

Proyecto Base de Datos:NETFLIX(Base de Datos de una plataforma).



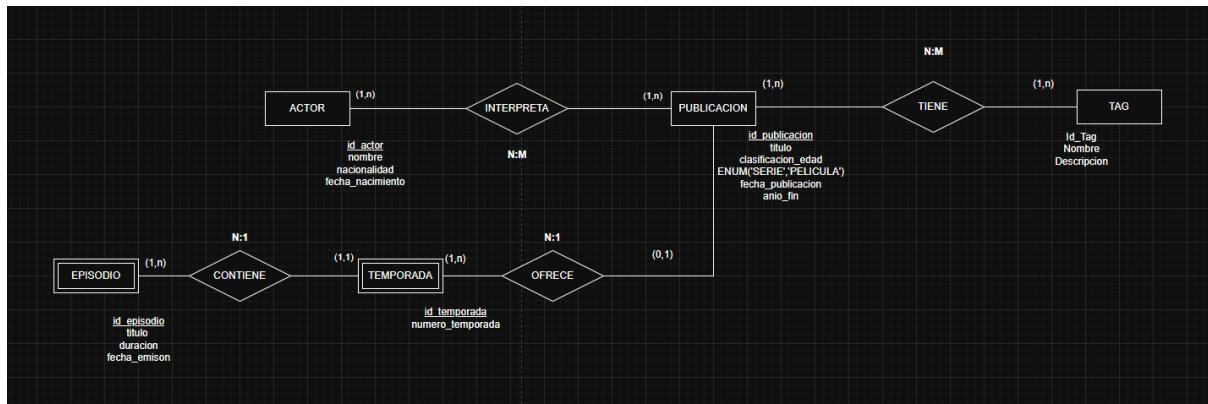
Índice:

1. Descripción.	2
2. Diagrama MER extendido.	2
3. Modelo Relacional.	3
4. Dbeaver.	4
5. Consultas Dbeaver.	5
6. Vistas Dbeaver.	11

1. Descripción.

El objetivo de crear una Base de Datos de Netflix consiste en reunir y organizar información sobre sus series, películas, los repartos de guión ... de forma que sea más fácil consultar y analizar sus datos. Con todo esto la plataforma puede ser más precisa y hace recomendaciones más exactas a cada usuario según lo que le gusta o no e ir mejorando periódicamente la experiencia para los que utilicen esta plataforma de streaming

2. Diagrama MER extendido.



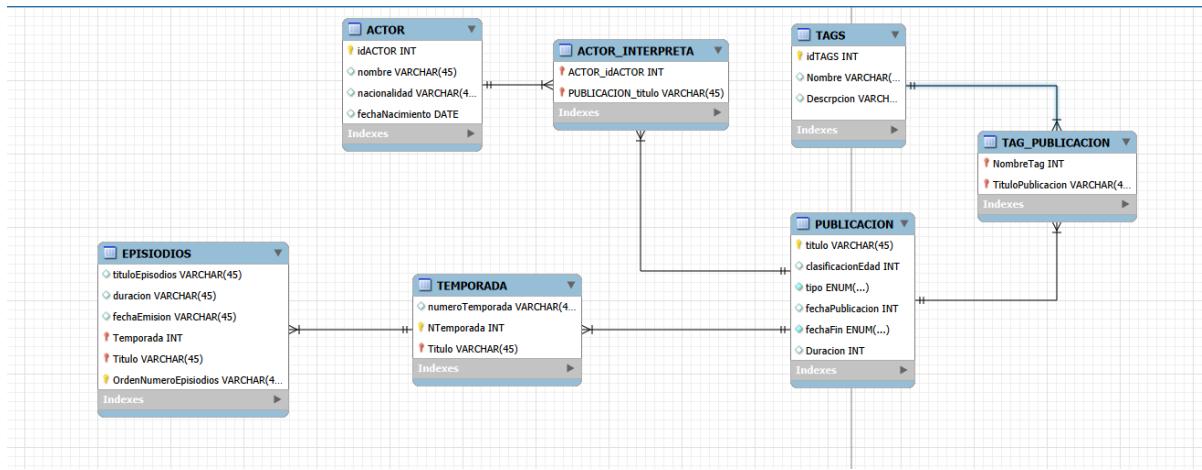
Como podemos ver tenemos aquí mi diagrama MER extendido en el contiene unas 5 entidades con 4 relaciones.

Entidades: Actor, Publicación, Tag, Temporada, Episodio.

