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 III курсу
групи КВ-91
Бубнов Ілля
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 – сайт місцевого кінотеатру під відкритим небом.

При проектуванні бази даних «сайт кінотеатру» можна виділити наступні сутності: відомості про фільм (movie), відомості про конкретний сеанс (session), відомості про акаунт покупця (account), відомості про робочого, який буде відповідальний за показ фільму (worker).

Атрибути заданих сутностей:

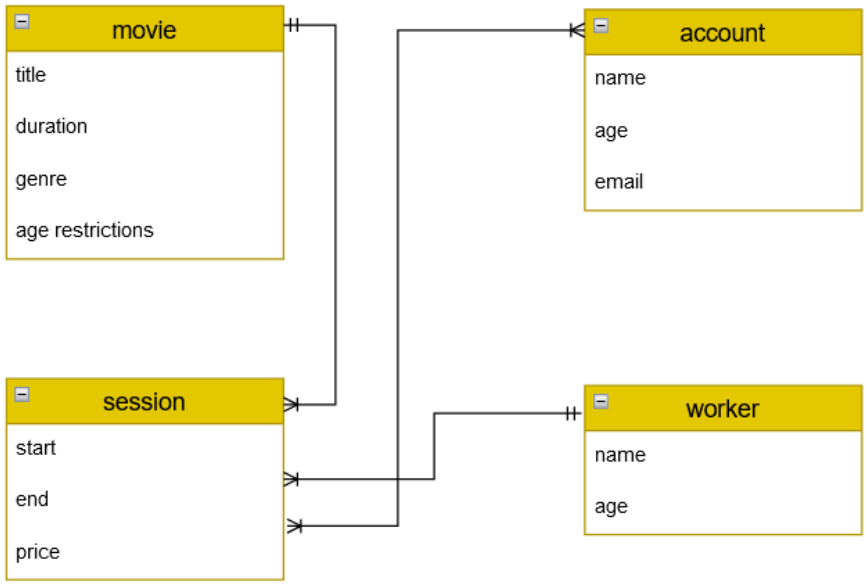
1. movie: title, duration, genre, age restrictions
2. session: start, end, price
3. account: name, age, email
4. worker: name, age

Опис зв’язків

Конкретний фільм може бути показаний на різних сеансах, тож між сутностями movie і session зв’язок 1:N

На нашому сайті не продають квитки, оскільки кінотеатр під відкритим небом передбачає перегляд кіно з машини, тож прив’язки до місць в залі нема, а попасти на фільм можна шляхом онлайн запису (бронювання) на сеанс. Тож на конкретний сеанс можуть записатися багато глядачів, також конкретний аккаунт глядача може бути записаний на багато сеансів, тож між сутностями session і account зв’язок N:M

Конкретний робочий може бути відповідальний за багато сеансів, однак на один сеанс достатньо одного працівника, тож між сутностями worker і session зв’язок 1:N.



ER-діаграма побудована за нотацією “Пташиної лапки (Crow’s foot)”, задана ER-діаграма була побудована у додатку draw.io

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:

1. 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.

2. 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.

