# MEMORIA

## PRÁCTICA 2 - BASE DE DATOS DE CINE

### Asignatura: Bases de datos

**Grupo J2.04, curso 2023-2024**

**Fecha de entrega : 08/05/2024**


**PARTICIPANTES:**

**ATHANASIOS USERO SAMARÁS, EMAIL : 839543@unizar.es**
**CURRO VALERO CASAJÚS , EMAIL : 842545@unizar.es**
**DANIEL RAMÓN MONTERRUBIO, EMAIL: 872428@unizar.es**

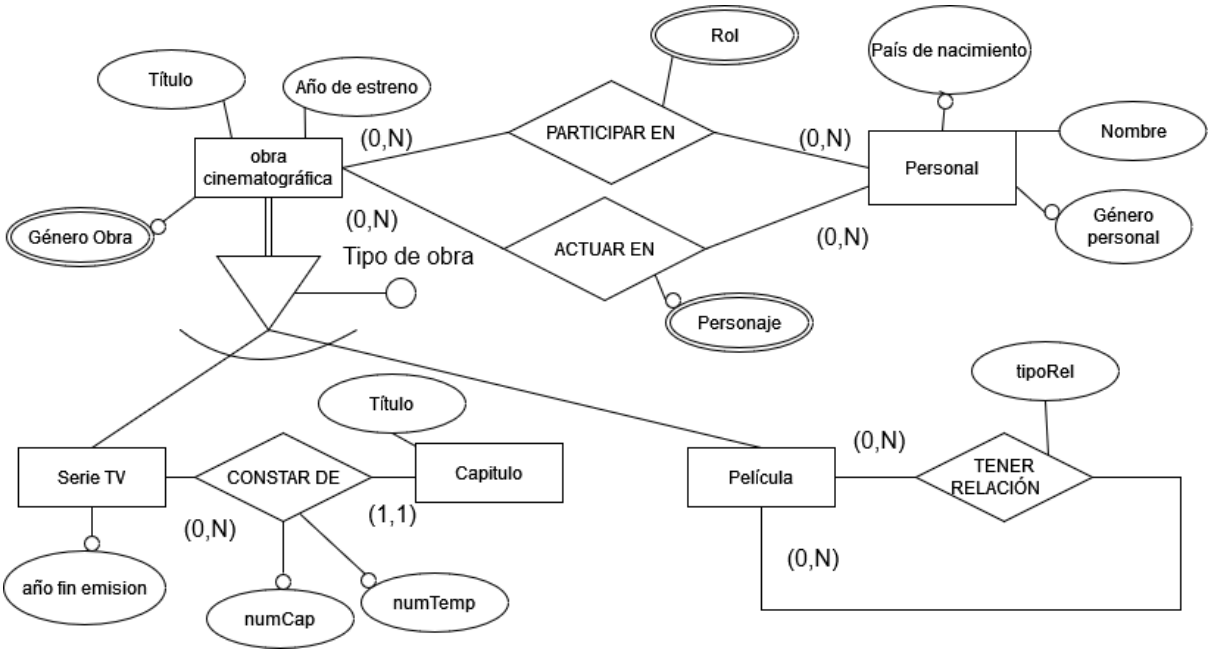
# Índice

# PARTE 1. Creación de una base de datos

## 1.1. Esquema E/R

Antes de nada, se debe mencionar una decisión, que no premisa, acerca del modelado conceptual de la base de datos y su relación con el universo de discurso. A diferencia de la anterior práctica, donde no se tomó en cuenta propiamente el lote de datos hasta la propia población, en la presente ocasión sí que se realizó un análisis previo para diseñar el modelo conceptual. Esto se debe a que, en este caso, el universo de discurso (el universo cinematográfico) contiene variables más ambiguas, que no se pueden esclarecer meramente por el enunciado, y para las que puede servir estudiar *muestras* de lo que este supone. Algunas de las consultas más relevantes, y que se pueden ir mencionando a lo largo de la memoria, se encuentran en el Anexo 3. Consultas adicionales.

El esquema E/R global diseñado es el siguiente:

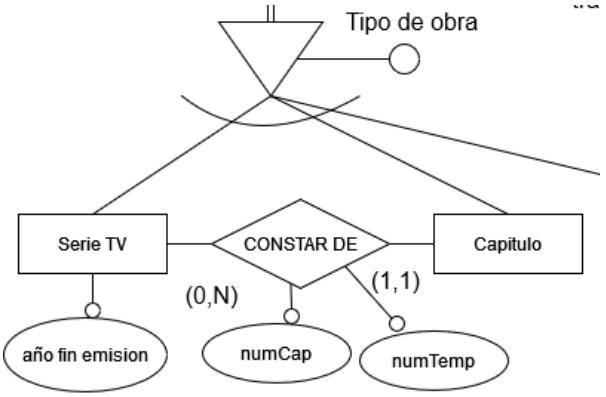



- **RESTRICCIONES:**
  1. Una película no puede tener relación consigo misma.
  2. Los tipos de relaciones entre películas pueden ser *precuelas, secuelas o remakes.*
  3. La película que es sujeto de una relación (la que es secuela, precuela o remake de otra) debe tener un fecha de estreno posterior a la referenciada.
  4. Sean *x, y, z* películas. Si *z* es secuela de *y*, e *y* secuela de *x,* entonces *z* es secuela de *x.* Análogamente con precuelas. Esta es una relación de transitividad.
  5. El año de estreno de toda obra cinematográfica debe ser superior o igual a 1895 (datación del comienzo del cine).
  6. El año de fin de emisión de una serie de TV debe ser igual o posterior a su año de estreno.
  7. Una serie no puede constar de dos capítulos con mismo número de capítulo y número de temporada.
  8. El rol en la relación *Participar en* tiene que ser diferente a actor.

- **PREMISAS:**
    1. Ningún miembro del personal se puede identificar por su nombre, el cual es su único atributo obligatorio. Es decir, es perfectamente posible la coincidencia de dos *Juan Rodriguez* (que además pueden aparecer con distintos lugares de nacimiento, confirmando que son distintas personas). No solo esto, sino que incluso pueden aparecer dos personas diferentes llamadas igual en una misma obra, como ocurre con los dos *Jose Nieto* en "*Hay que matar a B.*".
    2. Ninguna obra cinematográfica se puede identificar por su título y año de estreno, ni tampoco adicionalmente por el tipo de obra que constituye (así es que *Historias de la puta Mili,* estrenada en 1994, hace referencia tanto a una peli como a una serie en el lote de datos proporcionado).
    3. Se considera que el género del personal puede ser masculino o femenino, pero no ambos, y que no siempre tiene que ser conocido. En este sentido, hay 9746 nombres sin género asociado en el lote de datos entregados, frente a los 37884 totales (nombres, no personal). Además, no se tratan los roles con una perspectiva de género. En este sentido, con independencia de como se escriba, quien trabaja como director no trabaja propiamente como director, sino que lo hace con la particularidad de dirigir.
    4. Si una serie de TV no ha terminado de emitirse, se considera que su periodo de fin de emisión es desconocido, no el año actual. Algunas veces se suele actualizar anualmente con la fecha actual, pero este no es el caso.
    5. Se considera que una película no puede tener más de un tipo de relación con otra, algo condicionado por el conjunto de relaciones con el que se trabaja.

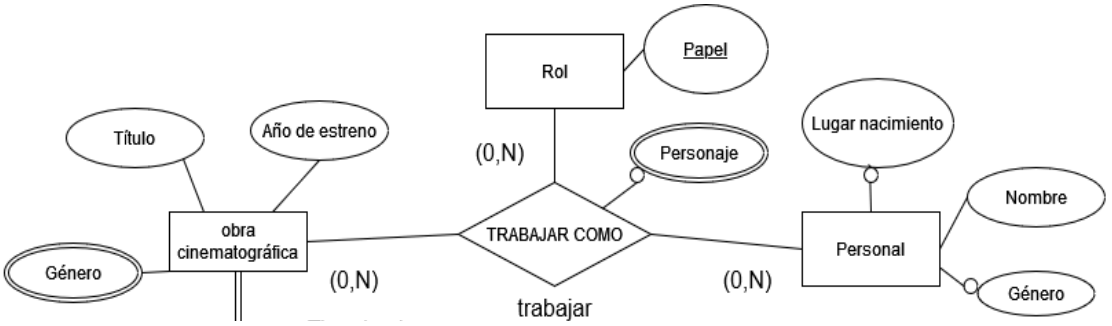
Ahora bien, hay bastantes partes del modelo E/R cuya resolución bien vino provista de discusión:

- Quizás una de las más sugerentes fue la idea del **capítulo**. Tal como se presenta dentro del enunciado, el capítulo es un elemento a catalogar. Es decir, cada serie tiene un listado de capítulos, con su título, número de capítulo y número de temporada. Ahora bien, tal como se ofrecen los datos, bien se podría considerar un capítulo como otra obra cinematográfica, puesto que aparece junto a su año de estreno, no el de la serie (90 capítulos emitidos con posterioridad al inicio de emisión de su serie aparecen así); y a su vez se acompaña con personajes que participaron en este capítulo en concreto. En este caso, el capítulo sería en cierto modo tanto obra como elemento a listar dentro de una serie:



Cabe tener presente que en este modelo aparecen nuevas restricciones, como que todo personaje que trabaja en un capítulo lo haga también en la serie, o que el año de estreno del capítulo se incluya en el periodo de emisión de la serie. En todo caso, como los capítulos dentro de nuestro contexto de uso no cobran apenas importancia (tan solo se ofrecen datos de 157 capítulos, frente a las 5518 obras de otros tipos que existen), se descartó esta aproximación y se aceptó la del enunciado, aún teniendo en cuenta que se pierde la profundidad de esa nueva dimensionalidad.

