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»

«Ознайомлення з базовими операціями СУБД PostgreSQL»

Виконав:
студент групи КВ-82
Ященко Іван Васильович

Перевірив:
_____
_____
_____

Метою роботи є здобуття практичних навичок створення реляційних баз даних за допомогою PostgreSQL.

У звіті щодо пункту №1 завдання має бути:

– перелік сутностей з описом їх призначення;
– графічний файл розробленої моделі «сутність-зв'язок»;
– назва нотації.

У звіті щодо пункту №2 завдання має бути:

– опис процесу перетворення (наприклад, "сутність А було перетворено у таблицю А, а зв'язок R (M:N) зумовив появу додаткової таблиці R1 тощо);
– схему бази даних у графічному вигляді **з назвами таблиць (!) та зв'язками між ними.**

У звіті щодо пункту №3 завдання має бути:

– пояснення щодо відповідності схеми бази даних нормальним формам НФ1, НФ2 та НФ3. У випадку невідповідності надати опис необхідних змін у схемі;
– У випадку проведення змін у схемі бази даних надати оновлену версію схеми, інакше - не наводити схему.

У звіті щодо пункту №4 завдання має бути:

– навести копії екрану з pgAdmin4, що відображають назви та типи стовпців (доступне у закладці "Columns" властивостей "Properties" таблиць дерева об'єктів у pgAdmin4);
– навести копії екрану з pgAdmin4, що відображають вміст таблиць бази даних у PostgreSQL. Таблиці на зображенні обов'язково **повинні мати назву**!

Опис предметної галузі

(Сервіс продажів квитків в кіно). При проектуванні даної галузі можна виділити такі сутності.

Зал (Room) – призначений для ідентифікації кінозали де будуть проходити сеанси кінофільмів і також розподіл кімнат на кольори для їх відмінності(зроблено на прикладі кінозалів Butterfly у м. Київ), сеанс(show_time) створений для ідентифікації сеансу кінофільму де також зберігається ідентифікатори фільму та дата проведення сеансу.

фільм (Film) – створений для ідентифікації фільму, його імені та час його проходження.

місце/ряд (Seat/Row) – створений для ідентифікації місця для сидіння, включає в себе ідентифікатори місця і ряду,

клієнт (Client) - створений для ідентифікації клієнта, його імені та віку.

номер телефону клієнту (Telephone) – створений для ідентифікації номера телефону клієнту , що включає в себе номер телефону , ідентифікатор клієнту та сам номер телефону.

квиток(Ticket) - створений для ідентифікації квитка на кіно, що

включає в себе ідентифікатор квитка, ідентифікатор місця для сидіння, ідентифікатор сеансу, та ідентифікатор клієнту.

В багатьох кіно-залах може відбуватися декілька сеансів за один день (M:N);

В кожному сеансі може відтворюватись лише один фільм (1:1);

В кожному залі є багато місць(1:N);

В кожному квитку може бути одне місце, сеанс та зал (1:1);

У одного клієнта може бути багато квитків (1:N) або 0 квитків;

У одного клієнта може бути декілька телефонів (1:N);

Концептуальна модель учбової предметної області " Сервіс продажу квитків кіно "