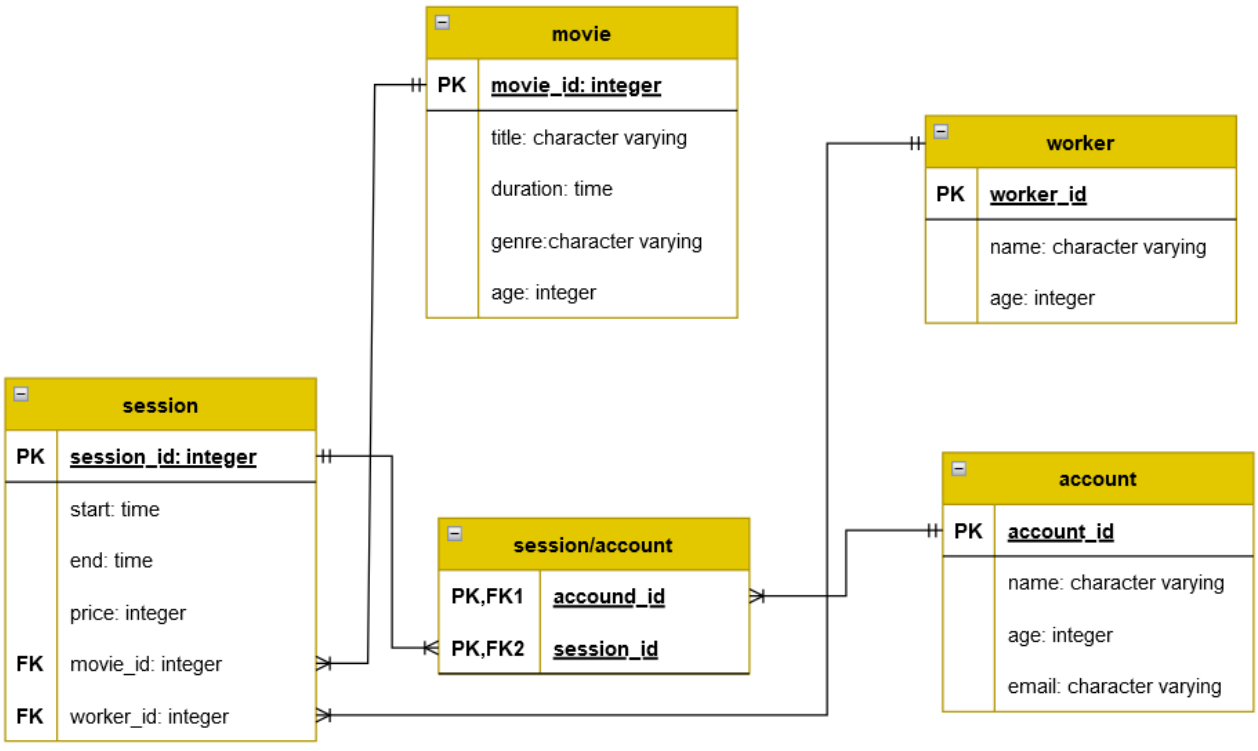
