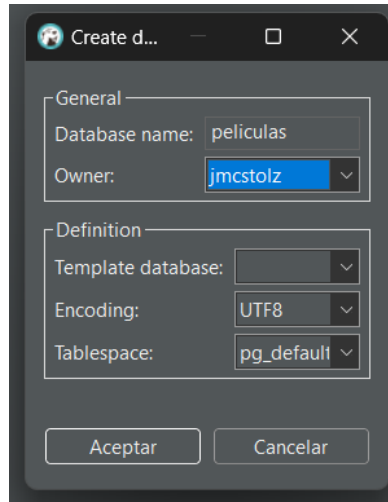


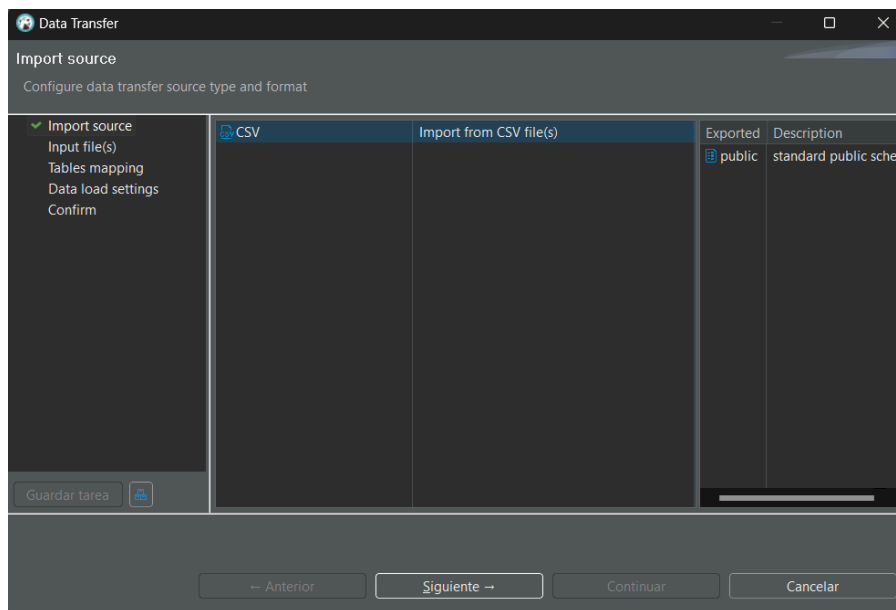
DESAFÍO EVALUADO - Definición de Tablas


José Contreras Stoltze
28-03-2024


Primero se crea la BD utilizando Dbeaver y postgresql.



Se importan los archivos .csv adjuntos para el desafío.