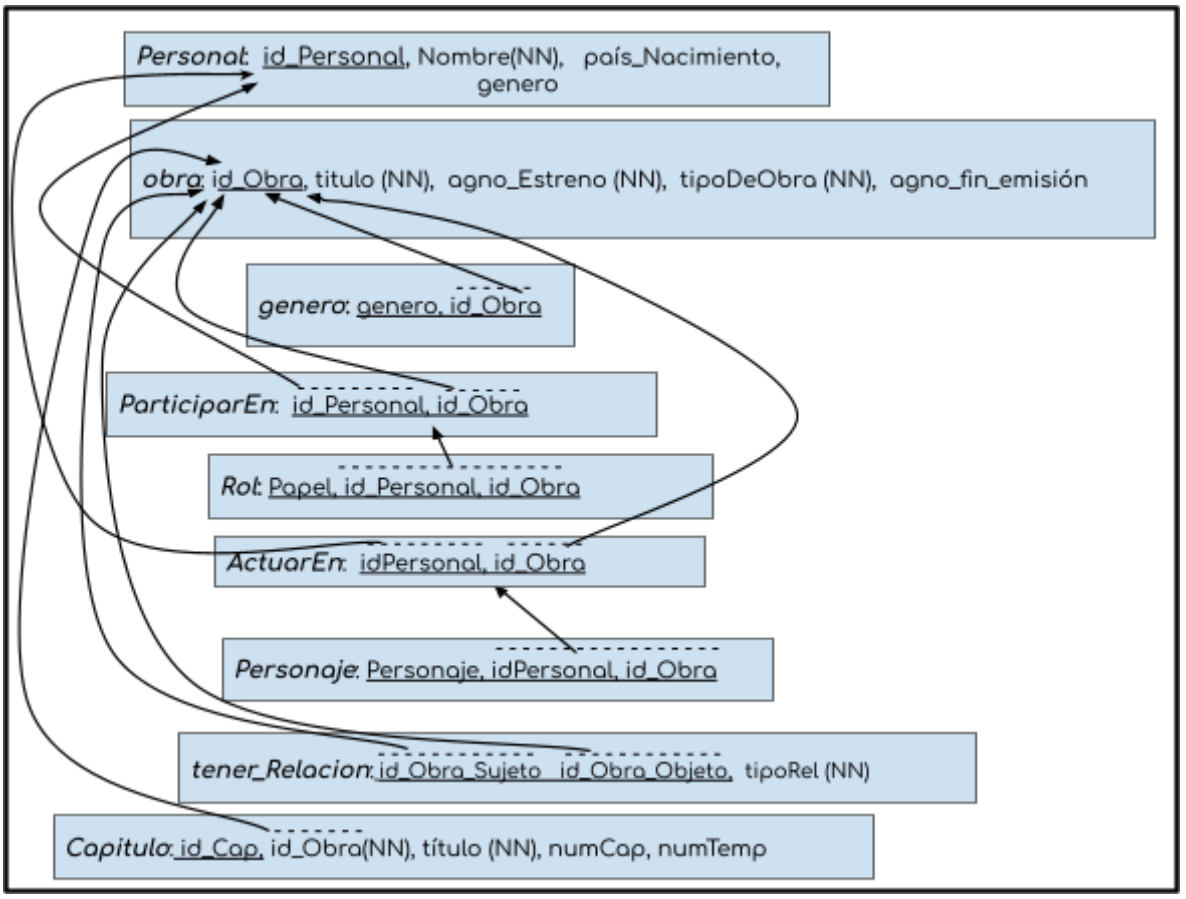
- En torno a las **relaciones entre *personal* y *obras cinematográficas*** también se plantearon diversas alternativas. En un principio, se propuso incluir una nueva entidad, *rol,* que participa en una relación ternaria con *personal* y *obra cinematográfica*, relación que cuenta con un atributo opcional multivaluado para personajes. Pero este modelado se descartó porque, además de que requiere una nueva restricción (ningún personal participando con un rol distinto al de actor puede tener un personaje), posteriormente, en el modelo relacional, revelaría que no se encuentra en cuarta forma normal.




- Incluso desde el definitivo modelado se plantearon diversas alternativas en lo que respecta a las **relaciones *Participar en* y *actuar en***. Por una parte, en vez de plantear el rol como un atributo multivaluado, se podría haber incluido como una entidad, siendo la relación así ternaria. Por otra parte, también se podía plantear el *personaje* como una entidad. Pero en este caso, su inclusión mediante una relación ternaria no es adecuada al ser su participación opcional, siendo una agregación la implementación más acertada. Por último, se debe indicar que la **relación *participar en* se considera alternativa a la de actuar.** Es decir, que un miembro del personal actúe en una peli se modela en la relación *Actuar En,* mientras que si participa en cualquier otro rol se verá reflejado en la relación *Participar En (*de ahí que el rol no pueda ser de actor en dicha relación*).*
- Por último, es notable la cantidad de **atributos opcionales** en el modelo. Dado el enunciado, por ejemplo, bien se podría haber indicado que el género era obligatorio. Y, posteriormente, en la creación de tablas, adaptarse asumiendo que el género no podía ser obligatorio ya que ello supondría descartar datos justamente no descartables (así como las consecuencias que ello conlleva). De aquí recalcar el enfoque ya expuesto al inicio: *que conocer las características de los datos a tratar (al menos en este caso) no equivale a reducir el concepto de cine a una colección concreta, sino a (ayudar a) inducirlo a partir de ella.*

# 1.2 Esquema relacional

El mapeado directo del modelo E/R a relacional es el siguiente:



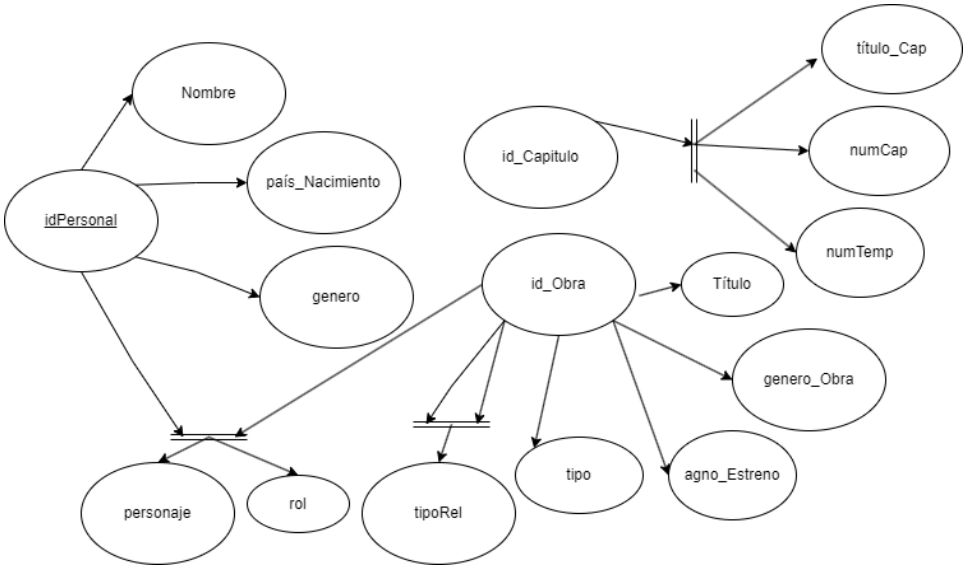

Este hereda todas las restricciones y premisas del modelo E-R. Asimismo, cabe destacar la **aparición de las nuevas relaciones** *Personaje, Rol y Genero*, que aparecían como atributos multivaluados de las relaciones *Actuar En, Participar En y* la entidad *Obra Cinematográfica,* pero que al transformar se han convertido en nuevas relaciones ya que el modelo relacional no admite atributos multivaluados. Con ellas aparece una nueva restricción, y es que toda ocurrencia (*id_Personal, id_Obra*) en *ParticiparEn* debe contar con al menos una ocurrencia en la tabla *Rol.*
Aparte, como en el modelo relacional toda relación requiere una clave primaria, y ni las obras, ni los capítulos, ni el personal se podían identificar naturalmente, se han creado **claves primarias artificiales** para *Personal, Obra* y *Capítulo.*

Por último, se debe realizar un análisis, aunque sea somero, del **mapeado de la especialización** de *Obra* en *Serie tv* o *Película.* Como se puede observar, se ha optado por mantener la entidad general, de manera que aparece el atributo *tipo (*tipo serie o película) y el campo para el año de fin de emisión es opcional. Consecuentemente, aparecen nuevas restricciones. El campo *agno_fin_emisión* deberá ser nulo en caso de que la obra sea una película. Además, ahora las relaciones de las entidades especializadas pasan a depender de la general. De este modo, se debe indicar que en *tener_Relacion, id_Obra_Sujeto* y *id_Obra_Objeto* deben hacer referencia a una obra de tipo película; mientras que en *Capítulo, id_Obra* referencia a una obra de tipo serie.

Otra opción era la de mantener las entidades especializadas, pero en este caso el modelo se podría complicar por tener que duplicarse la tabla de *Genero,* de *participarEn,* de *actuarEn* y los

atributos de la entidad *obra Cinematográfica* en el modelo E/R (aunque desaparecerían las restricciones anteriores). Además, se podía plantear también mantener todas las entidades, pero por ser la especialización exclusiva y obligatoria sería necesario asegurar que un mismo identificador no aparezca en una serie y película simultáneamente, además de que se necesitará un control en la población para que no se introduzca directamente sobre obra. Así, se decidió mantener la entidad general porque aunque es cierto que implica controlar las relaciones de la entidades especializadas, tan solo contiene un campo que será nulo en algunos casos, resuelve con facilidad la restricción de exclusividad y obligatoriedad, y además evitará tablas parcialmente redundantes.