Relaciones:

- Un actor interpreta una o varias publicaciones, y una publicaciones es interpretada por uno o varios actores.
- Una publicación puede tener una o varias etiquetas (Tag), y un tag puede pertenecer a una o varias publicaciones.
- Una publicación (Serie) ofrece una o varias temporadas, y una temporada pertenece a 0 o una publicación.
- Una temporada contiene uno o varios episodios, y un episodio pertenece a una temporada.

3. Modelo Relacional.

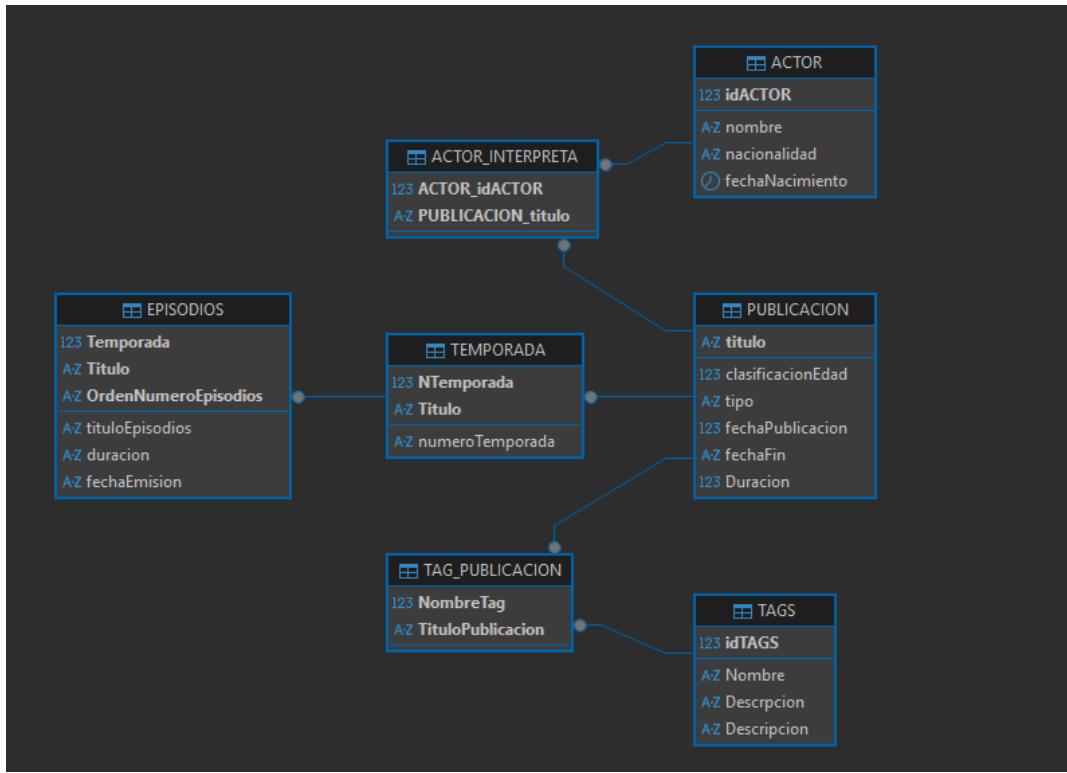


Aquí vemos el modelo MER extendido pasado al modelo Modelo relacional (a tablas). Como vemos han salido nuevas tablas gracias a las N:M del anterior modelo (Modelo MER extendido).

Las nuevas tablas son las siguientes:

- Actor_Interpreta.
- Tag_publicación.

4. Dbeaver.



5. Consultas Dbeaver.

Consulta 1:

```
-- 1.Mostrar las publicaciones con sus tags y cuántos actores participan (solo las primeras 15)

select p.titulo , p.tipo,
       COUNT(distinct tp.NombreTag) as num_tags,
       COUNT(distinct ai.ACTOR_idACTOR) as num_actores
  from PUBLICACION p
 left join TAG_PUBLICACION tp on tp.TituloPublicacion = p.titulo
 left join TAGS t on tp.NombreTag = t.idTAGS
 left join ACTOR_INTERPRETA ai on p.titulo = ai.PUBLICACION_titulo
 group by p.tipo, p.titulo
 order by num_tags desc
 limit 15;
```

PUBLICACION 1 ×

select p.titulo , p.tipo, COUNT(distinct tp.NombreTag) as num_tags, COUNT(distinct ai.ACTOR_idACTOR) as num_actores

