

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”*

Виконав:

студент ІII курсу

групи КВ-91

Тегза Д. І.

Перевірив:

Павловський В. І.

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 занести декілька рядків даних у кожну з таблиць засобами pgAdmin 4.

*Вимоги до ER-моделі:*

1. Сутності моделі предметної галузі мають містити зв’язки типу 1:N або N:M.
2. Кількість сутностей у моделі – 3-4. Кількість атрибутів у кожній сутності: від двох до п’яти.
3. Передбачити наявність зв’язку з атрибутом.

Для побудови ER-діаграм використовувати одну із нотацій: Чена, “Пташиної лапки (Crow’s foot)”, UML.

**Опис предметної галузі**

Для даної лабораторної роботи я вибрав тему – Гуртожитки КПІ.

При проектуванні бази даних «Гуртожиток» можна виділити наступні сутності: загальні відомості про гуртожиток (dormitory), відомості про кімнату (room), відомості про тип кімнати (room\_type), студент або жилець (inmate).

Атрибути заданих сутностей:

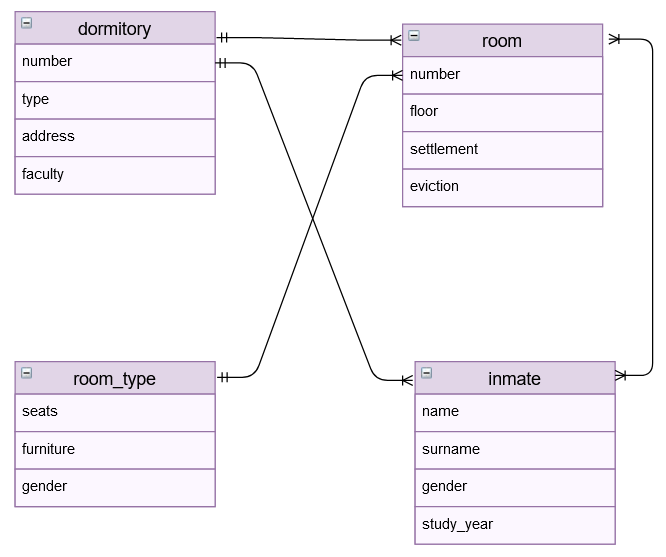
* + - 1. dormitory: number, type, address, faculty.
      2. room: number, floor, settlement, eviction
      3. room\_type: seats, furniture, gender.
      4. inmate: name, surname, gender, study\_year.

**Опис зв’язків**

У конкретному гуртожитку находиться багато кімнат, тому між сутностями dormitory і room зв’язок 1:N

У конкретному гуртожитку може проживати багато студентів, тому між сутностями dormitory і inmate зв’язок 1:N

Певному типу кімнат можуть належати зразу багато кімнат, тому між сутностями room\_type і room зв’язок 1:N

В одній кімнаті може проживати декілька студентів, також один студент може проживати в різних кімнатах, тому між сутностями room і inmate зв’язок M:N

ER-діаграма побудована за нотацією “Пташиної лапки (Crow’s foot)”, задана ER-діаграма була побудована у додатку [draw.io](https://app.diagrams.net/)

**Перетворення концептуальної моделі у схему баз даних**

Для кожної сутності створюється таблиця. Причому кожному атрибуту сутності відповідає стовпець таблиці. В даній моделі перетворення в схему баз даних відбувалося за такими правилами:

Якщо зв'язок типу 1: N і клас приналежності сутності на стороні N є обов'язковим, то необхідно побудувати таблицю для кожної сутності. Первинний ключ сутності повинен бути первинним ключем відповідної таблиці. Первинний ключ сутності на стороні 1 додається як атрибут в таблицю для сутності на стороні N. Даний зв’язок можна спостерігати між сутностями dormitory і room, dormitory і inmate, room\_type і room.