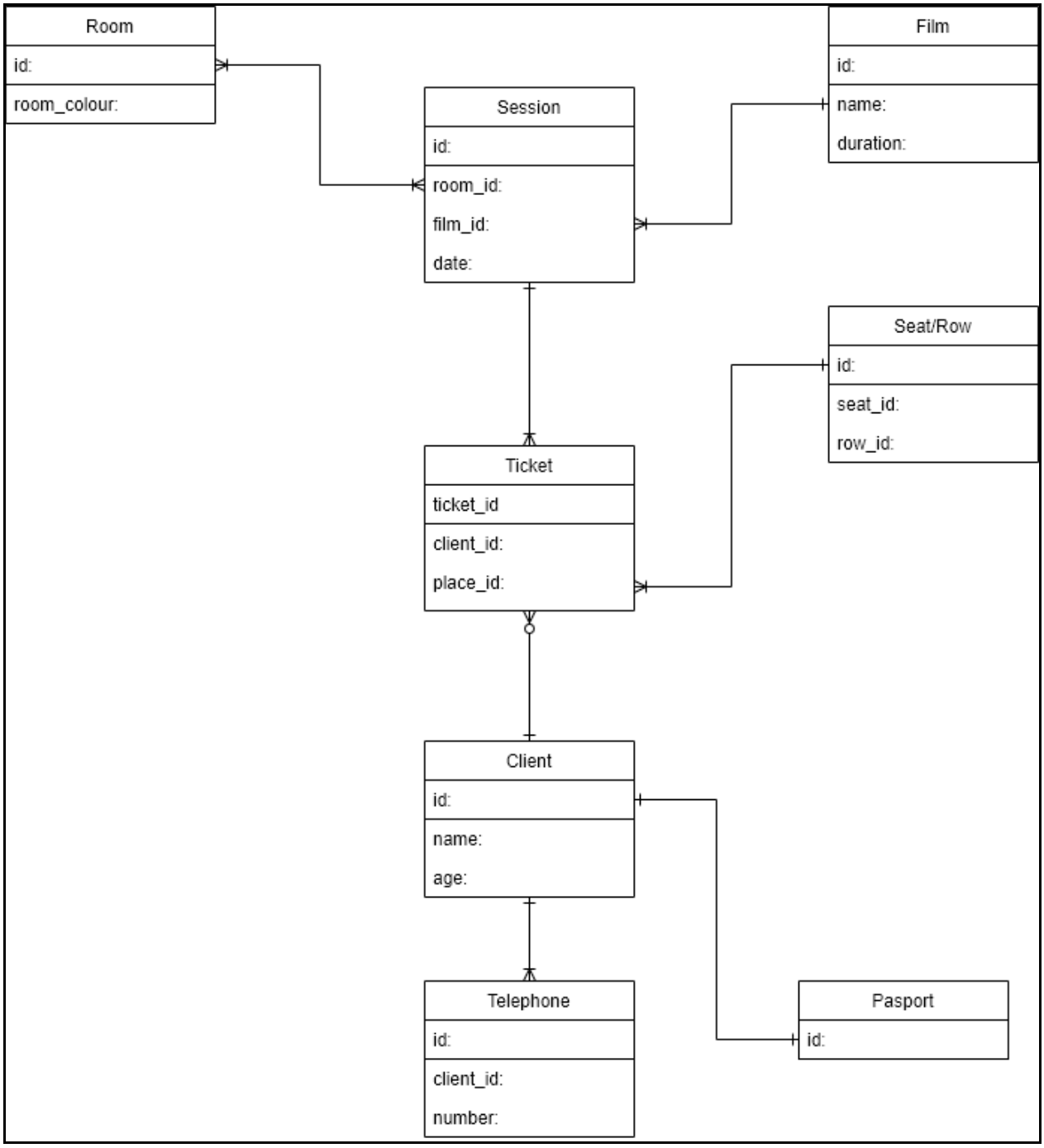


Рисунок 1 - Концептуальна модель предметної області " Сервіс продажу квитків кіно "

Нотація: Пташина лапка + засоби програми drow.io

Пояснення щодо відповідності схеми бази даних нормальним формам:

Схема бази даних відповідає 1НФ тому, що схема передбачає 1 елемент в кожній комірці.

Схема бази даних відповідає 2НФ тому, що по-перше відповідає 1НФ, а по-друге

Схема не включає в собі залежності від декількох потенційних ключів, а залежить лише від одного.

Схема бази даних відповідає 3НФ тому, що по-перше вона відповідає 2НФ, а по-друге відсутні транзитивні функціональні залежності неключових атрибутів від ключових.