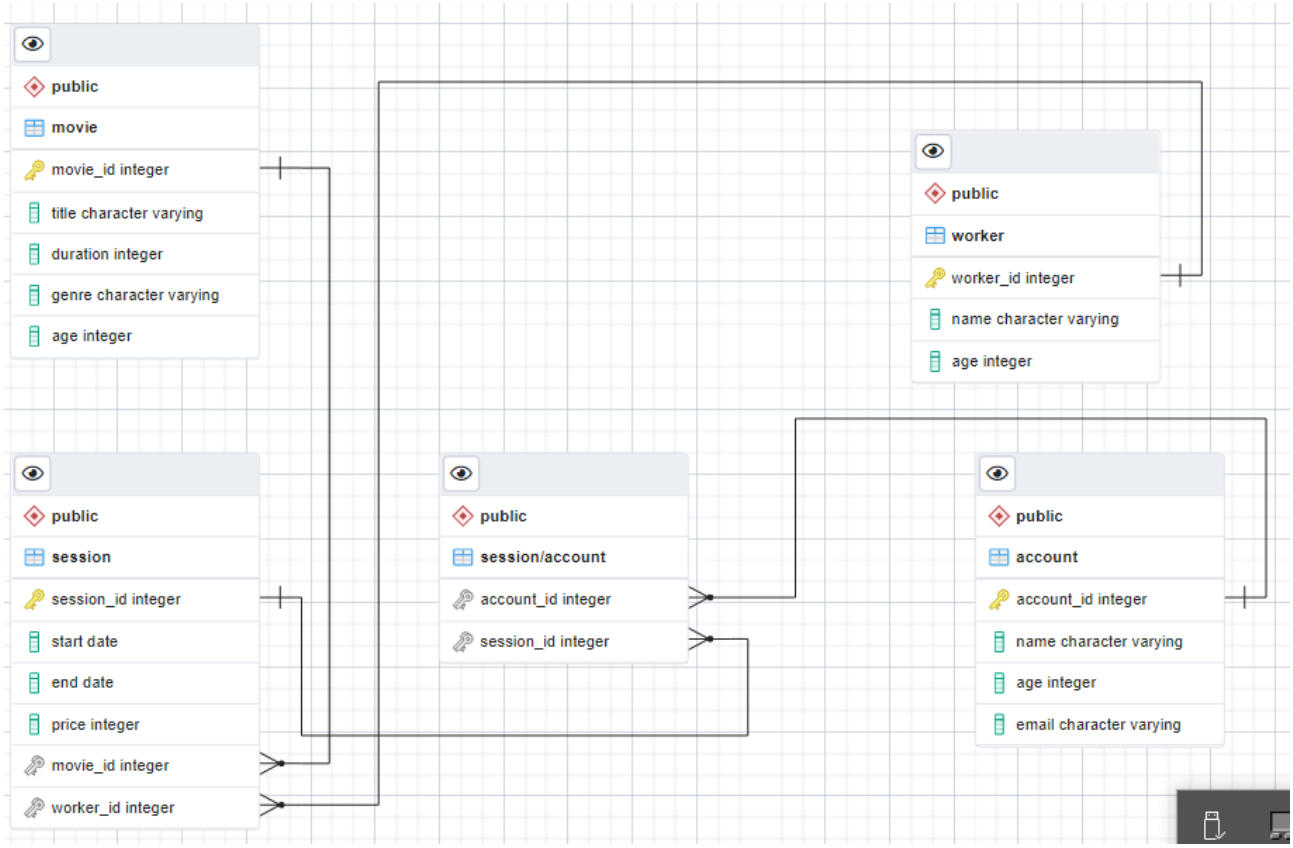