Якщо зв'язок типу N: M, то необхідно побудувати три таблиці - по одній для кожної сутності і одну для зв'язку. Первинний ключ сутності повинен бути первинним ключем відповідної таблиці. Таблиця для зв'язку серед своїх атрибутів повинна мати ключі обох сутностей. Даний зв’язок можна спостерігати між сутностями room і inmate.

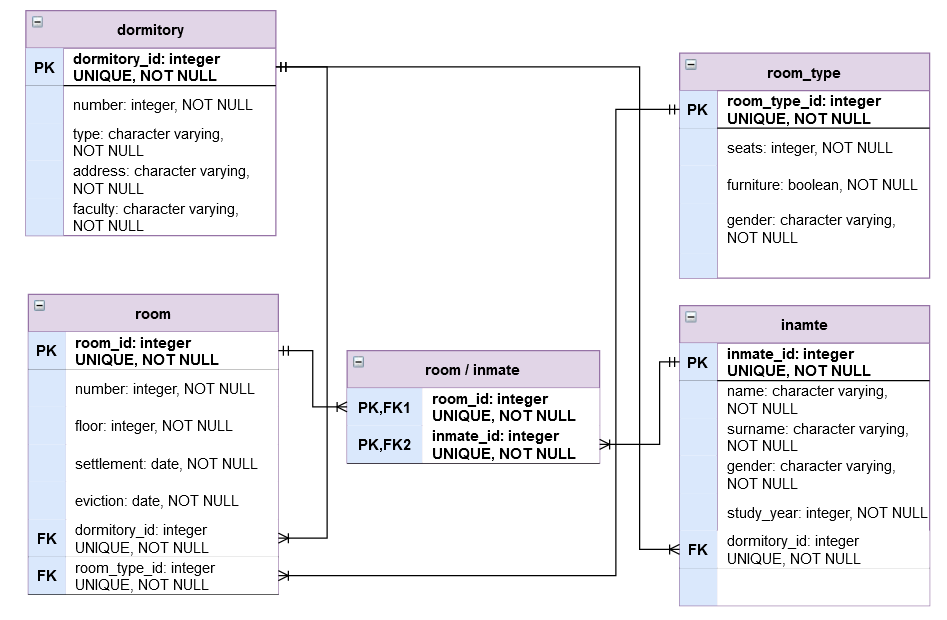
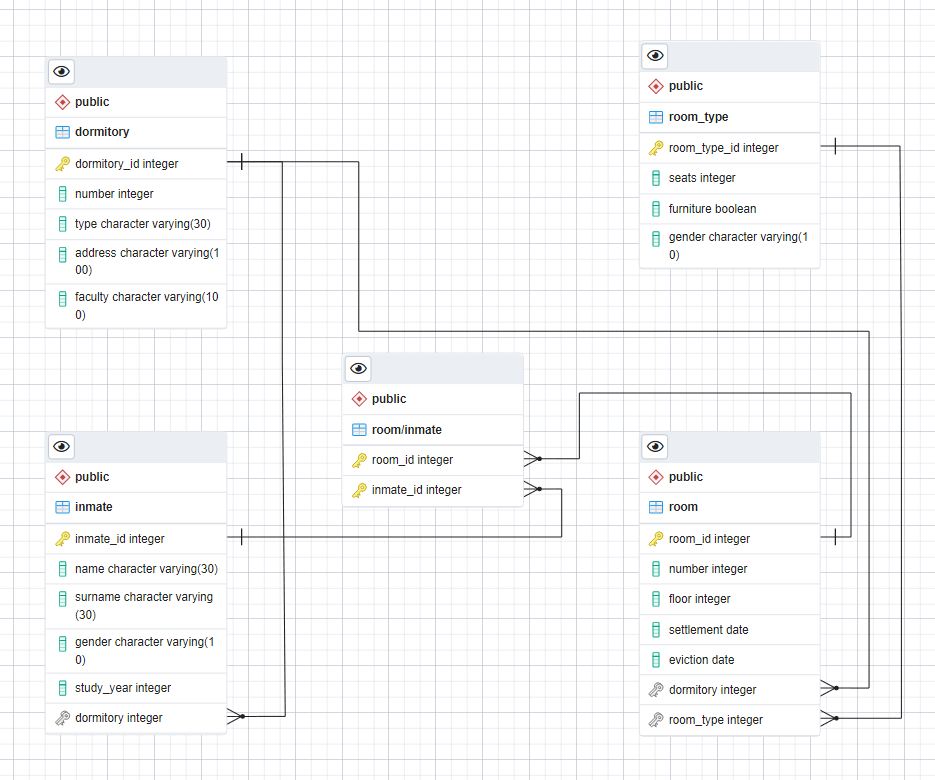