Логічна модель (Структура) БД “ Сервіс продажу квитків кіно ” (засобами SqlDMB)

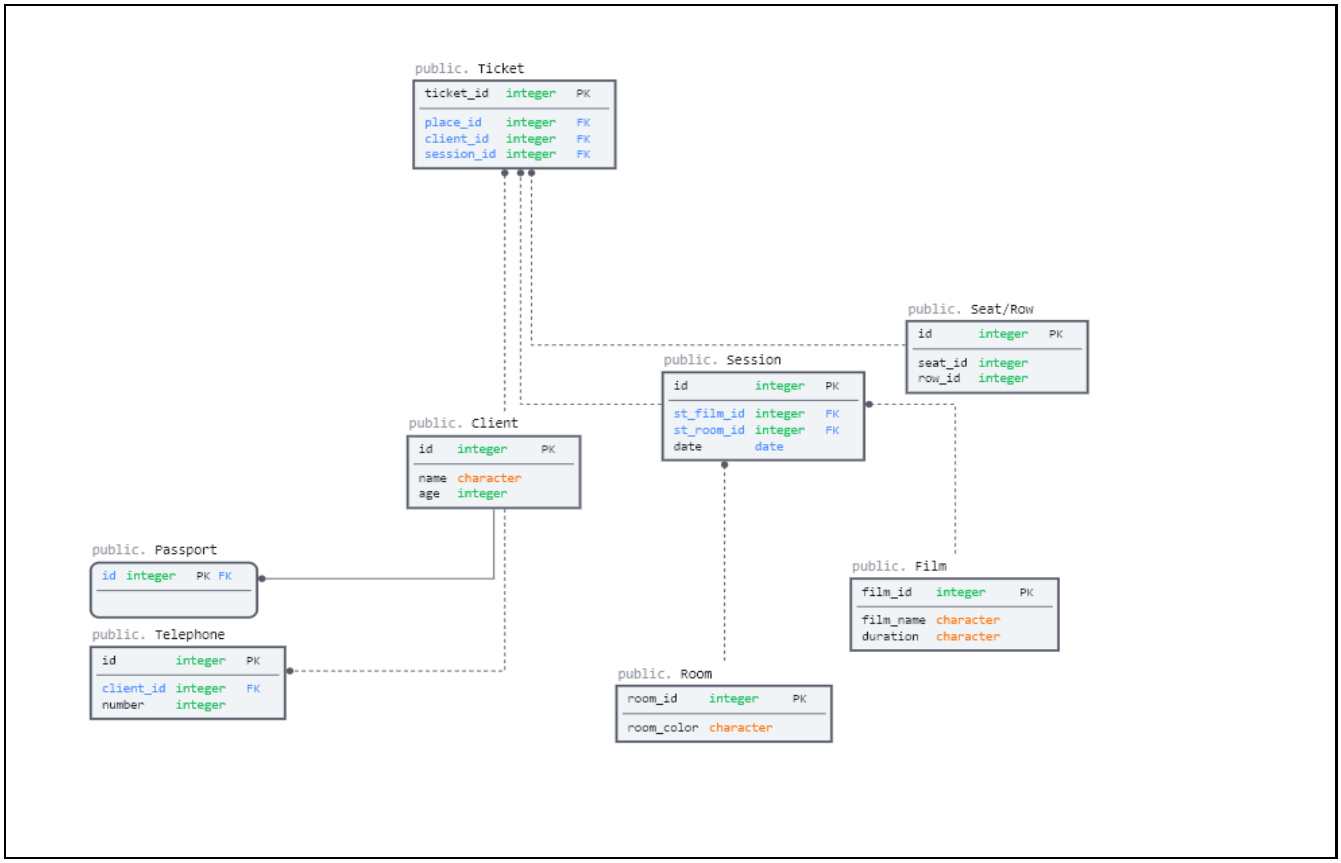


Рисунок 2 - Логічна модель (Структура) БД “ Сервіс продажу квитків кіно ” (засобами SqlDMB)

Опис структури БД “ Сервіс продажу квитків кіно ”

