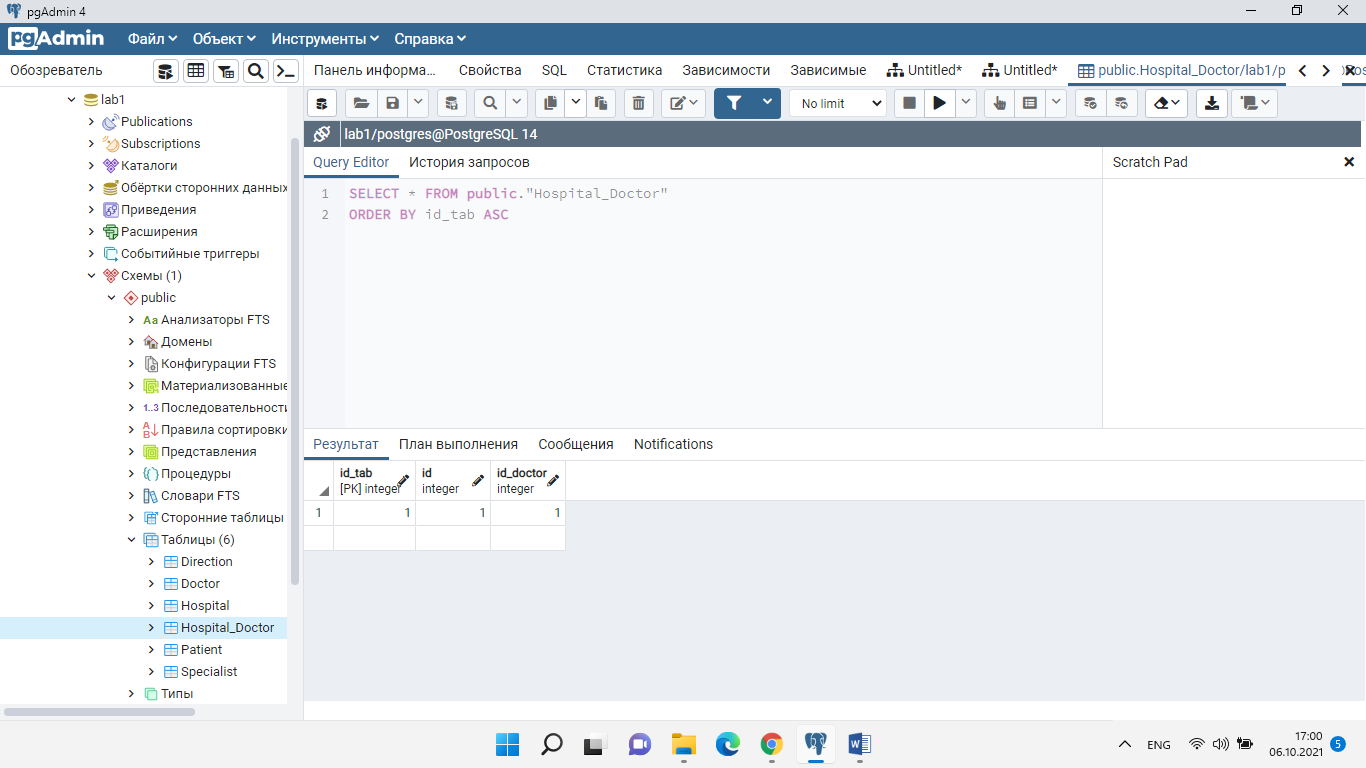


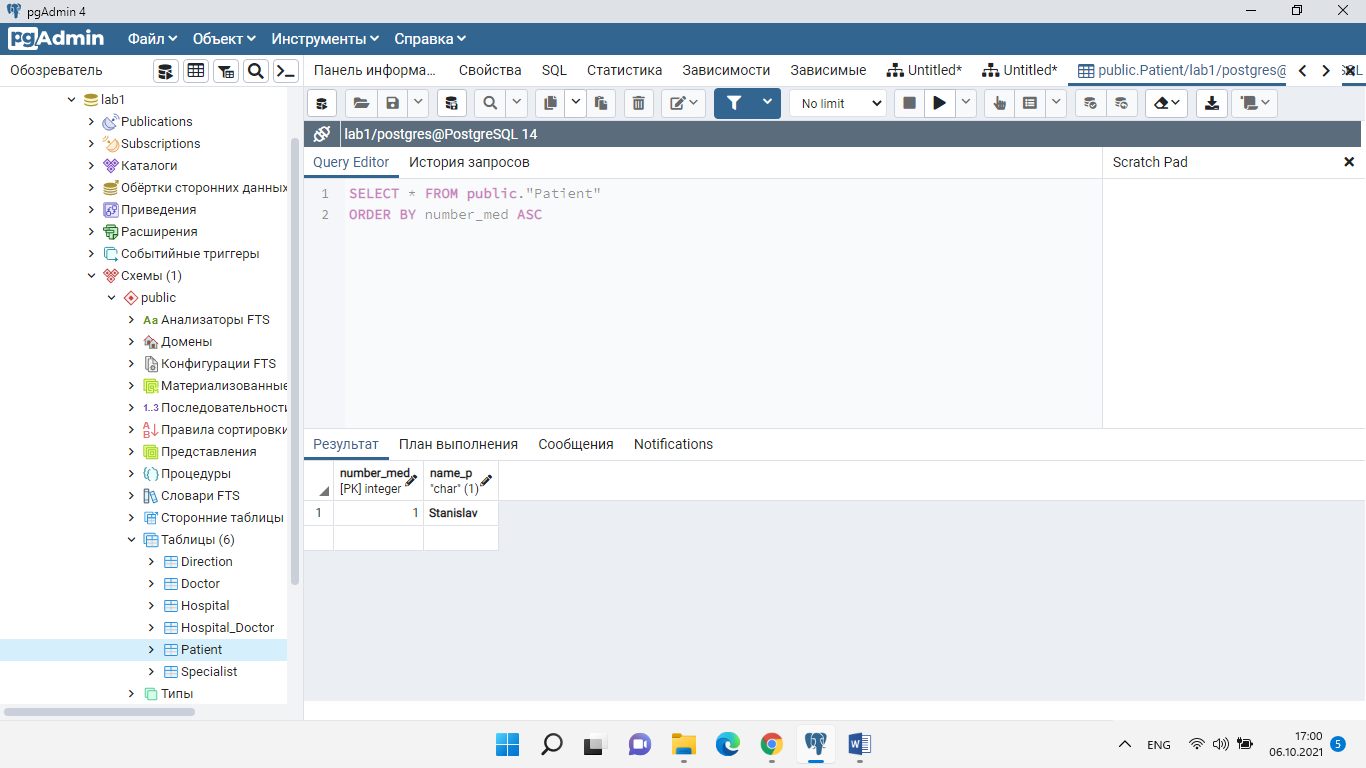
**Таблиці заповнені даними (уривки знімків екрану pgAdmin4)**