En cuanto a la **normalización** del modelo, el esquema está en 1FN debido a que los atributos multivaluados son ajenos al modelo relacional. Además, construyendo un grafo de dependencias funcionales,




se puede comprobar asimismo que ningún atributo no clave depende funcionalmente de parte de la clave, por lo que se cumple la 2FN. Este solo podría incumplirse en la relación *tener_Relación,* pero claramente no lo hace, pues el tipo de una relación entre dos entidades depende de esas dos entidades. Además, el modelo se encuentra en 3FN. Cualquier duda al respecto se explica sencillamente con la *necesidad* de crear claves artificiales. Ni el nombre identifica al personal (por lo que no se puede inferir de un nombre un país de nacimiento), ni tampoco lo hace el título de un capítulo o el título y año de estreno de una obra. Por último, también se verifica la FNBC. En este aspecto, remarcar que una combinación de nombre, género y país de nacimiento puede representar a más de una persona, correspondiendo a más de un identificador. Así, el modelo inicial se ajusta a la normalización exigida.

## 1.3 Creación de tablas

```
Unset
/* Creación de tablas*/

-- Tabla del personal
CREATE SEQUENCE secP START WITH 1 INCREMENT BY 1;
CREATE TABLE personal(
       id_Personal NUMBER(10) PRIMARY KEY,
       nombre VARCHAR(70) NOT NULL,
       genero VARCHAR(1),
       pais_Nac VARCHAR(200),
       CONSTRAINT idPersonal_val CHECK (id_Personal > 0),
       CONSTRAINT genero_Valor CHECK(genero IN ('m', 'f'))
);


-- Tabla de Obras Cinematográficas
CREATE SEQUENCE secO START WITH 1 INCREMENT BY 1;
CREATE TABLE obraCinem(
       id_Obra NUMBER(10) PRIMARY KEY, titulo VARCHAR(200) NOT NULL,
       agno_Estreno number(4) NOT NULL, tipoDeObra VARCHAR(20) NOT NULL,
       agno_fin_Emision number(4),
       CONSTRAINT id_Obra_VAL CHECK (id_Obra > 0),
       CONSTRAINT periodoEmision_NULO CHECK((tipoDeObra = 'movie' AND
              agno_fin_Emision IS NULL) OR tipoDeObra = 'tv series'),
       CONSTRAINT agno_fin_Emision_VAL CHECK(agno_fin_Emision IS NULL OR
(agno_fin_Emision IS NOT NULL AND agno_fin_Emision >= agno_Estreno)),
       CONSTRAINT agnoEstreno_VAL CHECK (agno_Estreno >= 1895)
);

-- Tabla de géneros de obras cinematográficas
CREATE TABLE genero(
       genero VARCHAR(20),
       id_Obra NUMBER(10),
       CONSTRAINT genero_PK PRIMARY KEY (genero, id_Obra),
       CONSTRAINT obraGenero_FK FOREIGN KEY (id_Obra) REFERENCES
obraCinem(id_Obra)
                                                    ON DELETE CASCADE
);

-- Tabla de otros trabajos de personal con un rol dado en obras
CREATE TABLE participarEn(
       id_Personal NUMBER(10),
       id_Obra NUMBER(10),
       CONSTRAINT participarEn_PK PRIMARY KEY (id_personal, id_Obra),
       CONSTRAINT idPersonal_FK FOREIGN KEY (id_Personal)
```

```
              REFERENCES personal(id_Personal),
       CONSTRAINT obra_FK FOREIGN KEY (id_Obra)
              REFERENCES obraCinem(id_Obra)
);


-- Tabla de roles de una persona en una obra
CREATE TABLE rol(
       rol VARCHAR(30),
       id_Personal NUMBER(10),
       id_Obra NUMBER(10),
       CONSTRAINT rol_PK PRIMARY KEY (id_personal, rol, id_Obra),
       CONSTRAINT participarEn_FK FOREIGN KEY (id_Personal, id_Obra)
                     REFERENCES participarEn(id_Personal, id_Obra),
       CONSTRAINT act_NOVAL CHECK(rol <> 'actor' AND rol <> 'actress')
);
/* NO IMPLEMENTADA POR CUOTA DE ESPACIO
PARTITION BY LIST (rol)
(PARTITION pr1_Director VALUES('director'),
 PARTITION pr2_OtrosRoles VALUES(DEFAULT));*/

-- Tabla de participaciones como actores
CREATE TABLE actuarEn(
       id_Personal NUMBER(10),
       id_Obra NUMBER(10),
       CONSTRAINT actuarEn_PK PRIMARY KEY (id_personal, id_Obra),
       CONSTRAINT idAct_FK FOREIGN KEY (id_Personal)
       REFERENCES personal(id_Personal),
       CONSTRAINT obraAct_FK FOREIGN KEY (id_Obra)
       REFERENCES obraCinem(id_Obra)
);
-- Tabla de personajes
CREATE TABLE  personaje(
       personaje  VARCHAR(100),
       id_Personal NUMBER(10),
       id_Obra NUMBER(10),
       CONSTRAINT personaje_PK PRIMARY KEY (personaje, id_Personal, id_Obra),
       CONSTRAINT actuar_FK FOREIGN KEY (id_Personal, id_Obra)
                 REFERENCES actuarEn(id_Personal, id_Obra)
                           ON DELETE CASCADE
);

-- Tabla de relaciones entre películas
CREATE TABLE RelacionPeli(
       id_Obra_Sujeto NUMBER(10),
       id_Obra_Objeto NUMBER(10),
       tipoRel VARCHAR(10) NOT NULL,
       CONSTRAINT RelacionPeli_PK PRIMARY KEY (id_Obra_Sujeto,
```

```
        id_Obra_Objeto),
        CONSTRAINT peliSujeto_FK FOREIGN KEY (id_Obra_Sujeto)
              REFERENCES obraCinem(id_Obra),
        CONSTRAINT peliObjeto_FK FOREIGN KEY (id_Obra_Objeto)
              REFERENCES obraCinem(id_Obra),
        CONSTRAINT relacionPeli_NOREF CHECK (id_Obra_Sujeto <>
         id_Obra_Objeto),
        CONSTRAINT tpRelacion_VAL
        CHECK  (tipoRel IN ('remake','secuela','precuela'))
);

-- Tabla de capítulos
CREATE SEQUENCE secC START WITH 1 INCREMENT BY 1;
CREATE TABLE capitulo(
        id_Capitulo NUMBER(5),
        titulo VARCHAR(200) NOT NULL,
        id_Serie NUMBER(10) NOT NULL,
        numCap NUMBER(3),
        numTemp NUMBER(2),
        CONSTRAINT capitulo_PK PRIMARY KEY (id_Capitulo),
        CONSTRAINT serie_FK FOREIGN KEY (id_Serie)
              REFERENCES obraCinem(id_Obra)
);
```

El proceso de creación de tablas ha sido directo dado el modelo relacional. Esto se debe, en gran medida, a que el propio estudio de los datos proporcionados ha permitido llevar a cabo decisiones coherentes que, de otro modo, no se habrían llegado a considerar hasta bien llegados a cierto punto (como bien se ha mencionado en el apartado 1).

Por otra parte, se han podido implementar en la propia creación de tablas algunas de las restricciones del modelo, como aquellas relativas a dominios de datos, la restricción de especialización del campo *agno_Fin_Emision,* o la de reflexividad de las relaciones entre películas (una película no puede participar en una relación consigo misma). El resto se tratará en el apartado de creación de disparadores. Entre estos, mencionar que la unicidad del número de capítulo y temporada para una serie no se pudo implementar porque restricción *UNIQUE* no no se adapta bien a valores nulos.

Por último, se ha seguido un orden determinado en la creación de tablas, de modo que toda tabla se crea con posterioridad a las tablas que referencia por medio de claves ajenas. El mismo criterio se ha seguido en el borrado de las tablas.

# PARTE 2: Introducción de datos y ejecución de consultas

## 2.1 Población de la BD

La población ha sido la etapa que más tiempo ha ocupado en el presente proyecto, y por ello merece la pena extenderse en su explicación. El código con el que se ha poblado se adjunta en el Anexo 2. Población de la base debido a su extensión, pero se recomienda su lectura.

En primer lugar, **la propia información** que se representaba en la base de datos traía un problema, en tanto que ni el personal, ni las obras, ni los capítulos podían identificarse naturalmente, sino a través de claves artificiales. Además, en la propia colección de datos tampoco se seguía un formato para identificarlos. Por lo tanto, la mejor resolución del problema pasó a equivaler a la mejor manera de suponer cómo identificar al personal. En el caso de las obras, se supuso que cada combinación de (*título, año de estreno, tipo de película*) era única. Dicho de otro modo, las dos películas *Frozen* estrenadas en 2015 se consideran como una sola en esta base. Podría haberse intentado reparar esta brecha empleando datos adicionales de la tabla, como la lista de prioridades de los personajes (dos personajes con mismo orden de prioridad han de pertenecer a películas diferentes), pero ello generaría más diferenciaciones derivadas. En cuanto a capítulos, se decidió "identificar" un capítulo con su título, serie de televisión, número de capítulo y número de temporada.