СУТНІСТЬ	АТРИБУТ	ТИП(Розмір)
Сутність *“**Room**”* Вміщує інформацію про залу кінотеатру	**id** – унікальний ID категорії в БД **room_colour** – колір зали в кінотеатрі для розпізнавання(зроблено на прикладі кінозалів Butterfly у м.Київ)	Числовий Текстовий(30)
Сутність *“**Session**”* Вміщує інформацію про сеанс у кінотеатрі	**id** - унікальний ID сеансу **date** – дата проходження заходу **room_id** – ID зали, де буде проходити сеанс **film_id** – ID фільму, що буде проходити під час сеансу.	Числовий Дата Числовий Числовий
Сутність *“**Film**”* Вміщує інформацію про фільм	**id** – унікальний ID фільму **name** – ім’я фільму **duration** – протяжність фільму	Числовий Текстовий(30) Текстовий(30)
Сутність *“**Seat/Row**”* *Вміщує інформацію про місце сидіння*	**id** – унікальний ID місця **seat_id** – ID місця для сидіння **row_id** – ID ряду для сидіння	Числовий Числовий Числовий
Сутність *“**Ticket**”* *Вміщує інформацію щодо білету у кіно*	**ticket_id** – унікальний ID білету **client_id** – ID клієнта **place_id** - ID місця	Числовий Числовий Числовий
Сутність *“**Client**”* Вміщує у собі інформацію про покупця	**id** - унікальний ID клієнта **name** – ім’я клієнта **age** – вік клієнта	Числовий Текстовий(30) Числовий
Сутність “***Phone*** ” Вміщує інформацію щодо телефонів покупців	**id** – унікальний ID мобільного номера **client_id** - ID клієнта чий телефон **number** – номер телефону	Числовий Числовий Числовий
Сутність “**Passport**” Вміщує інформацію паспортних даних користувача	**id** унікальний ID паспортних даних користувача	Числовий

Додаток Б1. Структура БД " Сервіс продажу квитків кіно "

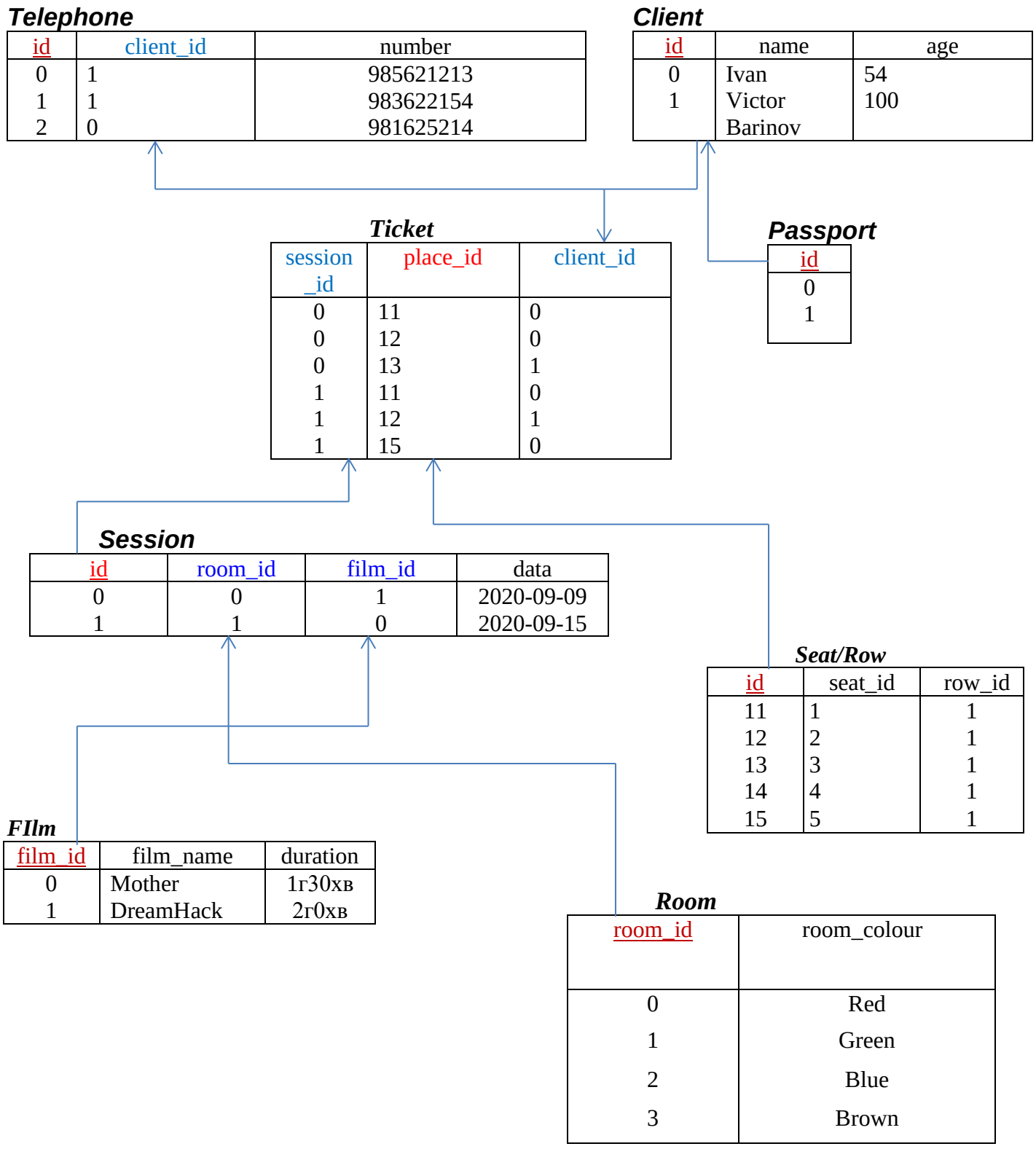
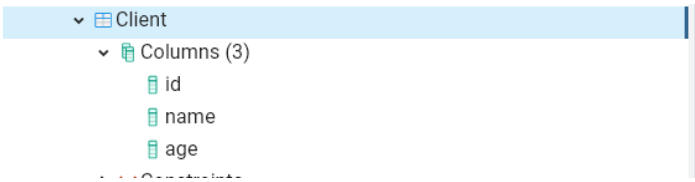