Схема бази даних, побудовано у додатку [draw.io](https://app.diagrams.net/)

Після розробки моделі предметної галузі «Готель» та перетворення її у схему бази даних, було створено дану базу даних у додатку pgAdmin 4



Опис структури Бази Даних

|  |  |  |
| --- | --- | --- |
| Відношення | Атрибут | Тип атрибуту |
| **dormitory** –містить дані про готель | **dormitory\_id** – унікальний ідентифікатор **number** – номер гуртожитку **type** – тип (блочний/не блочний) **address** – адреса **faculty** – факультет, який поселяють у даний гуртожиток | Integer integer character varying character varying character varying |
| **room\_type** – містить дані про тип кімнати | **room\_type\_id** – унікальний ідентифікатор **seats** – кількість місць у кімнаті **furniture** – кімната здається разом із меблями, чи без **gender** – жіноча кімната або чоловіча | integer integer boolean character varying |
| **room** – містить дані про кімнату | **room\_id** – унікальний ідентифікатор **number** – номер кімнати **floor** – поверх **settlement** – дата поселення **eviction** – дата виселення **dormitory\_id** – ідентифікатор гуртожитку **room\_type\_id** – ідентифікатор типу кінмати | integer integer integer date date integer integer |
| **inmate** – містить дані про студента (жильця) | **inmate\_id** – унікальний ідентифікатор **name** – ім’я студента **surname** – прізвище **gender** – стать **study\_year** – курс (рік навчання) **dormitory\_id** – ідентифікатор гуртожитку | integer character varying character varying character varying integer integer |
| **room/inmate** – відношення жильців до кімнат | **room\_id** – ідентифікатор кімнати **inmate\_id** – ідентифікатор жильця | integer integer |

Нормалізація схеми бази даних до третьої нормальної форми

Функціональні залежності для кожної таблиці

1. dormitory

* dormitory\_id -> number (номер гуртожитку залежить від унікального коду гуртожитку)
* dormitory\_id -> type (тип гуртожитку залежить від унікального коду гуртожитку)
* dormitory\_id -> address (адрес гуртожитку залежить від унікального коду гуртожитку)
* dormitory\_id -> faculty (факультет, який поселяють в даний гуртожиток, залежить від унікального коду гуртожитку)

1. room\_type

* room\_type\_id -> seats (кількість місць в кімнаті залежить від унікального коду типу кімнати)
* room\_type\_id -> furniture (наявність меблів в кімнаті залежить від унікального коду типу кімнати)
* room\_type\_id -> gender (стать, яку поселяють в кімнату, залежить від унікального коду типу кімнати)

1. room

* room\_id -> number (номер кімнати залежить від унікального коду кімнати)
* room\_id -> floor (поверх, на якому знаходиться кімната, залежить від унікального коду кімнати)
* room\_id -> settlement (дата заселення в кімнату залежить від унікального коду кімнати)
* room\_id -> eviction (дата виселення із кімнати залежить від унікального коду кімнати)