El **personal**, en sí mismo, acarrea casi toda complicación en el proceso de población. Aunque en la base de datos tan solo se contenga información de su nombre, género y país de nacimiento, en el lote de datos se incluía además el lugar de nacimiento (más o menos detallado), el lugar de muerte, la fecha de nacimiento y la de muerte. En este caso, se decidió diferenciar al personal a partir de su nombre, género y lugar/fecha de nacimiento/muerte, y hacerlo dentro de sus participaciones en cada obra. El problema se presentaba cuando en una misma obra participaban dos personas con el mismo nombre y género, concretamente cuando había dos datos del mismo tipo asociados a un mismo nombre. Aquí se diferenció manualmente a partir del resto de información de la fila (por ejemplo, del personaje representado) y de otras participaciones en diferentes películas. Esta se consideró como la mejor manera de proceder, aunque no contemplase todos los casos, como en los que en una misma película hay solo una combinación de datos pero uno de ellos aparece más veces que el resto, lo que podría significar que hay una persona más con ese dato en concreto. En todo caso, el método empleado se consideró lo suficientemente riguroso como para no plantear otras posibilidades.

Una vez se estableció un método para diferenciar la distintas entidades, el proceso de población se realizó exclusivamente desde Oracle, a partir de la tabla *datosdb.datospeliculas* proporcionada. La población, bien hay que mencionarlo, consume bastante tiempo. Esto se debe a dos razones. Por una parte, a que se preservó la rigurosidad del método sin hacerla peligrar por una mejora de eficiencia. La segunda razón era el propio medio desde el que se pobló, pues posiblemente a partir de otro, como el poblado por ficheros *csv* junto con un tratamiento anterior de la información, la población habría ocupado menos tiempo. Pero, bien hay que decir, en esta etapa del proyecto no hubo muchas preocupaciones por esta circunstancia, pues lo que se consideró importante fue el análisis y tratado de la información.

Así, se creó una tabla provisional, *tabla_Prov*, necesaria para almacenar todos los datos de personal en la misma fila, ya que en la tabla proporcionada aparecen en filas separadas. También contenía el resto de información relevante con la que poblar la base de datos, con el fin de facilitar los *JOINS* a la hora de insertar en las tablas representando relaciones entre personal y obras, ya que ambas poseen claves artificiales. Para agilizar el proceso (de hecho, de no proceder así el tiempo se disparaba exponencialmente), también se incluyeron variables "booleanas" para representar si el personal contaba con datos contextuales o no, y para indicar si contaba con información contextual de cada tipo. Además, para los casos particulares (6 parejas) de personas con mismo nombre en una misma obra, se creó la vista materializada *bipersonas*. El proceso de población de tablas y vistas auxiliares fue el siguiente:

1. Se insertó al personal por obra sin sus datos contextuales (diferenciando entre personal sin datos y con a través de la variable booleana *personaVacía*).
2. Se actualizaron las filas donde *personaVacia* = 0. Para ello, se asignó 1 al booleano de cada dato contextual en las filas en las que existía información de ese tipo, se creó una vista materializada con la información de ese tipo, y finalmente se actualizó el campo dado en aquellas filas con la variable booleana a 1 desde la vista materializada creada.
3. Se insertó manualmente al personal de la vista materializada *bipersonas.*
4. Finalmente, se creó la vista materializada *pers_ext* con el personal y todos sus datos, asignándoles ya su clave artificial definitiva. Esto fue así porque sí se hubiese insertado directamente en la tabla *personal*, no habría manera de deducir el identificador de un miembro de personal en la tabla *tabla_Prov* al haber más de un personal con mismo nombre y nacionalidad.

Llegados a este punto, se estaba en disposición de poblar la base de datos. Dentro de esta fase, indicar que fue menester realizar algunas inserciones a través de varias fases allí donde se necesitaba un procesamiento adicional de la información, como para obtener el país de nacimiento partiendo del lugar de nacimiento, o para tratar el año de fin de emisión de una serie (que en el lote de datos aparece como '????' cuando aún no ha dejado de emitirse y en la base toma el valor NULL). Además, se crearon dos vistas materializadas auxiliares *fila_ext* y *link_ext,* donde se "precalculan" las operaciones JOIN de interés para la población de tablas relacionando a personal con obra y la de la tabla *relacionPeli,* respectivamente. Las condiciones de JOIN se basan en la decisión de identificación de entidades desarrollada al inicio del capítulo.

En conexión a esto, aunque no se considerasen los capítulos como "obras", sí que se empleó la información de personal participando en ellos. Esto es así porque muchas veces había personal en capítulos que no se asociaba directamente a la serie, o incluso había series sin ningún personal relacionado. Por ello, se determinó añadir al personal de capítulos en la serie a la que pertenecía en caso de que esta instancia no se hubiese introducido aún. Este fue un aspecto que debía incluirse en las operaciones de JOIN.

Para terminar, se debe incidir en que toda tabla o vista auxiliar debe ser eliminada al finalizar la población, y una vez se han poblado las tablas ya no se volverá a hacer uso del lote de datos proporcionado, pues entonces se caería en el error de degradar la base de datos a elemento complementario, siendo realmente este fundamental.

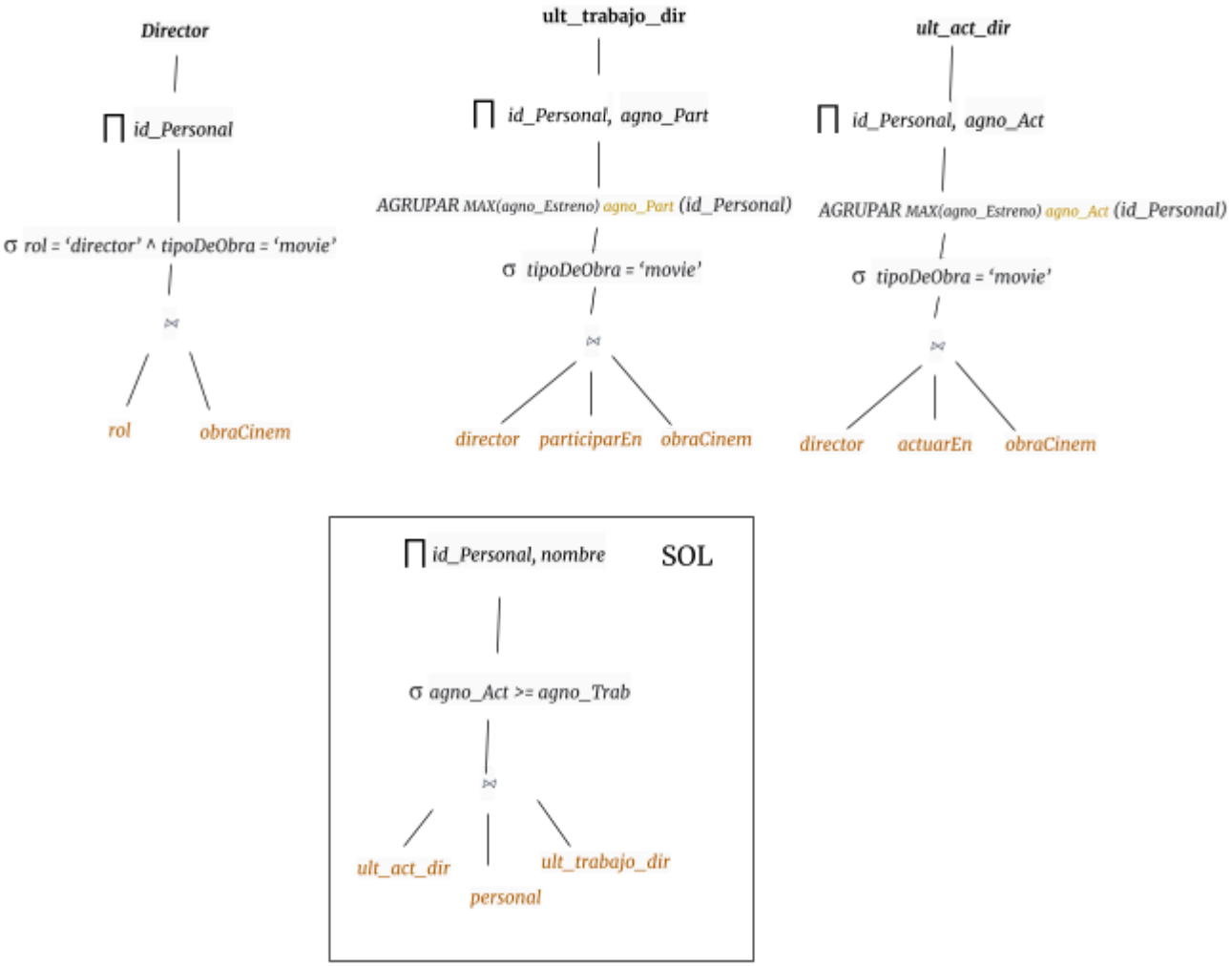
# 2.2 Consultas SQL

## 2.2.1 Consulta 1

**Directores para los cuales la última película en la que han participado ha sido como actor/actriz.**

La resolución de la primera consulta quedó condicionada por el modelo conceptual y lógico de la base, en tanto que las actuaciones del personal se encuentran en una tabla distinta al resto de participaciones. Así, el resultado se podía obtener de un modo sencillo comprobando que el último año con el que aparece un director en *actuarEn* sea mayor o igual al último año con el que aparece en *participarEn*. Es decir, buscando en qué tabla se encontraba la última participación del director.