Рисунок 3 - Структура БД " Сервіс продажу квитків кіно "

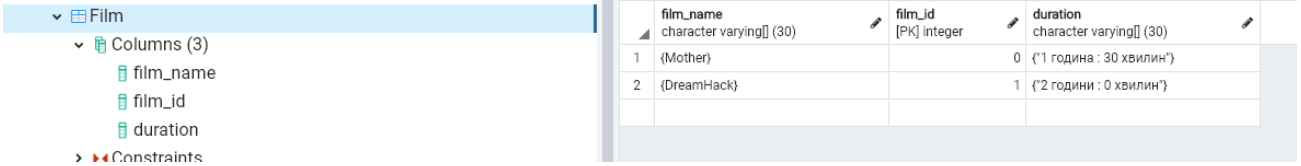
Структура БД “ Сервіс продажу квитків ” в pgAdmin IV



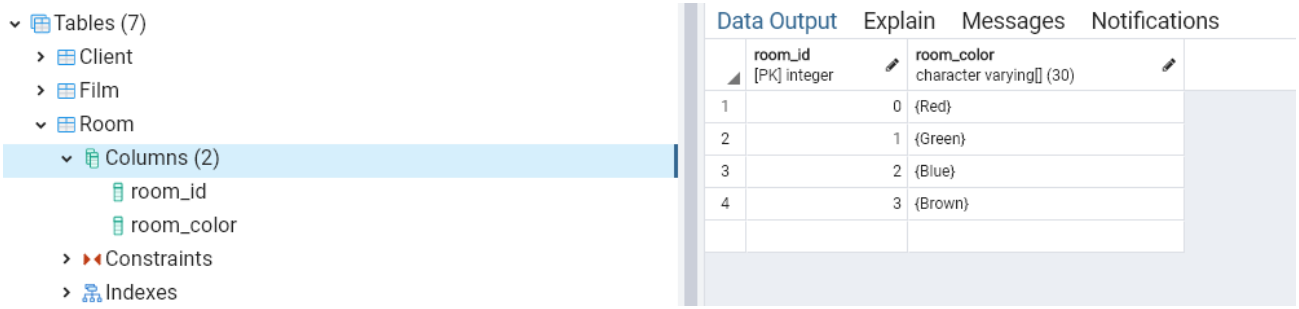
Data Output | Explain | Messages | Notifications

	id [PK] integer	name character varying[] (30)	age integer
1	0	{Ivan}	54
2	1	{"Victor Barinov"}	100