1. inmate

* inmate\_id -> name (ім’я студента, якого поселяють в кімнату, залежить від унікального коду студента)
* inmate\_id -> surname (прізвище студента, якого поселяють в кімнату, залежить від унікального коду студента)
* inmate\_id -> gender (стать студента, якого поселяють в кімнату, залежить від унікального коду студента)
* inmate\_id -> study\_year (рік навчання студента, якого поселяють в кімнату, залежить від унікального коду студента)

**Відповідність схеми бази даних до третьої нормальної форми**

Схема відповідає 1НФ, тому що:

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 неключовий атрибут нетранзитивно (без посередника) залежить від первинного ключа

SQL TEXT

-- Table: public.dormitory

-- DROP TABLE public.dormitory;

CREATE TABLE IF NOT EXISTS public.dormitory

(

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

"number" integer NOT NULL,

type character varying(30) COLLATE pg\_catalog."default" NOT NULL,

address character varying(100) COLLATE pg\_catalog."default" NOT NULL,

faculty character varying(100) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT dormitory\_pkey PRIMARY KEY (dormitory\_id)

)

TABLESPACE pg\_default;

ALTER TABLE public.dormitory

OWNER to postgres;

-- Table: public.inmate

-- DROP TABLE public.inmate;

CREATE TABLE IF NOT EXISTS public.inmate

(

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

name character varying(30) COLLATE pg\_catalog."default" NOT NULL,

surname character varying(30) COLLATE pg\_catalog."default" NOT NULL,

gender character varying(10) COLLATE pg\_catalog."default" NOT NULL,

study\_year integer NOT NULL,

dormitory integer NOT NULL,

CONSTRAINT inmate\_pkey PRIMARY KEY (inmate\_id),

CONSTRAINT fk\_inmate\_dormitory FOREIGN KEY (dormitory)

REFERENCES public.dormitory (dormitory\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE public.inmate

OWNER to postgres;

-- Table: public.room

-- DROP TABLE public.room;

CREATE TABLE IF NOT EXISTS public.room

(

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

"number" integer NOT NULL,

floor integer NOT NULL,

settlement date NOT NULL,

eviction date NOT NULL,

dormitory integer NOT NULL,

room\_type integer,

CONSTRAINT room\_pkey PRIMARY KEY (room\_id),

CONSTRAINT fk\_room\_dormitory FOREIGN KEY (dormitory)

REFERENCES public.dormitory (dormitory\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT fk\_room\_roomtype FOREIGN KEY (room\_type)

REFERENCES public.room\_type (room\_type\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE public.room

OWNER to postgres;

-- Table: public.room/inmate

-- DROP TABLE public."room/inmate";

CREATE TABLE IF NOT EXISTS public."room/inmate"

(

room\_id integer NOT NULL,

inmate\_id integer NOT NULL,

CONSTRAINT "room/inmate\_pkey" PRIMARY KEY (room\_id, inmate\_id),

CONSTRAINT fk\_inmate FOREIGN KEY (inmate\_id)

REFERENCES public.inmate (inmate\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION,

CONSTRAINT fk\_room FOREIGN KEY (room\_id)

REFERENCES public.room (room\_id) MATCH SIMPLE

ON UPDATE NO ACTION

ON DELETE NO ACTION

)

TABLESPACE pg\_default;

ALTER TABLE public."room/inmate"

OWNER to postgres;

-- Table: public.room\_type

-- DROP TABLE public.room\_type;

CREATE TABLE IF NOT EXISTS public.room\_type

(

room\_type\_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

seats integer NOT NULL DEFAULT 4,

furniture boolean NOT NULL,

gender character varying(10) COLLATE pg\_catalog."default" NOT NULL,

CONSTRAINT room\_type\_pkey PRIMARY KEY (room\_type\_id)

)