	AZ titulo	AZ tipo	I23 num_tags	I23 num_actores
1	Pub 0247 Prohibida Juego	SERIE	10	5
2	Pub 0764 Ultima Leyenda	SERIE	9	3
3	Pub 0638 Sombria Horizonte	SERIE	9	1
4	Pub 0728 Rapida Ruta	SERIE	9	2
5	Pub 0884 Invisible Bosque	PELICULA	9	2
6	Pub 0334 Azul Clave	SERIE	9	1
7	Pub 0224 Dorada Bosque	PELICULA	9	4
8	Pub 0010 Perdida Mision	PELICULA	9	2
9	Pub 0003 Prohibida Clave	SERIE	8	5
10	Pub 0667 Silenciosa Trama	PELICULA	8	5
11	Pub 0112 Eterna Destino	SERIE	8	1
12	Pub 0275 Antigua Destino	PELICULA	8	3
13	Pub 0381 Oscura Ruta	PELICULA	8	2
14	Pub 0190 Extraña Señal	PELICULA	8	1
15	Pub 0959 Ultima Trama	SERIE	8	1

Consulta 2:

```
-- 2.Cuántas publicaciones hay de cada tipo (SERIE o PELICULA).
SELECT
  tipo,
  COUNT(*) AS total
FROM PUBLICACION
GROUP BY tipo;

-- 3.Sacar los actores que participan en más publicaciones que la media
SELECT
```

PUBLICACION 1 ×

SELECT tipo, COUNT(*) AS total FROM PUBLICACION GROUP BY tipo

	AZ tipo	I23 total
1	PELICULA	481
2	SERIE	519

Consulta 3:

The screenshot shows the MySQL Workbench interface. On the left, there's a toolbar with various icons. The main area has a query editor window with the following SQL code:

```
•-- 3.Sacar los actores que participan en más publicaciones que la media.
SELECT
    a.idACTOR,
    a.nombre,
    COUNT(*) AS total_publicaciones
FROM ACTOR a
JOIN ACTOR_INTERPRETA ai
    ON ai.ACTOR_idACTOR = a.idACTOR
GROUP BY a.idACTOR, a.nombre
HAVING COUNT(*) > (
    SELECT AVG(x.cnt)
    FROM (
        SELECT COUNT(*) AS cnt
        FROM ACTOR_INTERPRETA
        GROUP BY ACTOR_idACTOR
    ) x
)
ORDER BY total_publicaciones DESC;
```

Below the query editor is a results grid titled "ACTOR 1". The grid has three columns: "idACTOR", "nombre", and "total_publicaciones". The data is as follows:

	idACTOR	nombre	total_publicaciones
1	525	Carmen Ramos	12
2	178	Maria Navarro	11
3	342	Elena Muñoz	11
4	83	Maria Vazquez	10
5	94	Sergio Castro	10
6	117	Maria Garcia	10
7	290	Noelia Castro	10
8	309	Ana Ruiz	10
9	423	Javier Gil	10
10	440	Sofia Hernandez	10
11	13	Claudia Garcia	9
12	56	Ana Lopez	9
13	166	David Moreno	9
14	176	Carlos Romero	9
15	187	Marta Dominguez	9
16	222	Marta Hernandez	9
17	252	Sergio Romero	9
18	269	Carlos Gomez	9
19	269	Sofia Torres	0

Consulta 4:

```
● -- 4.Mostrar actores con su edad aproximada, nacionalidad en mayúsculas y cuántas publicaciones interpretan (si ninguna, 0).
SELECT
    a.idACTOR,
    a.nombre,
    UPPER(a.nacionalidad) AS nacionalidad,
    TIMESTAMPDIFF(YEAR, a.fechaNacimiento, CURDATE()) AS edad,
    IFNULL(COUNT(ai.PUBLICACION_titulo), 0) AS publicaciones
FROM ACTOR a
LEFT JOIN ACTOR INTERPRETA ai
    ON ai.ACTOR_idACTOR = a.idACTOR
GROUP BY a.idACTOR, a.nombre, a.nacionalidad, a.fechaNacimiento
ORDER BY publicaciones DESC
LIMIT 20;
● -- 5.Listar series y cuántas temporadas tiene cada una.
SELECT
```

ACTOR 1

Gilla	i23 idACTOR	AZ nombre	AZ nacionalidad	i23 edad	i23 publicaciones
1	525	Carmen Ramos	ITALIA	47	12
2	178	Maria Navarro	PORTUGAL	61	11
3	342	Elena Muñoz	ALEMANIA	38	11
4	117	Maria Garcia	CANADA	30	10
5	423	Javier Gil	COLOMBIA	46	10
6	440	Sofia Hernandez	PORTUGAL	42	10
7	94	Sergio Castro	ALEMANIA	36	10
8	83	Maria Vazquez	ESPAÑA	32	10
9	290	Noelia Castro	MEXICO	52	10
10	309	Ana Ruiz	CHILE	44	10
11	222	Marta Hernandez	CANADA	31	9
12	450	Sofia Alvarez	REINO UNIDO	43	9
13	13	Claudia Garcia	ARGENTINA	58	9
14	368	Sofia Torres	ESTADOS UNIDOS	60	9
15	176	Carlos Romero	REINO UNIDO	41	9
16	166	David Moreno	ESPAÑA	56	9
17	252	Sergio Romero	ITALIA	48	9
18	56	Ana Lopez	COLOMBIA	60	9
19	260	Carlos Gomez	ITALIA	60	9