A continuación se muestra el árbol sintáctico resultante:



Traduciendo el álgebra a lenguaje sql, se obtiene:

```
Unset
WITH
       -- personal que ha trabajado como director en películas
       director AS(
       SELECT DISTINCT id_Personal
       FROM rol  NATURAL JOIN obraCinem
       WHERE rol = 'director' AND tipoDeObra= 'movie'),
       -- Última participación de directores en películas (no como actores)
       ult_trabajo_dir AS(
       SELECT id_Personal, MAX(agno_Estreno) agno_Part
       FROM director NATURAL JOIN participarEn NATURAL JOIN obraCinem
       WHERE tipoDeObra = 'movie'
       GROUP BY id_Personal),
       -- última actuacion de directores como actores en películas
       ult_act_dir AS(
       SELECT id_Personal, MAX(agno_Estreno) agno_Act
       FROM director NATURAL JOIN actuarEn NATURAL JOIN obraCinem
       WHERE tipoDeObra = 'movie'
       GROUP BY id_Personal)
SELECT id_Personal "Directores cuya ultima participacion haya sido como
actor/actriz",
       nombre
FROM ult_act_dir NATURAL JOIN ult_trabajo_dir NATURAL JOIN personal
WHERE agno_Act >= agno_Part
ORDER BY nombre ASC;
```

La consulta devolvió **250** directores-actores (se muestran los primeros 20 resultados):

| idPersona | Nombre |
|---|---|
| 12409 | Abad, Diego |
| 4680 | Aguilar Vicente, Antonio |
| 29093 | Aguirre, FerNULLdo |
| 28861 | Albaladejo, Geli |
| 31973 | Alcalde, David |
| 26375 | Alcazar, Victor |
| 15115 | Alcobendas, Miguel |
| 6697 | Alessandrin, Patrick |
| 16218 | Amandoz, Koldo |
| 13905 | Almodovar, Pedro |
| 1816 | Amenabar, Alejandro |
| 26376 | Andueza, Jon |
| 29456 | Arevalo, Pilar |
| 13800 | Arias, Imanol |

| | |
|---|---|
| 11574 | Aristarain, Adolfo |
| 30426 | Armi?an, Jaime de |
| 34383 | Artacho, Raul |
| 1786 | Atanes, Carlos |
| …… | |
| 23414 | Yursky, Sergei |
| 18683 | Zhang, Yimou |

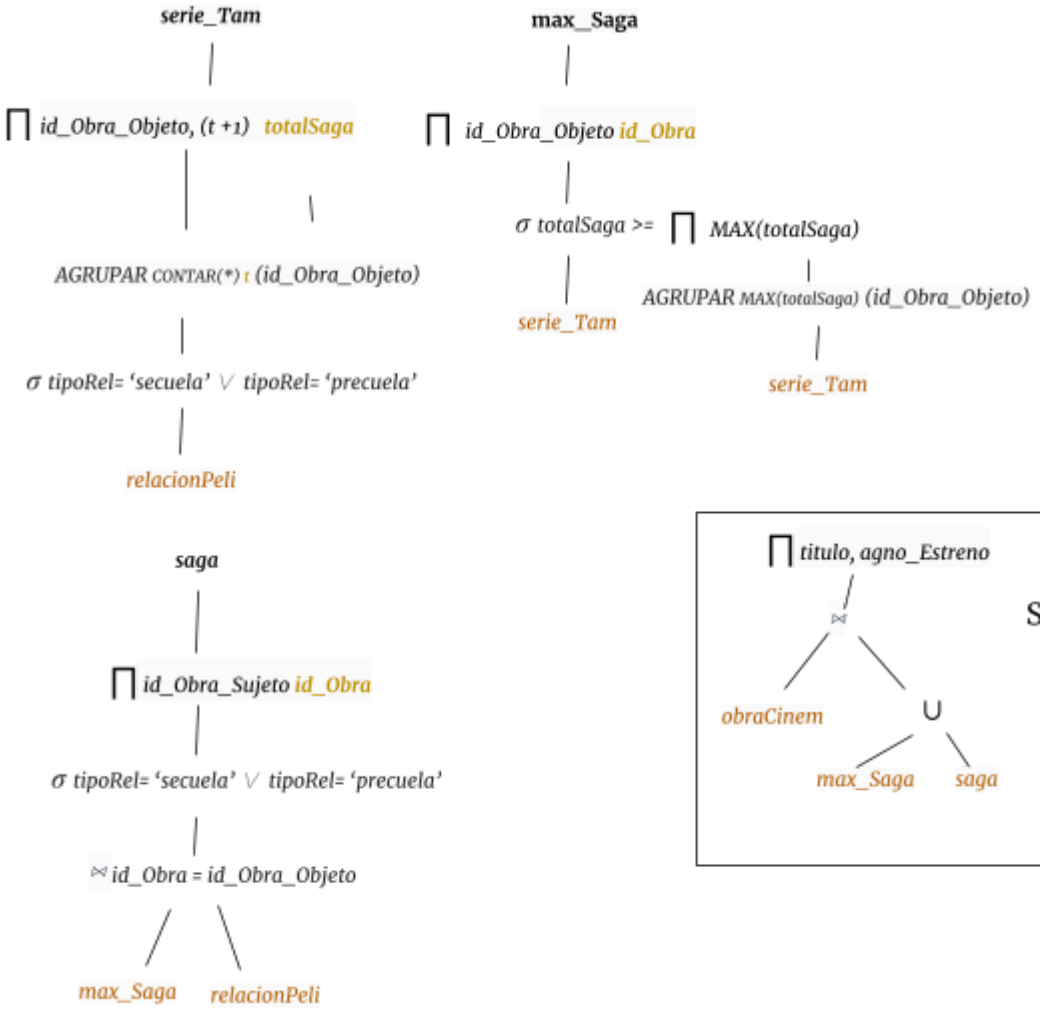
250 filas seleccionadas.

## 2.2.2 Consulta 2

**Obtener la saga de películas más larga (en número de películas), listando los títulos de las películas que la componen (incluyendo precuelas y secuelas).**

La clave a la hora de idear ha sido una premisa en el modelo, que aparece como anotación en el enunciado de la práctica. Y es que las relaciones entre películas poseen la propiedad de transitividad. Así, a partir de cualquier *obra_Objeto* se puede obtener la serie de precuelas/secuelas que le siguen. De este modo, con obtener el *id_Obra_Objeto* que mayor número de veces aparece relacionado por medio de precuelas o secuelas se tiene la película central de una saga, y a partir de ahí el resultado es directo. Nótese que no importa contar también la longitud de series de películas que no son la central, pues su longitud será de al menos una unidad menor que la central.

A continuación se muestra el árbol sintáctico resultante:



Traduciendo el álgebra a lenguaje sql, se obtiene:

```
Unset
WITH
       -- Tamaño de cada serie (saga o subsaga)
       serie_Tam AS(
       SELECT id_Obra_Objeto, (COUNT(*)+1) totalSaga
       FROM relacionPeli
       WHERE tipoRel = 'secuela' OR tipoRel = 'precuela'
       GROUP BY id_Obra_Objeto),
       -- Película central de la saga más larga
       max_Saga AS(
       SELECT id_Obra_Objeto id_Obra
       FROM serie_Tam
       WHERE totalSaga >= (SELECT MAX(totalSaga) FROM serie_Tam)),
       -- Resto de películas de la saga
       saga AS(
       SELECT id_Obra_Sujeto id_Obra
       FROM relacionPeli JOIN max_Saga ON (id_Obra = id_Obra_Objeto)
       WHERE tipoRel = 'secuela' OR tipoRel = 'precuela'),
       -- Saga completa
       saga_completa AS(
       SELECT * FROM max_Saga
       UNION
       SELECT * FROM saga)
SELECT titulo, agno_Estreno
FROM saga_completa NATURAL JOIN obraCinem
ORDER BY agno_Estreno ASC;
```

La respuesta obtenida por la base de datos ha sido:

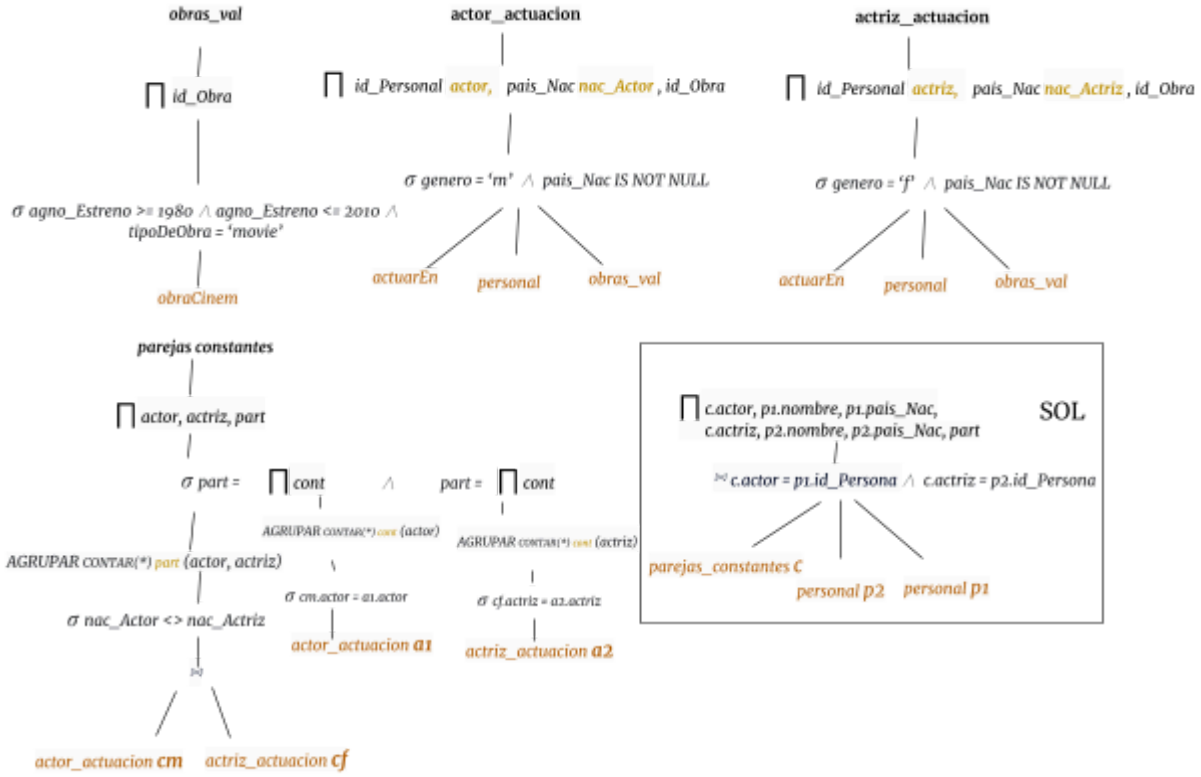
| Titulo | Agno_Estreno |
|---|---|
| La maldicion de la bestia | 1975 |
| El retorno del Hombre-Lobo | 1981 |
| La bestia y la espada magica | 1983 |
| El aullido del diablo | 1987 |
| Licantropo: El asesino de la luna llena | 1996 |