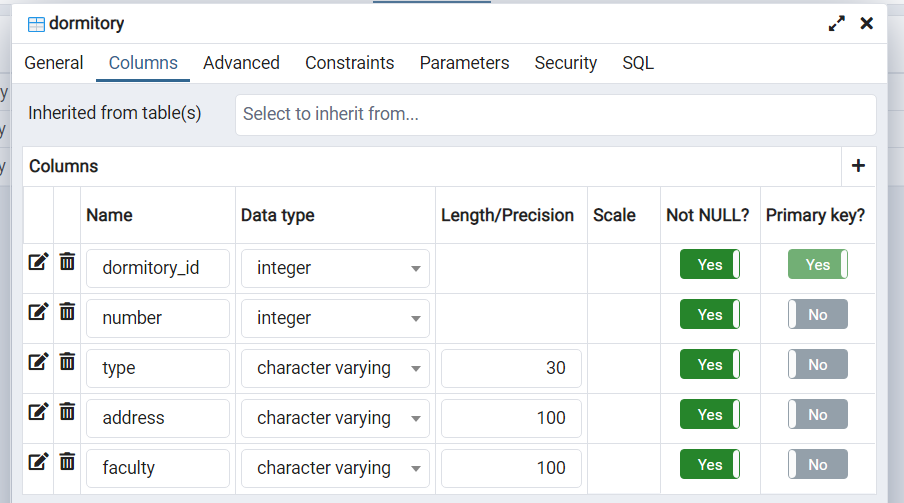
TABLESPACE pg\_default;

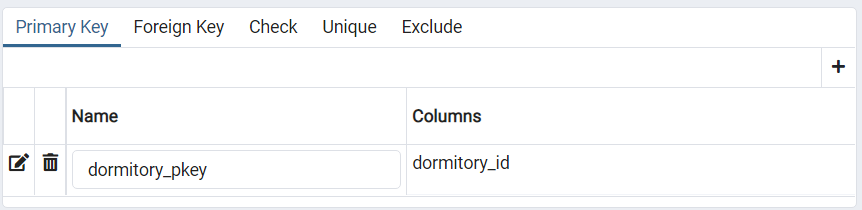
ALTER TABLE public.room\_type

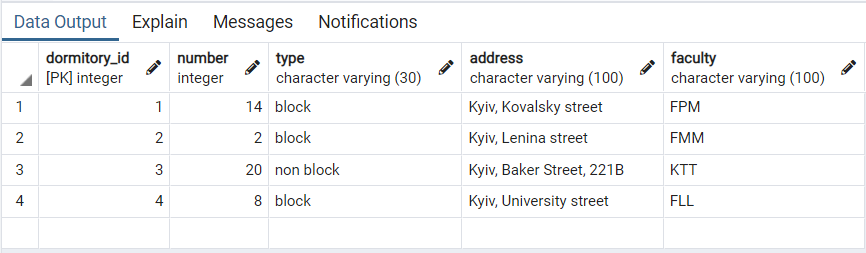
OWNER to postgres;