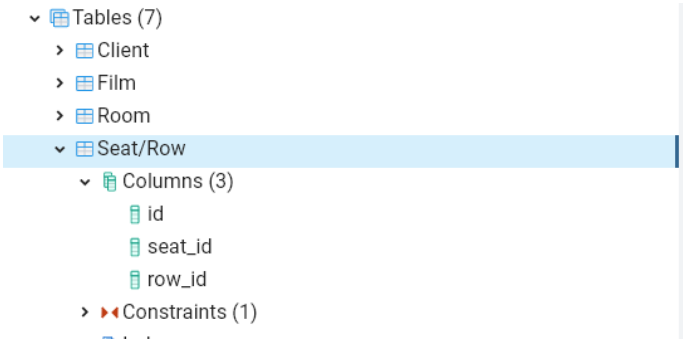
```sql
-- Table: public.Client

-- DROP TABLE public."Client";

CREATE TABLE public."Client"
(
    id integer NOT NULL,
    name character varying(30)[] COLLATE pg_catalog."default" NOT NULL,
    age integer NOT NULL,
    CONSTRAINT "Client_pkey" PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE public."Client"
    OWNER to postgres;
```




```sql
-- Table: public.Film

-- DROP TABLE public."Film";

CREATE TABLE public."Film"
(
    film_name character varying(30)[] COLLATE pg_catalog."default" NOT NULL,
    film_id integer NOT NULL DEFAULT nextval('"Film_film_id_seq"'::regclass),
    duration character varying(30)[] COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Film_pkey" PRIMARY KEY (film_id)
)

TABLESPACE pg_default;

ALTER TABLE public."Film"
    OWNER to postgres;
```



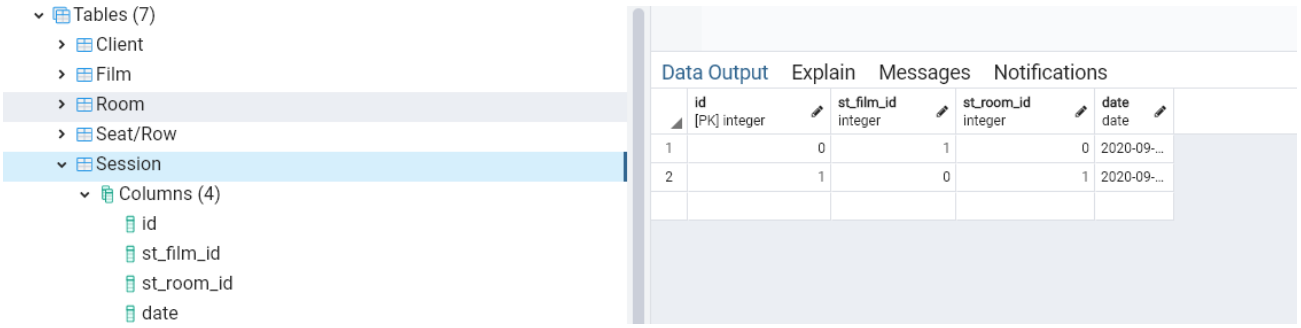

```sql
-- Table: public.Room

-- DROP TABLE public."Room";

CREATE TABLE public."Room"
(
    room_id integer NOT NULL DEFAULT nextval('"Room_room_id_seq"'::regclass),
    room_color character varying(30)[] COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT "Room_pkey" PRIMARY KEY (room_id)
)

TABLESPACE pg_default;

ALTER TABLE public."Room"
    OWNER to postgres;
```