## 2.2.3 Consulta 3

**Listar las parejas actor-actriz de distinta nacionalidad que siempre han trabajado juntos en películas entre 1980 y 2010, indicando en cuantas ocasiones lo han hecho (listar en orden decreciente según este dato), y el nombre y nacionalidad de cada uno.**

La tercera consulta también se ha fundamentado en un hecho importante. Concretamente, en que si dos actores han trabajado las mismas veces juntos que cada uno por separado, con toda seguridad se puede afirmar que siempre han trabajado juntos. Afirmar lo contrario redunda en una contradicción, ya que claramente el conjunto de películas en las que han colaborado juntos es un subconjunto del conjunto de películas del actor y actriz por cada lado. Y si conjunto y subconjunto poseen la misma cardinalidad, entonces se puede afirmar que son iguales.

A continuación se muestra el árbol sintáctico resultante:




Traduciendo el álgebra a lenguaje sql, se obtiene:

```
Unset
WITH
	-- Obras dentro del rango estipulado
	obras_val AS(
	SELECT id_Obra
	FROM obraCinem
	WHERE agno_Estreno >= 1980 AND agno_Estreno <= 2010
		AND tipoDeObra='movie'),
	-- Actuaciones de actores
```

```
        actor_actuacion AS(
        SELECT id_Personal actor, pais_Nac nac_Actor, id_Obra
        FROM  actuarEn NATURAL JOIN personal NATURAL JOIN obras_val
        WHERE genero = 'm' AND pais_Nac IS NOT NULL),
        -- Actuaciones de actrices
        actriz_actuacion AS(
        SELECT id_Personal actriz, pais_Nac nac_Actriz, id_Obra
        FROM actuarEn NATURAL JOIN personal NATURAL JOIN obras_val
        WHERE genero = 'f' AND pais_Nac IS NOT NULL),
        -- parejas constantes entre actor-actriz
        parejas_constantes AS(
        SELECT actor, actriz, COUNT(*) part
        FROM actriz_actuacion cf NATURAL JOIN actor_actuacion cm
        WHERE nac_actor <> nac_actriz
        GROUP BY actor, actriz
              HAVING COUNT(*) = (SELECT COUNT(*) FROM actor_actuacion a1
                                 WHERE cm.actor = a1.actor)
              AND
               COUNT(*) = (SELECT COUNT(*) FROM actriz_actuacion a2
                           WHERE cf.actriz = a2.actriz
                           GROUP BY ACTriz))
SELECT c.actor, p1.nombre nombre_Actor, p1.pais_Nac nac_Actor, c.actriz,
       p2.nombre nActriz, p2.pais_Nac nac_Actriz,part "colaboraciones
       juntos"
FROM parejas_constantes c
     JOIN personal p1 ON(c.actor=p1.id_Personal)
     JOIN personal p2 ON(c.actriz =p2.id_Personal)
ORDER BY part DESC;
```

La consulta devolvió **1765** participaciones constantes:

| Actor | Nombre_Actor | Nac_Actor | Actriz | Nombre_Actriz | Nac_Actriz | Colaboraciones |
|---|---|---|---|---|---|---|
| 6738 | Jouvet, Maurice | France | 7045 | Beltran, Nelly | Argentina | 2 |
| 16353 | Mitchum, Christopher | USA | 4592 | Lahaie, Brigitte | France | 2 |
| 18871 | Guard, Dominic | UK | 2305 | Bruni Tedeschi,Valeria | Italy | 1 |
| 4636 | Poron, Jean-Francois | France | 2307 | Da Marchi, Laura | Italy | 1 |
| 9548 | Locke, Philip | UK | 2307 | Da Marchi, Laura | Italy | 1 |
| 16325 | Fantasia, Franco | Greece | 2307 | Da Marchi, Laura | Italy | 1 |
| 16565 | Carradine, Keith | USA | 2307 | Da Marchi, Laura | Italy | 1 |
| 19113 | Schumann, Erik | Germany | 2307 | Da Marchi, Laura | Italy | 1 |
| 4642 | Rooney, Mickey | USA | 2309 | Clewes, Lorraine | UK | 1 |
| 9099 | Hootkins, William | USA | 2309 | Clewes, Lorraine | UK | 1 |
| ..... | | | | | | |
| 23719 | Smirnov, Andrey | USSR | 32830 | Tenyakova, Natalya | Russia | 1 |
| 33032 | Thompson, Christopher | UK | 32922 | Blanco, Matilde | Spain | 1 |

1765 filas seleccionadas.

# PARTE 3: Diseño Físico

## 3.1 Mejoras de Rendimiento

Para medir la eficiencia de las consultas y detectar posibles problemas de rendimiento, se han analizado sus **planes de ejecución** a partir de las herramientas proporcionadas por Oracle. Así:

1. La primera consulta comporta un coste total de CPU de 169. Gran parte de este (68) se emplea en recorrer la tabla de roles para obtener a directores. Asimismo, el HASH GROUP BY correspondiente al proceso de obtención de las últimas actuaciones de directores tiene un coste de 52. En total, la consulta ha necesitado 32 operaciones.

2. La segunda ha tenido un coste total de CPU de 30. Gran parte de este se emplea en el JOIN correspondiente a juntar las tablas relacionPeli y maxSaga con un coste de 21. En total, se necesitan 22 operaciones.

3. La tercera ha tenido un coste total de CPU de 335. De este, 101 (dos veces) se empleaban en realizar un escaneo completo en la tabla de actuaciones y de personal y así obtener actuaciones de mujeres y hombres. Por otra parte, el emparejar actores y actrices comporta un coste de 106, sobre todo porque se debe evaluar la igualdad tanto para actores como para actrices. En total, la consulta necesita 38 operaciones.

Primeramente, fijándose en las consultas 1 y 3, donde se emplean muchas veces operaciones de JOIN entre el personal/obra y las tablas que representan sus relaciones, se estudió la posibilidad de **precalcular estos JOINS.** Pero era necesario meditar acerca de su conveniencia, ya que pese a los atributos comunes (que son identificadores) casi nunca se modifican, sí es cierto que en estas tablas se suelen añadir tuplas con frecuencia. Aprovechando que se dispone del año de estreno de las obras, se calculó la media anual de nuevo personal por año (considerando su inserción con su primera aparición en una obra), así como de obras, participaciones y actuaciones. Se obtuvo que había 1527 nuevos miembros de personal cada año, 220 nuevas películas, 2297 actuaciones y 1305 participaciones. Probablemente, las tablas de personal o actuaciones serán las que comporten un mayor movimiento en la base, pero dado que el año de estreno de las obras es una información de mucho interés y sus inserciones son más escasas, se estudió precalcular su valor en el JOIN entre obras y participaciones/actuaciones. Ahora bien, la medida no se pudo probar por falta de espacio.

```
Unset
CREATE BITMAP INDEX obra_act ON actuarEn(ObraCinem.agno_Estreno)
FROM obraCinem, actuarEn
WHERE obraCinem.id_Obra = actuarEn.id_Obra;

CREATE BITMAP INDEX obra_act ON participarEn(ObraCinem.agno_Estreno)
FROM obraCinem, actuarEn
WHERE obraCinem.id_Obra = participarEn.id_Obra;
```

Para mejorar la eficiencia de la primera consulta, y como también se prevé mayor importancia para el subconjunto de participaciones como directores, se propuso **particionar la tabla *rol*** en directores y el resto de papeles:

```
Unset
CREATE TABLE rol(
             ...
)
PARTITION BY LIST (rol) (PARTITION pr1_Director VALUES('director'),
       PARTITION pr2_OtrosRoles VALUES(DEFAULT));
```

El problema es que está partición, como cualquier otra, no se podía implementar por falta de espacio en el servidor. Alternativamente, se probó creando un índice, aunque no se considerase la opción más apropiada porque los atributos del campo *rol* pueden ser cadenas largas. De todos modos, no se acabó implementando porque no mejoraba la consulta 1.