Таблиці бази даних у pgAdmin 4

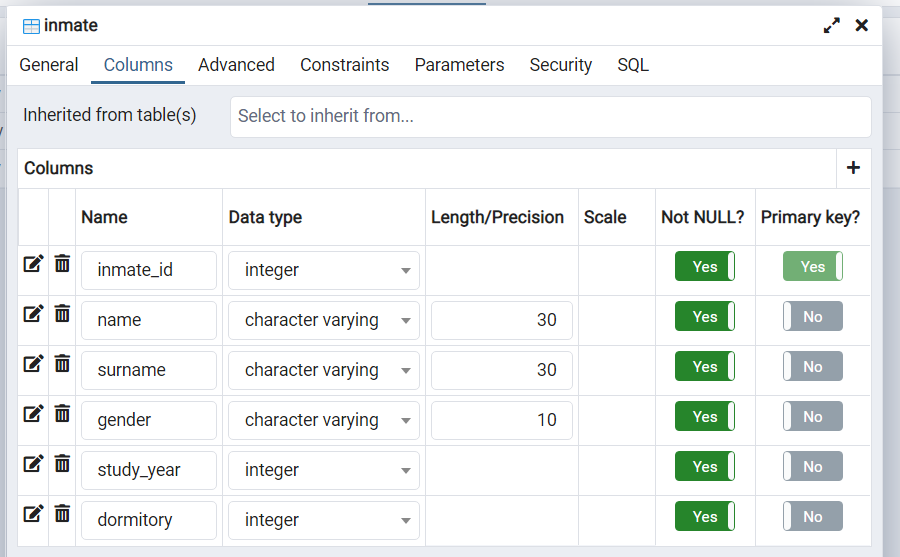
Dormitory



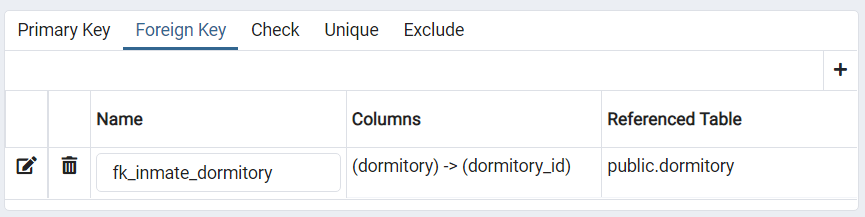




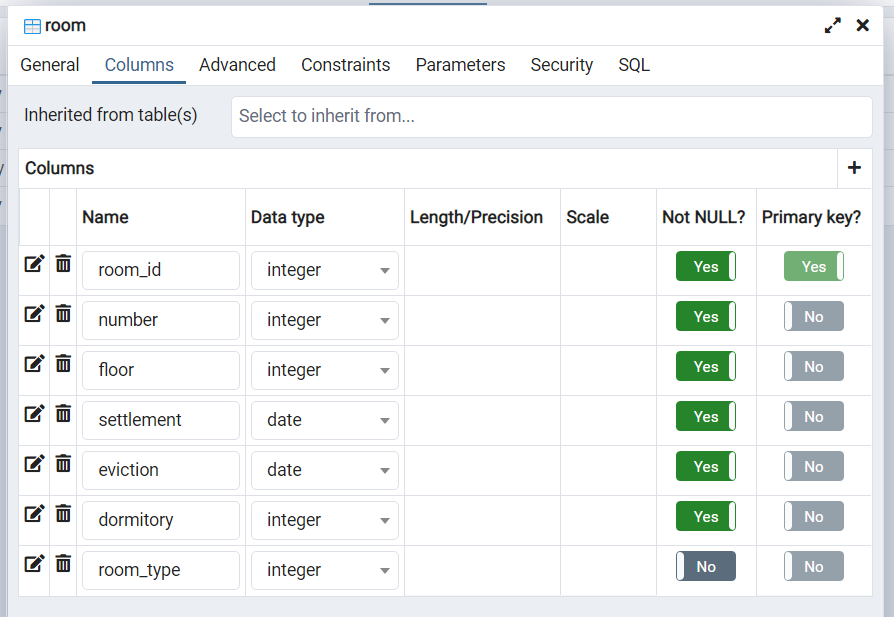
Inmate

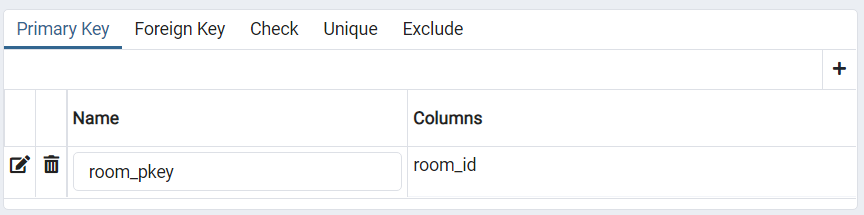