Consulta 5:

- -- 5.Listar series y cuántas temporadas tiene cada una.

```
SELECT p.titulo AS serie,
       COUNT(*) AS temporadas
  FROM PUBLICACION p
  JOIN TEMPORADA t
    ON t.Titulo = p.titulo
 WHERE p.tipo = 'SERIE'
 GROUP BY p.titulo
 ORDER BY temporadas DESC
LIMIT 20;
```
- -- 6.Para cada serie y temporada, cuántos episodios hay.

```
SELECT
```

PUBLICACION 1 X

SELECT p.titulo AS serie, COUNT(*) AS temporadas FROM

Enter a SQL expression to filter results (use Ctrl+Space)

	AZ → serie	123 temporadas
1	Pub 0021 Antigua Horizonte	5
2	Pub 0061 Nueva Juego	5
3	Pub 0159 Infinita Ritual	5
4	Pub 0154 Azul Plan	5
5	Pub 0126 Eterna Plan	5
6	Pub 0107 Nueva Sueño	5
7	Pub 0033 Secreta Hielo	5
8	Pub 0101 Perdida Hielo	5
9	Pub 0167 Invisible Fuego	5
10	Pub 0053 Roja Sueño	5
11	Pub 0077 Dorada Sombra	5
12	Pub 0020 Rapida Hielo	5
13	Pub 0157 Perdida Sombra	5
14	Pub 0018 Dorada Ruta	5
15	Pub 0180 Valiente Destino	5
16	Pub 0168 Prohibida Fuego	5
17	Pub 0156 Valiente Señal	5
18	Pub 0105 Extraña Ruta	5

Consulta 6:

```
• -- 6.Para cada serie y temporada, cuántos episodios hay.  
SELECT  
    e.Titulo AS serie,  
    e.Temporada,  
    COUNT(*) AS episodios  
FROM EPISODIOS e  
GROUP BY e.Titulo, e.Temporada  
ORDER BY episodios DESC  
LIMIT 20;  
....  
• -- 7.Top 10 tags más usados en publicaciones.  
SELECT  
    t.idTAGS,  
    t.Nombre.
```

EPISODIOS 1 ×

Enter a SQL expression to filter results (use Ctrl+Space)

Guilla	AZ serie	123 Temporada	123 episodios
Texto	1 Pub 0174 Secreta Ruta	1	12
Record	2 Pub 0096 Sombria Bosque	1	12
	3 Pub 0041 Roja Plan	1	12
	4 Pub 0043 Antigua Mision	1	12
	5 Pub 0214 Azul Hielo	1	12
	6 Pub 0198 Ultima Ritual	1	12
	7 Pub 0033 Secreta Hielo	1	12
	8 Pub 0049 Salvaje Señal	1	12
	9 Pub 0027 Sombria Horizonte	1	12
	10 Pub 0022 Secreta Fuego	1	12
	11 Pub 0199 Antigua Mision	1	12
	12 Pub 0101 Perdida Hielo	1	12
	13 Pub 0019 Invisible Sombra	1	12
	14 Pub 0185 Perdida Clave	1	12
	15 Pub 0016 Sagrada Bosque	1	12
	16 Pub 0151 Rapida Sombra	1	12
	17 Pub 0059 Dorada Señal	1	12
	18 Pub 0078 Silenciosa Plan	1	12

Consulta 7:

```
•-- 7.Top 10 tags más usados en publicaciones.  
SELECT  
    t.idTAGS,  
    t.Nombre,  
    COUNT(*) AS veces_usado  
FROM TAGS t  
JOIN TAG_PUBLICACION tp  
    ON tp.NombreTag = t.idTAGS  
GROUP BY t.idTAGS, t.Nombre  
ORDER BY veces_usado DESC  
LIMIT 10;  
  
•-- VISTAS  
-- Vista 1: publicaciones con número de tags y act  
CREATE OR REPLACE VIEW v_publicacion_resumen AS  
  
TAGS 1 X  
SELECT idTAGS, Nombre, COUNT(*) AS veces_usado FF | Enter a SQL expression to filter results (u  


| idTAGS | Nombre              | veces_usado |
|--------|---------------------|-------------|
| 1      | Crimen_459          | 12          |
| 2      | Thriller_368        | 12          |
| 3      | Thriller_513        | 11          |
| 4      | Suspense_165        | 11          |
| 5      | Terror_42           | 11          |
| 6      | Drama_162           | 11          |
| 7      | Terror_239          | 11          |
| 8      | Ciencia Ficcion_367 | 10          |
| 9      | Drama_292           | 10          |
| 10     | Familia_124         | 10          |