Схема бази даних, побудовано у додатку draw.io

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



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

```
-- Database: cinema


-- DROP DATABASE cinema;


CREATE DATABASE cinema
    WITH
    OWNER = postgres
    ENCODING = 'UTF8'
    LC_COLLATE = 'Ukrainian_Ukraine.1251'
    LC_CTYPE = 'Ukrainian_Ukraine.1251'
    TABLESPACE = pg_default
    CONNECTION LIMIT = -1;


-- Table: public.account


-- DROP TABLE public.account;


CREATE TABLE IF NOT EXISTS public.account
(
    account_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1
START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    name character varying COLLATE pg_catalog."default" NOT NULL,
    age integer NOT NULL,
    email character varying COLLATE pg_catalog."default" NOT NULL,
    CONSTRAINT account_pkey PRIMARY KEY (account_id)
)


TABLESPACE pg_default;


ALTER TABLE public.account
    OWNER to postgres;


-- Table: public.movie
```

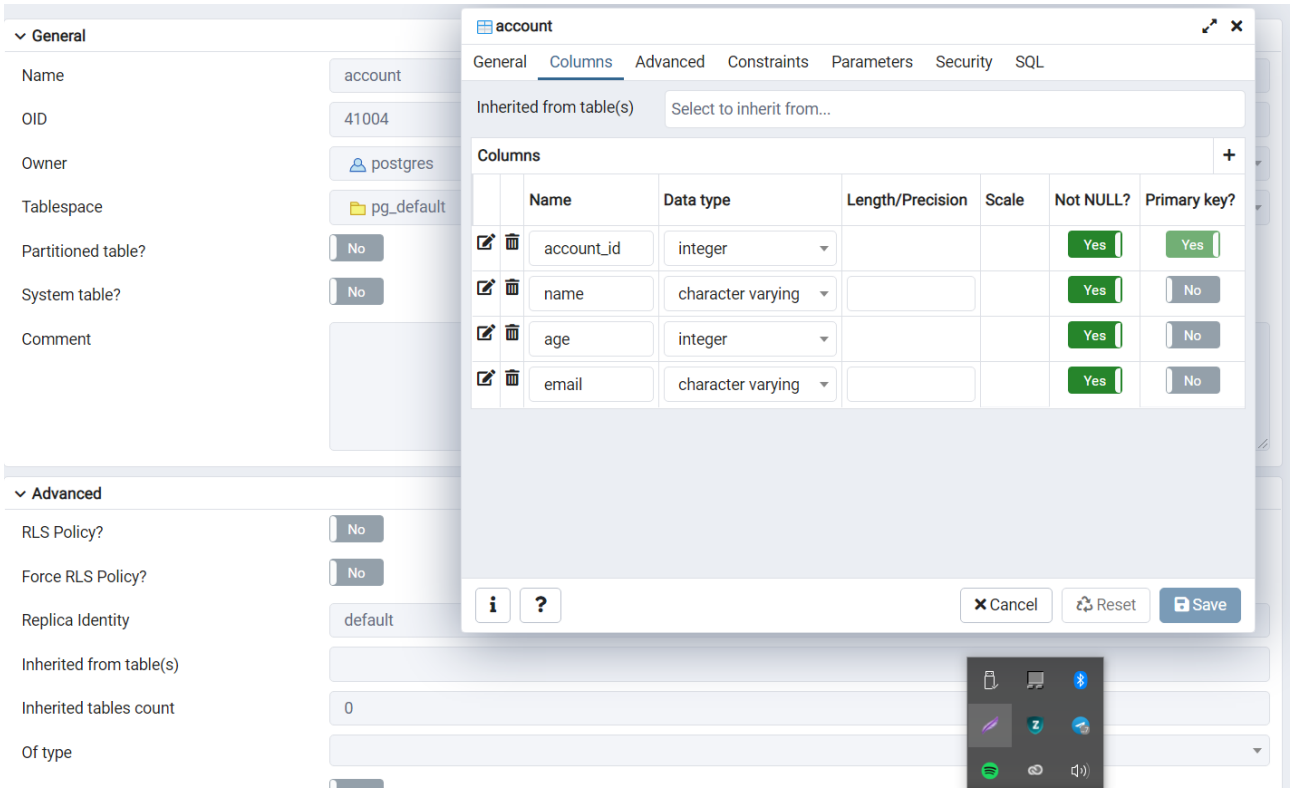
```
-- DROP TABLE public.movie;


CREATE TABLE IF NOT EXISTS public.movie
(
    movie_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    title character varying COLLATE pg_catalog."default" NOT NULL,
    duration integer NOT NULL,
    genre character varying COLLATE pg_catalog."default" NOT NULL,
    age integer NOT NULL,
    CONSTRAINT movie_pkey PRIMARY KEY (movie_id)
)

TABLESPACE pg_default;

ALTER TABLE public.movie
    OWNER to postgres;

-- Table: public.session

-- DROP TABLE public.session;

CREATE TABLE IF NOT EXISTS public.session
(
    session_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    start date NOT NULL,
    "end" date NOT NULL,
    price integer NOT NULL,
    movie_id integer NOT NULL,
    worker_id integer NOT NULL,
    CONSTRAINT session_pkey PRIMARY KEY (session_id),
    CONSTRAINT fk_movie_id FOREIGN KEY (movie_id)
        REFERENCES public.movie (movie_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_worker_id FOREIGN KEY (worker_id)
```

```
        REFERENCES public.worker (worker_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE public.session
    OWNER to postgres;

-- Table: public.session/account

-- DROP TABLE public."session/account";

CREATE TABLE IF NOT EXISTS public."session/account"
(
    account_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1
START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    session_id integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1
START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    CONSTRAINT fk_account_id FOREIGN KEY (account_id)
        REFERENCES public.account (account_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION,
    CONSTRAINT fk_session_id FOREIGN KEY (session_id)
        REFERENCES public.session (session_id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
)

TABLESPACE pg_default;

ALTER TABLE public."session/account"
    OWNER to postgres;

-- Table: public.worker
```