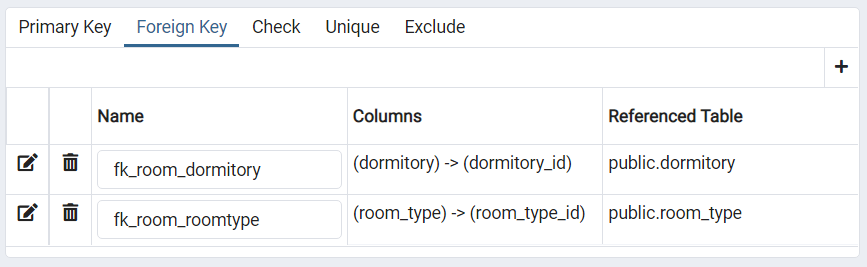


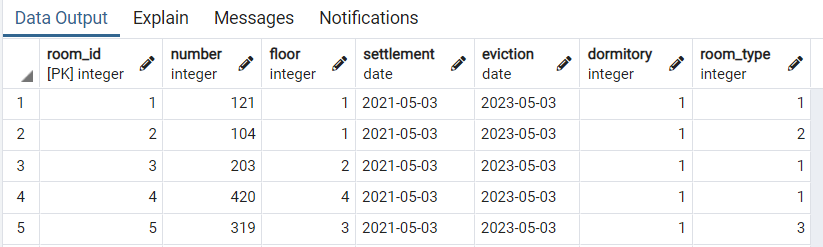


Room

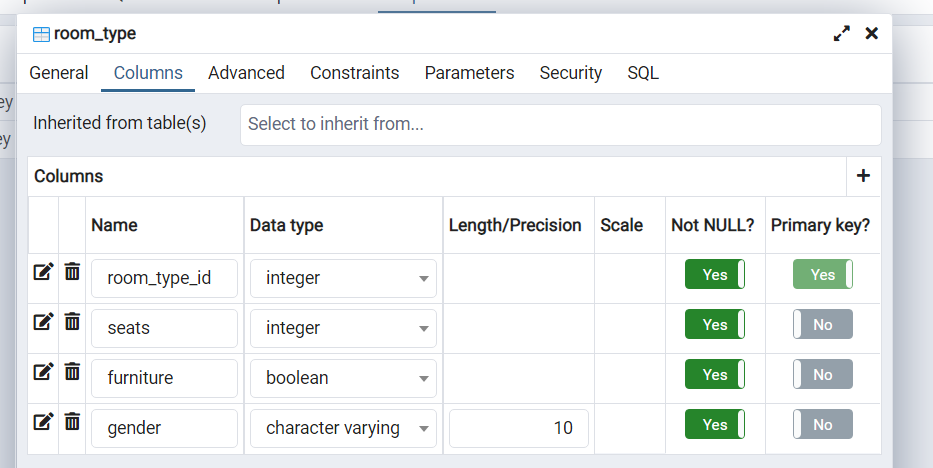


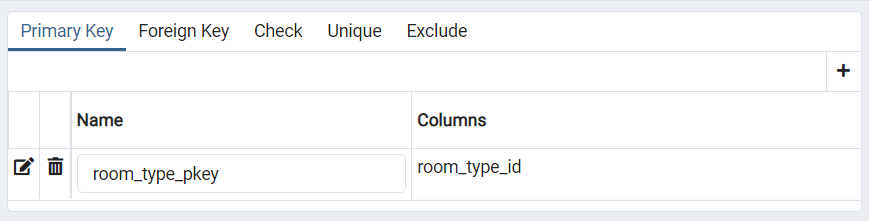


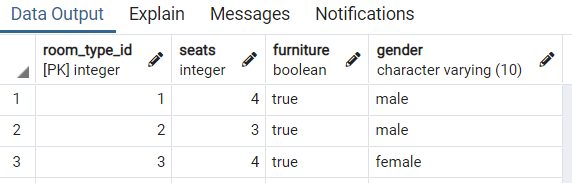




Room\_type







Room/inmate