Por otra parte, también se propuso crear una **vista materializada** con las últimas participaciones como actor de un director, por el coste que suponen y porque no ocupan mucho espacio necesariamente (en el momento de la consulta 456 filas). Con ella, se remodeló la consulta 1:

```
Unset
-- Vista materializada
CREATE MATERIALIZED VIEW ult_act_dir AS
WITH
       -- personal que ha trabajado como director
       director AS(
       SELECT DISTINCT id_Personal
       FROM rol  NATURAL JOIN obraCinem
       WHERE rol = 'director' AND tipoDeObra= 'movie')
SELECT id_Personal, MAX(agno_Estreno) agno_Act
FROM director NATURAL JOIN actuarEn NATURAL JOIN obraCinem
WHERE tipoDeObra = 'movie'
GROUP BY id_Personal;

-- Consulta 1
WITH
       -- Última participación de directores en obras (no como actores)
       ult_trab_dir AS(
       SELECT id_Personal, MAX(agno_Estreno) agno_Trab
       FROM ult_act_dir NATURAL JOIN participarEn NATURAL JOIN obraCinem
       WHERE tipoDeObra = 'movie'
       GROUP BY id_Personal)
SELECT id_Personal "Directores cuya ultima participacion haya sido como
actor/actriz",
```

```
      nombre
FROM ult_trab_dir NATURAL JOIN ult_act_dir NATURAL JOIN personal
WHERE agno_Act >= agno_Trab
ORDER BY nombre ASC;
```

Asimismo, se creó un índice sobre la clave ajena de personal en la tabla con participaciones de personal (que no actuaciones), ya que se realiza una agrupación sobre esta columna en la consulta 1. Sin embargo, no tuvo un impacto visible en la ejecución de la misma.

```
Unset
CREATE INDEX Personal_Trabajar_idx ON participarEn(id_Personal);
```

Para facilitar la filtración en la columna de con el tipo de relación en la tabla *RelacionPelis* se propuso crear un **índice** sobre dicha columna. Primeramente, se estudió implantar un índice BITMAP, ya que en la base de datos tan solo se trabajará con tres tipos de relaciones distintas. Pero finalmente se creó un índice normal porque, aunque su impacto en coste era idéntico, comportaba menos operaciones en la consulta 2. Además, también se creó un índice sobre la columna *id_Obra_Objeto* para facilitar las funciones de agrupación sobre la misma.

```
Unset
-- Índice sobre el tipo de relaciones entre películas
CREATE INDEX tipoRel_idx ON relacionPeli(tipoRel);
-- Índice sobre id de películas objeto de relación
CREATE INDEX idPeliObjeto_idx ON relacionPeli(id_Obra_Objeto);
```

Por otra parte, se creó una **vista materializada** con el tamaño de la serie de películas a partir de una dada. Es cierto que se podía haber propuesto materializar la consulta con la película central con la saga más larga, ya que esta se emplea posteriormente para sacar el resto de películas, pero realizarlo suponía materializar prácticamente toda la consulta, y no se sabe hasta qué punto era rentable:

```
Unset
CREATE MATERIALIZED VIEW serie_tam AS
       SELECT id_Obra_Objeto, (COUNT(*)+1) totalSaga
       FROM relacionPeli
       WHERE tipoRel = 'secuela' OR tipoRel = 'precuela'
       GROUP BY id_Obra_Objeto;
WITH
       -- Película central de la saga más larga
```

```
        max_Saga AS(
        SELECT id_Obra_Objeto id_Obra
        FROM serie_Tam
        WHERE totalSaga >= (SELECT MAX(totalSaga) FROM serie_Tam)),
        -- Resto de películas de la saga
        saga AS(
        SELECT id_Obra_Sujeto id_Obra
        FROM relacionPeli JOIN max_Saga ON (id_Obra = id_Obra_Objeto)
        WHERE tipoRel = 'secuela' OR tipoRel = 'precuela'),
        -- Saga completa
        saga_completa AS(
        SELECT * FROM max_Saga
        UNION
        SELECT * FROM saga)
SELECT titulo, agno_Estreno
FROM saga_completa NATURAL JOIN obraCinem
ORDER BY agno_Estreno ASC;
```

Además, como el género del personal solo toma tres valores (*'f', 'm'* y NULL), se propuso crear un **índice BITMAP** sobre dicha columna en *Personal (*también se planteó particionar la tabla personal respecto a la misma, pero no se pudo*)*. Esta decisión fue en teoría más que acertada, puesto que redujo el coste en CPU de la consulta 3 a 283:

```
Unset
CREATE BITMAP INDEX generoPersonal_idx ON personal(genero);
```

Otra decisión importante giró en torno al **número de actuaciones de cada personal.** Materializar las vistas con las actuaciones de personal en películas discriminando o no por el género se consideraba inasumible por el tamaño de estas (hubiese sido más razonable precalcular el JOIN entre la tabla de personal y de actuaciones). Asimismo, se consideró un campo adicional en la tabla de personal *numActuaciones,* con el número de actuaciones del personal en películas. Este se trata de un atributo derivado, por lo que exige mantener su consistencia mediante un trigger. Ahora bien, se trata de un atributo bastante específico (no considera actuaciones en series, ni trabajos con otros roles), y además su mantenimiento con triggers podía ser poco conveniente al dispararse con cada nueva actuación, por lo que se observó que no era una opción tan atractiva como se podía esperar. Otro planteamiento era el de crear una vista materializada con el número de actuaciones por personal, pero incluiría 23055 filas, lo que supone el 10,87% de la base de datos, y además resultó empeorar el rendimiento de la consulta:

```
Unset
-- Numero de actuaciones en películas de actores
CREATE MATERIALIZED VIEW num_Actuaciones_M AS
WITH
       -- Obras dentro del rango estipulado
       obras_val AS(
       SELECT id_Obra
       FROM obraCinem
       WHERE agno_Estreno >= 1980 AND agno_Estreno <= 2010
              AND tipoDeObra='movie')
SELECT id_Personal actor, COUNT(DISTINCT id_Obra) part
FROM actuarEn NATURAL JOIN personal NATURAL JOIN obras_val
WHERE genero = 'm' AND pais_Nac IS NOT NULL
GROUP BY id_Personal;

-- Número de actuaciones en películas de actrices
CREATE MATERIALIZED VIEW num_Actuaciones_F AS
WITH
       -- Obras dentro del rango estipulado
       obras_val AS(
       SELECT id_Obra
       FROM obraCinem
       WHERE agno_Estreno >= 1980 AND agno_Estreno <= 2010
              AND tipoDeObra='movie')
SELECT id_Personal actriz, COUNT(DISTINCT id_Obra) part
FROM actuarEn NATURAL JOIN personal NATURAL JOIN obras_val
WHERE genero = 'f' AND pais_Nac IS NOT NULL
GROUP BY id_Personal;
```

En contrapartida, se propuso crear una **vista materializada** con las obras válidas para la consulta, que son 2884:

```
Unset
CREATE MATERIALIZED VIEW obras_val AS
       SELECT id_Obra
       FROM obraCinem
       WHERE agno_Estreno >= 1980 AND agno_Estreno <= 2010
              AND tipoDeObra='movie';
```

**En conclusión**, las mejoras introducidas finalmente consiguieron reducir el coste de las consultas, aunque en la tercera hubo menor margen de maniobra para no perjudicar el resto de operaciones en la base de datos. Así, la creación de una vista materializada en la consulta 1 redujo el coste en CPU de 169 a 61 y el número de operaciones de 32 a 15. En la consulta 2, la creación de índices redujo el coste en 3 unidades, y la vista materializada posterior a 21. Por último, en la consulta 3 el índice BITMAP rebajó el coste de 335 a 283, y la creación posterior de la vista materializada a 259; pero se necesitó el mismo número de operaciones.

## 3.2 Triggers

Existen una serie de restricciones que se han podido verificar con la estructura de tablas definidas (como la exclusividad de especialización a través del atributo *tipoDeObra*), o con restricciones de integridad (como que una película no puede participar en una relación consigo misma). Pero hay otras que no. Entre ellas, se encuentran:

1. En *tener_Relacion, id_Obra_Objeto* y *id_Obra_Sujeto* deben hacer referencia a una obra de tipo película.
2. En *Capítulo, id_Obra* referencia a una obra de tipo serie.
3. Una serie no puede constar de un capítulo con el mismo número de capítulo y temporada.
4. En *RelacionPeli,* la obra *sujeto* debe haberse estrenado con posterioridad a la obra *objeto.*
5. Toda ocurrencia (*id_Personal, id_Obra*) en *ParticiparEn* debe contar con al menos una ocurrencia en la tabla *rol.*
6. Sean *x, y, z* películas. Si *z* es secuela de *y*, e *y* secuela de *x,* entonces *z* es secuela de *x.* Análogamente con precuelas.

De estas, la quinta se decidió no resolver por su complejidad, puesto que en los procesos de población su trigger no podría estar activo directamente al necesitar introducir una participación en obra antes de especificar el rol con el que se participa. La sexta restricción, más que una restricción, se convirtió en un aspecto a tratar en el proceso de población para que el modelo conceptual tuviese sentido. Así, se trató de diseñar un trigger que extendiese iterativamente relaciones por transitividad hasta no poder más, pero esta idea se abandonó porque las técnicas necesarias excedían los contenidos del curso (además de que el lote de datos proporcionado ya cumplía esta propiedad, por lo que la consulta 2 no se veía perjudicada).

## TRIGGER 1: Unicidad del número de capítulo dentro de una temporada

El primer trigger preserva la restricción de que ninguna serie puede tener capítulos con el mismo número de capítulo y mismo número de temporada:

```
Unset
CREATE OR REPLACE TRIGGER capitulos_Orden
AFTER INSERT OR UPDATE ON capitulo
DECLARE
       caps_Repetidos NUMBER;
BEGIN
       -- Contar el número de pareja de capítulos distintos de una serie
       -- con mismo número de capítulo y de temporada
       SELECT COUNT(*) INTO caps_Repetidos
       FROM capitulo c1
       JOIN capitulo c2 ON (c1.id_Serie=c2.id_Serie AND c1.id_Capitulo >
                           c2.id_Capitulo)
       WHERE c1.numCap = c2.numCap AND c1.numTemp = c2.numTemp;
       -- Si existe más de una pareja mostrar error
       IF caps_Repetidos > 0 THEN
       raise_application_error(-20200, 'ERROR. Hay ' || caps_Repetidos ||
                                  'parejas de capitulos con mismo numero de
                                   capitulo y de temporada.');
       END IF;
END;
```

El trigger se dispara después de una sentencia para evitar así problemas de mutación, y se especifica para procesos de inserción, pero también de actualización.