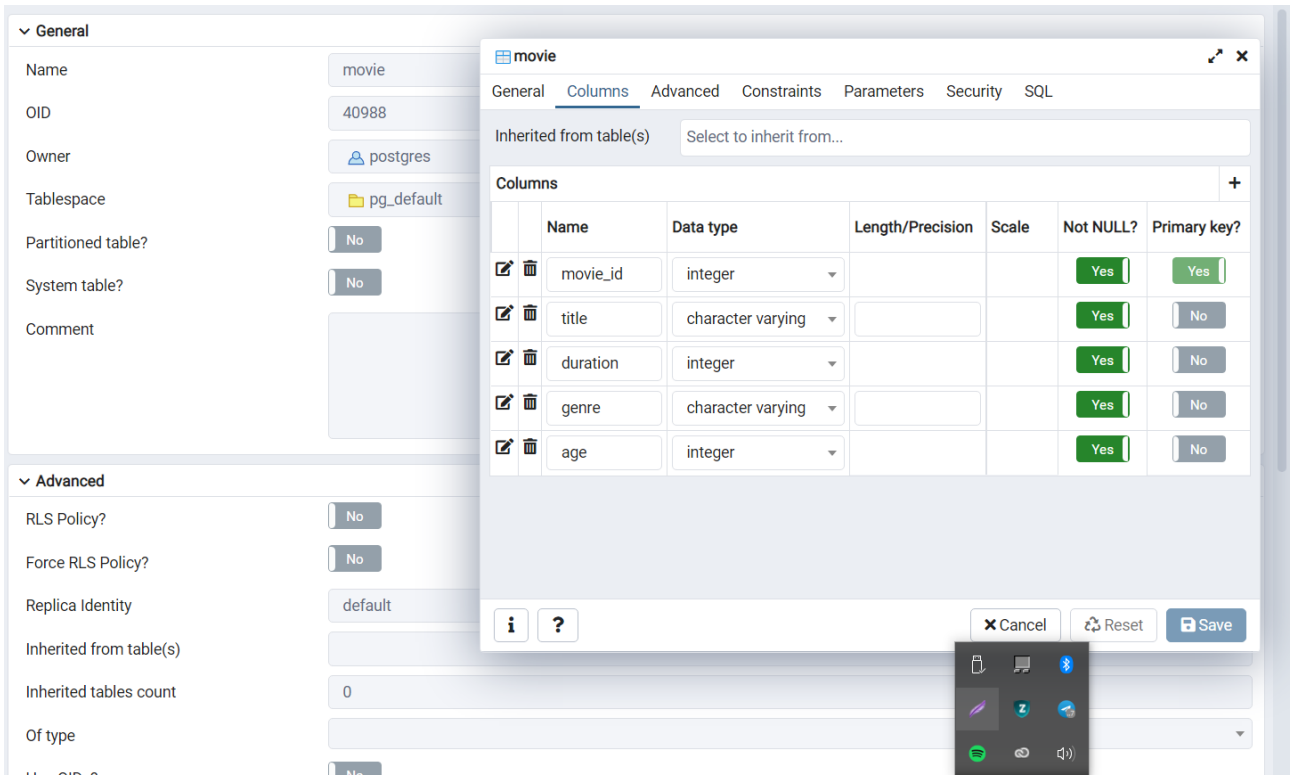
```
-- DROP TABLE public.worker;


CREATE TABLE IF NOT EXISTS public.worker

(

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

    name character varying COLLATE pg_catalog."default" NOT NULL,

    age integer NOT NULL,

    CONSTRAINT worker_pkey PRIMARY KEY (worker_id)

)


TABLESPACE pg_default;


ALTER TABLE public.worker

    OWNER to postgres;
```