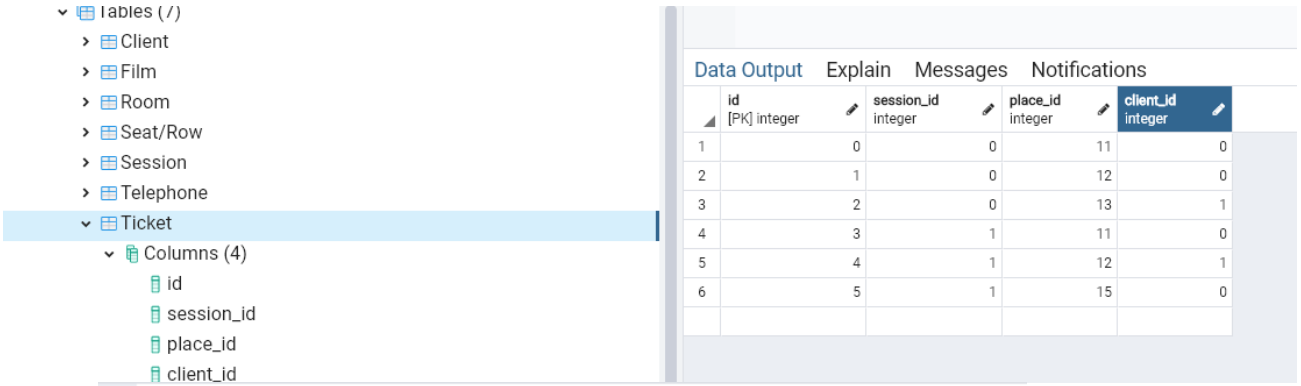
Data Output | Explain | Messages | Notifications

	id [PK] integer	seat_id integer	row_id integer
1	11	1	1
2	12	2	1
3	13	3	1
4	14	4	1
5	15	5	1

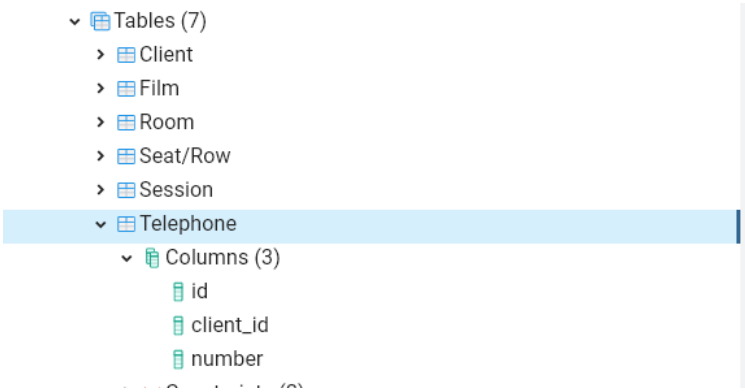
```sql
-- Table: public.Seat/Row

-- DROP TABLE public."Seat/Row";

CREATE TABLE public."Seat/Row"
(
    id integer NOT NULL,
    seat_id integer NOT NULL,
    row_id integer NOT NULL,
    CONSTRAINT "Seat/Row_pkey" PRIMARY KEY (id)
)

TABLESPACE pg_default;

ALTER TABLE public."Seat/Row"
    OWNER to postgres;
```




```sql
-- Table: public.Session

-- DROP TABLE public."Session";

CREATE TABLE public."Session"
(
    id integer NOT NULL DEFAULT nextval('"ShowTime_id_seq"'::regclass),
    st_film_id integer NOT NULL DEFAULT nextval('"ShowTime_film_id_seq"'::regclass),
    st_room_id integer NOT NULL DEFAULT nextval('"ShowTime_room_id_seq"'::regclass),
    date date NOT NULL,
    CONSTRAINT "ShowTime_pkey" PRIMARY KEY (id),
    CONSTRAINT st_film_id FOREIGN KEY (st_film_id)
        REFERENCES public."Film" (film_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT st_room_id FOREIGN KEY (st_room_id)
        REFERENCES public."Room" (room_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public."Session"
    OWNER to postgres;
-- Index: fki_st_film_id

-- DROP INDEX public.fki_st_film_id;

CREATE INDEX fki_st_film_id
    ON public."Session" USING btree
    (st_film_id ASC NULLS LAST)
    TABLESPACE pg_default;
-- Index: fki_st_room_id

-- DROP INDEX public.fki_st_room_id;

CREATE INDEX fki_st_room_id
    ON public."Session" USING btree
    (st_room_id ASC NULLS LAST)
    TABLESPACE pg_default;
```