```

6. Vistas Dbeaver.

Vista 1:

```
CREATE OR REPLACE VIEW v_publicacion_resumen AS
SELECT
    p.titulo,
    p.tipo,
    COUNT(DISTINCT tp.NombreTag) AS num_tags,
    COUNT(DISTINCT ai.ACTOR_idACTOR) AS num_actores
FROM PUBLICACION p
LEFT JOIN TAG_PUBLICACION tp
    ON tp.TituloPublicacion = p.titulo
LEFT JOIN ACTOR_INTERPRETA ai
    ON ai.PUBLICACION_titulo = p.titulo
GROUP BY p.titulo, p.tipo;
```

```
-- Vista 1: publicaciones con número de tags y actores
CREATE OR REPLACE VIEW v_publicacion_resumen AS
SELECT
    p.titulo,
    p.tipo,
    COUNT(DISTINCT tp.NombreTag) AS num_tags,
    COUNT(DISTINCT ai.ACTOR_idACTOR) AS num_actores
FROM PUBLICACION p
LEFT JOIN TAG_PUBLICACION tp
    ON tp.TituloPublicacion = p.titulo
LEFT JOIN ACTOR_INTERPRETA ai
    ON ai.PUBLICACION_titulo = p.titulo
GROUP BY p.titulo, p.tipo;
```

Estadísticas 1

Name	Value
Updated Rows	0
Execute time	0,229s
Start time	Sat Feb 14 13:39:25 CET 2026
Finish time	Sat Feb 14 13:39:25 CET 2026
Query	-- VISTAS -- Vista 1: publicaciones con número de tags y actores CREATE OR REPLACE VIEW v_publicacion_resumen AS SELECT p.titulo, p.tipo, COUNT(DISTINCT tp.NombreTag) AS num_tags, COUNT(DISTINCT ai.ACTOR_idACTOR) AS num_actores FROM PUBLICACION p LEFT JOIN TAG_PUBLICACION tp ON tp.TituloPublicacion = p.titulo LEFT JOIN ACTOR_INTERPRETA ai ON ai.PUBLICACION_titulo = p.titulo GROUP BY p.titulo, p.tipo

Vista 2:

```
CREATE OR REPLACE VIEW v_series_temporadas AS
SELECT
    p.titulo AS serie,
    COUNT(*) AS temporadas
FROM PUBLICACION p
JOIN TEMPORADA t
    ON t.Titulo = p.titulo
WHERE p.tipo = 'SERIE'
GROUP BY p.titulo;
```

The screenshot shows a database interface with a query editor and a statistics panel.

Query Editor:

```
-- Vista 2: series y número de temporadas
CREATE OR REPLACE VIEW v_series_temporadas AS
SELECT
    p.titulo AS serie,
    COUNT(*) AS temporadas
FROM PUBLICACION p
JOIN TEMPORADA t
    ON t.Titulo = p.titulo
WHERE p.tipo = 'SERIE'
GROUP BY p.titulo;
```

Statistics Panel:

Name	Value
Updated Rows	0
Execute time	0,208s
Start time	Sat Feb 14 13:40:25 CET 2026
Finish time	Sat Feb 14 13:40:25 CET 2026
Query	-- Vista 2: series y número de temporadas CREATE OR REPLACE VIEW v_series_temporadas AS SELECT p.titulo AS serie, COUNT(*) AS temporadas FROM PUBLICACION p JOIN TEMPORADA t ON t.Titulo = p.titulo WHERE p.tipo = 'SERIE' GROUP BY p.titulo

7. Valoración del Proyecto.

Este proyecto me ha ayudado a entender mejor cómo diseñar y trabajar con una base de datos relacional desde cero. He aprendido a crear tablas correctamente, establecer relaciones entre ellas y realizar consultas multitablea con JOIN, GROUP BY y subconsultas. También he comprendido la importancia de definir bien las claves primarias y foráneas para evitar errores.

En general, me ha servido para reforzar mis conocimientos de SQL y ganar más confianza trabajando con bases de datos en un entorno real.