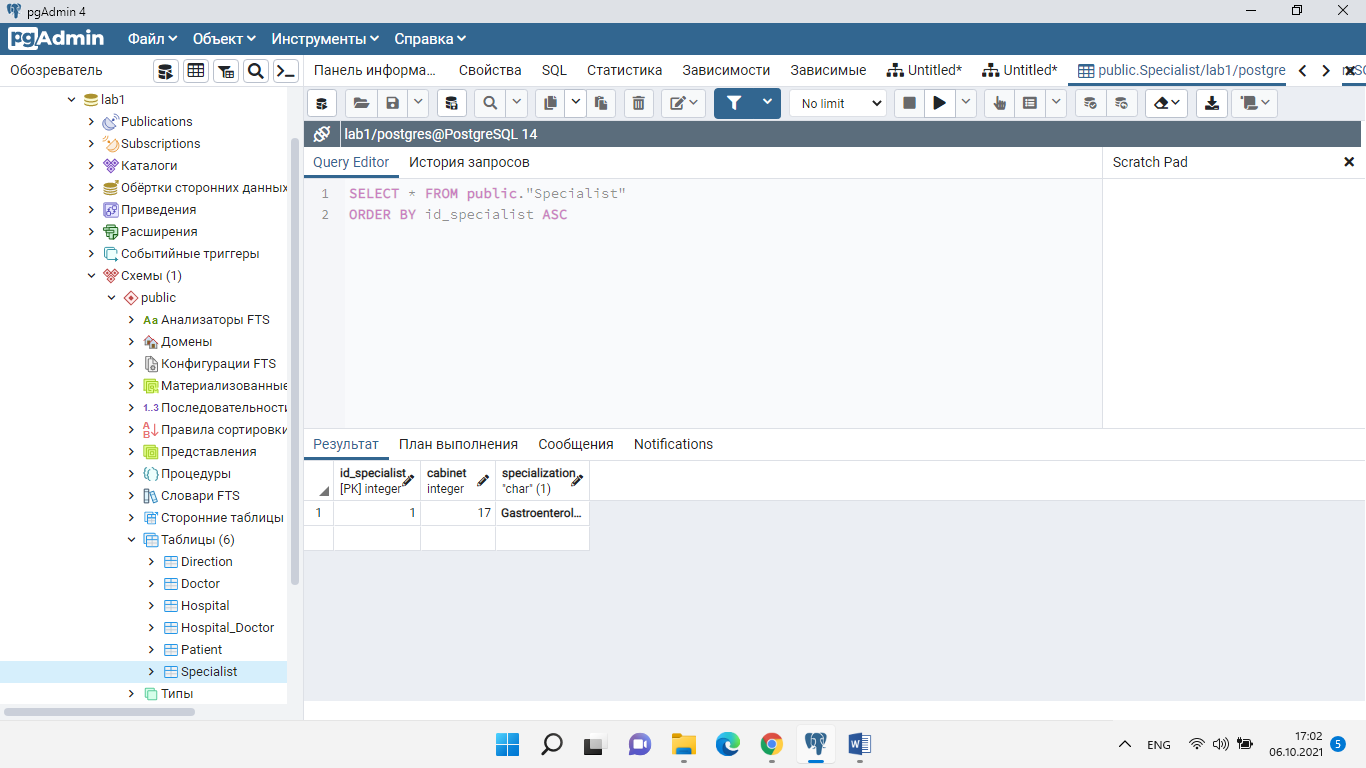












**Таблиці в коді SQL**

-- Table: public.Direction

-- DROP TABLE public."Direction";

CREATE TABLE IF NOT EXISTS public."Direction"

(

"number" integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

number\_med integer NOT NULL,

id\_doctor integer NOT NULL,

id\_specialist integer NOT NULL,

data "char",

CONSTRAINT "Direction\_pkey" PRIMARY KEY ("number"),

CONSTRAINT id\_doctor FOREIGN KEY (id\_doctor)

REFERENCES public."Doctor" (id\_doctor) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

CONSTRAINT id\_specialist FOREIGN KEY (id\_specialist)

REFERENCES public."Specialist" (id\_specialist) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

CONSTRAINT number\_med FOREIGN KEY (number\_med)

REFERENCES public."Patient" (number\_med) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE public."Direction"

OWNER to postgres;

-- Table: public.Doctor

-- DROP TABLE public."Doctor";

CREATE TABLE IF NOT EXISTS public."Doctor"

(

id\_doctor integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

id\_specialist integer NOT NULL,

name\_doc "char" NOT NULL,

phone\_num integer,

CONSTRAINT "Doctor\_pkey" PRIMARY KEY (id\_doctor),

CONSTRAINT id\_specialist FOREIGN KEY (id\_specialist)

REFERENCES public."Specialist" (id\_specialist) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE public."Doctor"

OWNER to postgres;

-- Table: public.Hospital

-- DROP TABLE public."Hospital";

CREATE TABLE IF NOT EXISTS public."Hospital"

(

id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

name "char" NOT NULL,

address "char" NOT NULL,

phone integer,

CONSTRAINT "Hospital\_pkey" PRIMARY KEY (id)

)

TABLESPACE pg\_default;

ALTER TABLE public."Hospital"

OWNER to postgres;

-- Table: public.Hospital\_Doctor

-- DROP TABLE public."Hospital\_Doctor";

CREATE TABLE IF NOT EXISTS public."Hospital\_Doctor"

(

id\_tab integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

id integer NOT NULL,

id\_doctor integer NOT NULL,

CONSTRAINT "Hospital\_Doctor\_pkey" PRIMARY KEY (id\_tab),

CONSTRAINT id FOREIGN KEY (id)

REFERENCES public."Hospital" (id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID,

CONSTRAINT id\_doctor FOREIGN KEY (id\_doctor)

REFERENCES public."Doctor" (id\_doctor) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

NOT VALID

)

TABLESPACE pg\_default;

ALTER TABLE public."Hospital\_Doctor"

OWNER to postgres;

-- Table: public.Patient

-- DROP TABLE public."Patient";

CREATE TABLE IF NOT EXISTS public."Patient"

(

number\_med integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

name "char" NOT NULL,

CONSTRAINT "Patient\_pkey" PRIMARY KEY (number\_med)

)

TABLESPACE pg\_default;

ALTER TABLE public."Patient"

OWNER to postgres;

-- Table: public.Specialist

-- DROP TABLE public."Specialist";

CREATE TABLE IF NOT EXISTS public."Specialist"

(

id\_specialist integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

cabinet integer NOT NULL,

specialization "char" NOT NULL,

CONSTRAINT "Specialist\_pkey" PRIMARY KEY (id\_specialist)

)

TABLESPACE pg\_default;

ALTER TABLE public."Specialist"

OWNER to postgres;