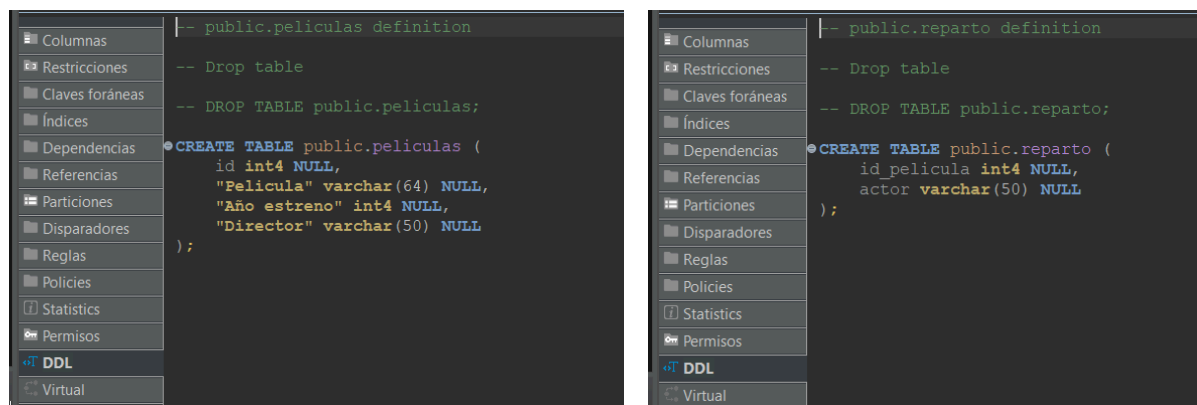


 peliculas.csv

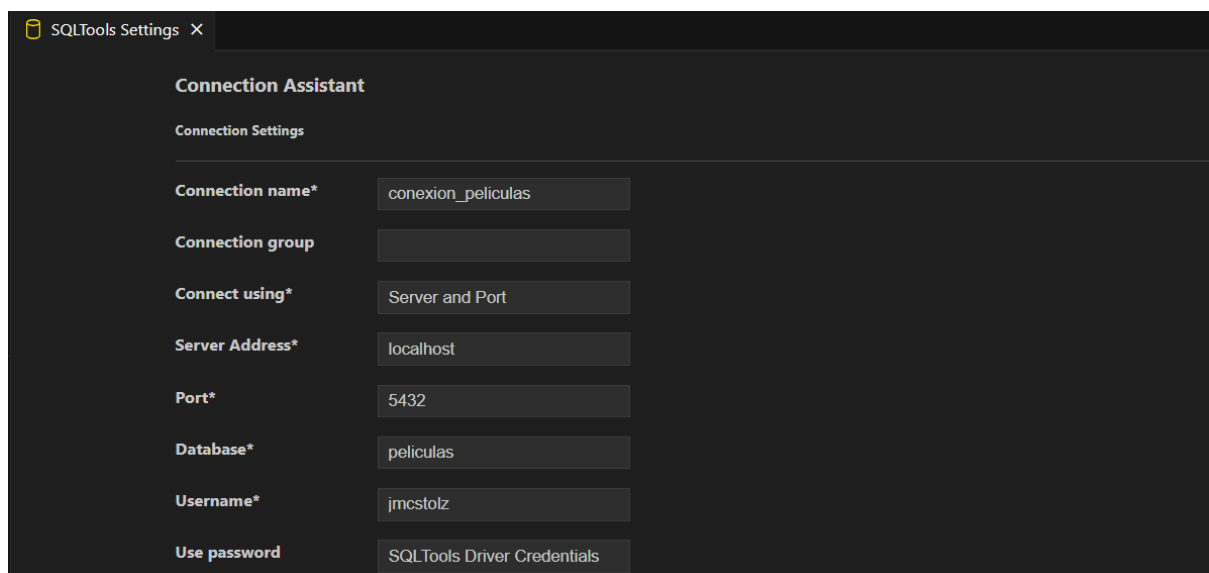
 reparto.csv



Una vez creadas las tablas, podemos ver cómo se ha definido su estructura.



Luego, trabajamos con Visual Studio Code, estableciendo una conexión con la BD.



En archivo .sql adjunto se presentan todas las consultas, incluidas los DDL de la generación de la base de datos y tablas sólo a modo de complemento, ya que no se ejecutarán nuevamente, a no ser que se quiera repetir el proceso desde cero (requerimientos 1 y 2).

```
conexion_peliculas.session.sql X
conexion_peliculas.session.sql
  ▶ Run on active connection | ≡ Select block
1
2  -- 1. Crea base de datos 'peliculas'
3  CREATE DATABASE peliculas;
4
5  -- 2.1 Crea la tabla reparto (importando csv en dbeaver)
6  CREATE TABLE public.peliculas (
7      id int4 NULL,
8      "Película" varchar(64) NULL,
9      "Año estreno" int4 NULL,
10     "Director" varchar(50) NULL
11 );
12
13 -- 2.2 Crea la tabla reparto (importando csv en dbeaver)
14 CREATE TABLE public.reparto (
15     id_pelicula int4 NULL,
16     actor varchar(50) NULL
17 );
```

Requerimiento 3 retorna el id de la película titulada "Titanic".

```
20
21  -- 3. Obtener ID de la película "Titanic"
22  -- (Se complementa con el título)
23  SELECT id, "Película"
24  FROM peliculas
25  WHERE "Película" = 'Titanic';
26
```

```
{...} consulta3.sql.json > ...
1  [
2      {
3          "id": 2,
4          "Película": "Titanic"
5      }
6  ]
```