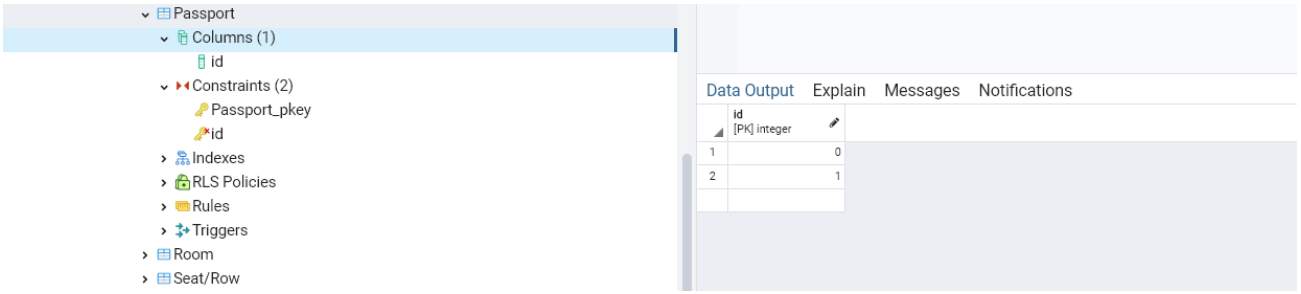

```sql
-- Table: public.Ticket

-- DROP TABLE public."Ticket";

CREATE TABLE public."Ticket"
(
    place_id integer NOT NULL,
    session_id integer NOT NULL,
    client_id integer NOT NULL,
    CONSTRAINT "Ticket_pkey" PRIMARY KEY (place_id),
    CONSTRAINT client_id FOREIGN KEY (client_id)
        REFERENCES public."Client" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT place_id FOREIGN KEY (place_id)
        REFERENCES public."Seat/Row" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT session_id FOREIGN KEY (session_id)
        REFERENCES public."Session" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public."Ticket"
    OWNER to postgres;
-- Index: fki_session_id

-- DROP INDEX public.fki_session_id;

CREATE INDEX fki_session_id
    ON public."Ticket" USING btree
    (session_id ASC NULLS LAST)
    TABLESPACE pg_default;
```



Data Output | Explain | Messages | Notifications

	id [PK] integer	client_id integer	number integer
1	0	1	931645372
2	1	1	931993038
3	2	0	977298400

```sql
-- Table: public.Telephone

-- DROP TABLE public."Telephone";

CREATE TABLE public."Telephone"
(
    id integer NOT NULL,
    client_id integer NOT NULL,
    "number" integer NOT NULL,
    CONSTRAINT "Telephone_pkey" PRIMARY KEY (id),
    CONSTRAINT client_id FOREIGN KEY (client_id)
        REFERENCES public."Client" (id) MATCH FULL
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public."Telephone"
    OWNER to postgres;
-- Index: fki_client_id

-- DROP INDEX public.fki_client_id;

CREATE INDEX fki_client_id
    ON public."Telephone" USING btree
    (client_id ASC NULLS LAST)
    TABLESPACE pg_default;
```




```sql
-- Table: public.Passport

-- DROP TABLE public."Passport";

CREATE TABLE public."Passport"
(
    id integer NOT NULL,
    CONSTRAINT "Passport_pkey" PRIMARY KEY (id),
    CONSTRAINT id FOREIGN KEY (id)
        REFERENCES public."Client" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE public."Passport"
    OWNER to postgres;
```