Таблиці бази даних у pgAdmin 4

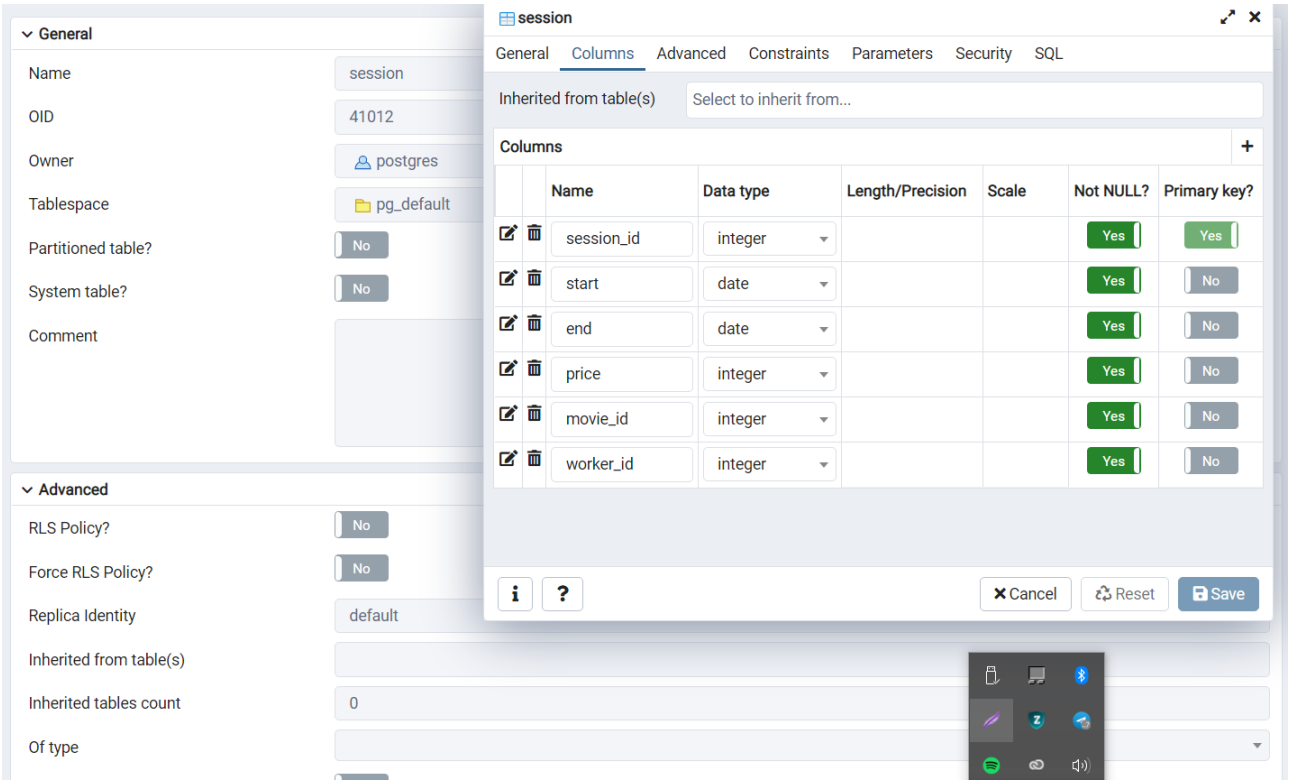
Account



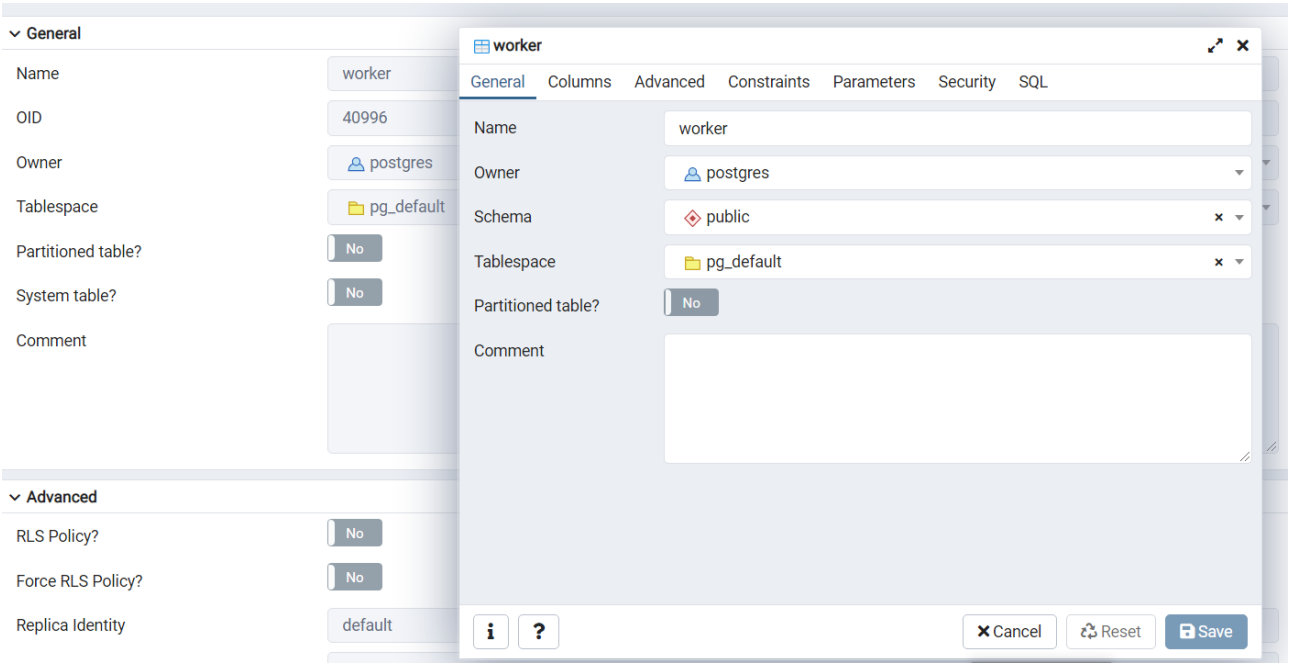
Movie



Session



Worker



Session / account