Requerimiento 4 retorna todos los actores del reparto de la misma película utilizando una subconsulta.

```
26
27 -- 4. Listar a todos los actores que aparecen en la película Titanic
28 SELECT actor
29 FROM reparto
30 WHERE id_pelicula in (
31     SELECT id
32     FROM peliculas
33     WHERE "Pelicula" = 'Titanic'
34 );
35
```

{...} consulta4.sql.json > ...

```
1  [
2    {
3      "actor": "Leonardo DiCaprio"
4    },
5    {
6      "actor": "Kate Winslet"
7    },
8    {
9      "actor": "Billy Zane"
10   },
11   {
12     "actor": "Kathy Bates"
13   },
14   {
15     "actor": "Frances Fisher"
16   },
17   {
18     "actor": "Bernard Hill"
19   },

```

El requerimiento 5 consulta cuántas películas tienen a Harrison Ford como actor. Lo realiza a través de un INNER JOIN.

```
35
36 -- 5. Consultar en cuántas de las películas top 100 participa Harrison Ford
37 SELECT COUNT(*) AS cantidad_peliculas
38 FROM peliculas
39 INNER JOIN reparto ON peliculas.id = reparto.id_pelicula
40 WHERE reparto.actor = 'Harrison Ford';
41
```

```
{...} consulta5.sql.json > ...
1  [
2    {
3      "cantidad_peliculas": "8"
4    }
5  ]
```

El requerimiento 6 muestra las películas estrenadas entre 1990 y 1999 ordenadas por título de manera ascendente (orden alfabético).

```
41
42 -- 6. Indicar las películas estrenadas entre los años 1990 y 1999 ordenadas por título de manera ascendente.
43 SELECT "Película",
44        "Año estreno"
45 FROM películas
46 WHERE "Año estreno" BETWEEN 1990 AND 1999
47 ORDER BY "Película" ASC;
48
```

```
{...} consulta6.json > ...
1  [
2    {
3      "Película": "American Beauty",
4      "Año estreno": 1999
5    },
6    {
7      "Película": "American History X",
8      "Año estreno": 1998
9    },
10   {
11     "Película": "Braveheart",
12     "Año estreno": 1995
13   },
14   {
15     "Película": "Cadena perpetua",
16     "Año estreno": 1994
17   },
18   {
19     "Película": "Eduardo Manostijeras",
20     "Año estreno": 1990
21   },
22   {
23     "Película": "El Padrino. Parte III",
24     "Año estreno": 1990
25   },
26 ]
```

Requerimiento 7 solicita mostrar la longitud de los títulos de películas en una columna con nombre definido.

```
48
49 -- 7. Mostrar los títulos con su longitud en columna 'longitud_titulo'
50 SELECT "Película",
51        |    LENGTH("Película") AS "longitud_titulo"
52 FROM películas;
53
```

```
{ } consulta7.json > ...
1  [
2    {
3      "Película": "Forest Gump",
4      "longitud_titulo": 11
5    },
6    {
7      "Película": "Titanic",
8      "longitud_titulo": 7
9    },
10   {
11     "Película": "El Padrino",
12     "longitud_titulo": 10
13   },
14   {
15     "Película": "Gladiator",
16     "longitud_titulo": 9
17   },
18   {
19     "Película": "El Señor de los anillos: El retorno del rey",
20     "longitud_titulo": 43
21   },
22   {
23     "Película": "El caballero oscuro",
24     "longitud_titulo": 19
25   },
26 ]
```

Por último, se selecciona la longitud mayor. Esto se realiza ordenando de mayor a menor y limitando la consulta a una fila.

```
53
54 -- 8. Consultar la longitud más grande entre todos los títulos
55 -- (Se complementa con el título)
56 SELECT "Película",
57        |    LENGTH("Película") AS "longitud_titulo"
58 FROM películas
59 ORDER BY "longitud_titulo" DESC
60 LIMIT 1;
```

```
{-} consulta8.sql,json > ...  
1  [  
2    {  
3      "Película": "Sweeney Todd: El barbero diabólico de la calle Fleet",  
4      "longitud_titulo": 52  
5    }  
6  ]
```

Saludos cordiales.