## TRIGGER 2: Consistencia de la relación de especialización *RelacionPeli*

El segundo trigger preserva la consistencia en la tabla *RelacionPeli* de modo que todas las parejas de obras que se introduzcan sean películas:

```
Unset
CREATE OR REPLACE TRIGGER relPeli
BEFORE INSERT OR UPDATE ON relacionPeli
FOR EACH ROW
DECLARE
	-- Variable para almacenar el número de obras (0-2) que no son
películas
	ObraSerie NUMBER(1);
BEGIN
	-- Contar número de obras de la relación que son películas
	SELECT COUNT(*) INTO ObraSerie
	FROM obraCinem
	WHERE (id_Obra = :NEW.id_Obra_Sujeto
		OR id_Obra = :NEW.id_Obra_Objeto)
		AND tipoDeObra = 'movie';
	-- Si alguna (o ambas) no son películas mostrar error
	IF ObraSerie < 2 THEN raise_application_error(-20200, 'La relacion
	('||:NEW.id_Obra_Sujeto || ') y ('||:NEW.id_Obra_Objeto||' )
	no esta definida sobre peliculas.');
	END IF;
END;
/
```

El trigger se dispara a nivel de fila, pues se tiene que consultar la tabla en la que se muestra el tipo de obra que es, y se realizar antes de la inserción y actualización, pues es un caso de error que tiene que ver con los datos a introducir y no con sus efectos en la tabla.

## TRIGGER 3: Orden de relación temporal entre películas que se relacionan

El tercer trigger preserva la restricción de que en la tabla *RelacionPeli* todas las películas que se vayan a insertar como obra_Sujeto siempre tengan una fecha de lanzamiento posterior a obra_Objeto:

```
Unset
CREATE OR REPLACE TRIGGER agno_Estreno_Rel_Pelis
BEFORE INSERT OR UPDATE ON relacionPeli
FOR EACH ROW
DECLARE
       -- Variable para almacenar el agno de estreno de la película sujeto
       agno_Peli_Sujeto NUMBER;
       -- Variable para almacenar el agno de estreno de la película objeto
       agno_Peli_Objeto NUMBER;
BEGIN
       -- Seleccionar agno de estreno de la película sujeto
       SELECT DISTINCT agno_Estreno INTO agno_Peli_Sujeto
       FROM obraCinem
       WHERE id_Obra = :NEW.id_Obra_Sujeto;
       -- Seleccionar agno de estreno de la película objeto
       SELECT DISTINCT agno_Estreno INTO agno_Peli_Objeto
       FROM obraCinem
       WHERE id_Obra = :NEW.id_Obra_Objeto;
       -- Mostrar error si la película sujeto se estreno con anterioridad
       IF agno_Peli_Sujeto < agno_Peli_Objeto THEN
       raise_application_error(-20210, 'La obra identificada por ' ||
       :NEW.id_Obra_Sujeto || ' no puede haberse estrenado con anterioridad
        a la obra relacionada identificada por ' || :NEW.id_Obra_Objeto);
       END IF;
END;
/
```

El trigger se dispara a nivel de fila, pues la consulta se realiza sobre la tabla de obras. Y por la misma razón que el anterior trigger,  se ejecuta antes de insertar o actualizar.

# Anexo 1. Evaluación individual

A continuación se muestra un desglose de las horas que ha empleado cada integrante del equipo en cada apartado:

| | **ATHANASIOS USERO** | **CURRO VALERO** | **DANIEL RAMÓN** | **Total tarea** |
|---|---|---|---|---|
| Trabajo de clase | 6h | 6h | 6h | 18h |
| Modelo E/R | 8 h | 4h | 2,5h | 14,5h |
| Esquema Relacional | 3 h | 2,5h | 1h | 6,5h |
| Creación Tablas | 2 h | 3h | 1,5h | 6,5h |
| Población | 14h | 5h | 1,5h | 20,5h |
| Consultas SQL | 4h | 5h | 3h | 12h |
| Rendimiento | 5h | 1h | 1h | 7h |
| Triggers | 6h | 4h | 2,5h | 12,5h |
| Total Persona | 48h | 30,5h | 19h | 97,5h |

# Anexo 2. Población de la base

```
Unset
/* Fichero con la población de tablas */
-- CREACIÓN DE TABLAS AUXILIARES
DROP SEQUENCE secT;
CREATE SEQUENCE secT START WITH 1 INCREMENT BY 1;
       CREATE TABLE  tabla_Prov(
       id_Personal NUMBER(10),
       nombre VARCHAR(70),
       genero VARCHAR(1),
       persona_Vac NUMBER(1),
       lugar_Nac  VARCHAR(200),
       bln          NUMBER(1),
       lugar_M VARCHAR(200),
       blm          NUMBER(1),
       fecha_Nac VARCHAR(60),
       bfn          NUMBER(1),
       fecha_M VARCHAR(60),
       bfm          NUMBER(1),
       rol_Personal VARCHAR(30),
       personaje    VARCHAR(100),
       titulo       VARCHAR(200),
       agno_Estreno NUMBER(4),
       tipoDeObra   VARCHAR(20),
       agno_fin_Emision VARCHAR(4),
       titulo_Serie   VARCHAR(200),
       agno_Estreno_Serie NUMBER(4),
       CONSTRAINT tabla_PK PRIMARY KEY(id_Personal)
);
-- Vista materializada con el nombre de personal referenciando a más de una
persona en una obra y la obra
       CREATE MATERIALIZED VIEW bipersonas AS
 SELECT DISTINCT d1.title, d1.production_year, d1.kind, d1.name
 FROM datosdb.datospeliculas d1
       JOIN datosdb.datospeliculas d2 ON(d1.title = d2.title AND
d1.production_year=d2.production_year
                                        AND d1.name=d2.name AND d1.kind =
d2.kind AND d1.gender = d2.gender)
 WHERE d1.info_context=d2.info_context AND  d1.person_info<>d2.person_info;


-- Insertar en la tabla participaciones de personal en obras donde aparece
sin datos
       INSERT INTO tabla_Prov
```

Unset

```
  WITH
       pers AS(SELECT DISTINCT name, gender, role, role_name, title,
production_year, kind, series_years, serie_title, serie_prod_year
             FROM datosdb.datospeliculas
             WHERE info_context IS NULL AND name IS NOT NULL)
  SELECT secT.NEXTVAL, name, gender, 1,NULL,0 , NULL, 0, NULL, 0, NULL, 0,
role, role_name,
       title, production_year, kind, SUBSTR(series_years,6,4), serie_title,
serie_prod_year
  FROM pers;

-- Insertar en la tabla participiaciones de personal en obras y sus datos
donde aparece con datos
       INSERT INTO tabla_Prov
  WITH
       pers AS(SELECT DISTINCT d.name, d.gender,d.role, role_name,
v d.title, d.production_year, kind, series_years, serie_title,
serie_prod_year
FROM datosdb.datospeliculas d WHERE d.info_context IS NOT NULL AND d.name IS
NOT NULL AND NOT EXISTS (SELECT * FROM bipersonas b  WHERE b.name=d.name AND
b.title=d.title AND b.production_year=d.production_year AND b.kind=d.kind))
SELECT secT.NEXTVAL, name, gender, 0, NULL,0, NULL,0, NULL,0,  NULL,0, role,
role_name, title, production_year, kind, SUBSTR(series_years,6,4),
serie_title, serie_prod_year
  FROM pers;

-- ACTUALIZAR EL LUGAR DE NACIMIENTO

--  Marcar las filas donde existan datos del lugar de nacimiento
       UPDATE tabla_Prov t
SET bln = 1
WHERE persona_Vac = 0 AND EXISTS(SELECT * FROM datosdb.datospeliculas
             WHERE t.nombre=name AND t.titulo=title
                    AND t.agno_Estreno=production_year
                    AND t.tipoDeObra=kind
                    AND info_context='birth notes');
```

Unset

```
-- Almacenar información de esas filas
CREATE MATERIALIZED VIEW ln AS
SELECT DISTINCT name, title, production_year, kind, person_info
FROM datosdb.datospeliculas
WHERE info_context='birth notes';
```

```
-- Actualizar el lugar de nacimiento en aquellas filas marcadas
       UPDATE tabla_Prov t
  SET lugar_Nac = (SELECT DISTINCT person_info FROM ln d WHERE
t.nombre=d.name AND t.titulo=d.title AND t.agno_Estreno=d.production_year
AND t.tipoDeObra=d.kind)
  WHERE persona_Vac = 0 AND bln = 1 AND NOT EXISTS (SELECT * FROM bipersonas
b WHERE b.name=nombre AND b.title=titulo AND b.production_year=agno_Estreno
AND b.kind=tipoDeObra);

DROP MATERIALIZED VIEW ln;

-- ACTUALIZAR LUGAR DE MUERTE

--  Marcar las filas donde existan datos del lugar de defunción
       UPDATE tabla_Prov t
SET blm = 1
WHERE persona_Vac = 0 AND  EXISTS( SELECT * FROM datosdb.datospeliculas
                                   WHERE t.nombre=name AND t.titulo=title
                                         AND t.agno_Estreno=production_year
                                         AND t.tipoDeObra=kind
                                         AND info_context='death notes');

-- Almacenar información de esas filas
CREATE MATERIALIZED VIEW lm AS
SELECT DISTINCT name, title, production_year, kind, person_info
FROM datosdb.datospeliculas
WHERE info_context='death notes';

-- Actualizar el lugar de defunción en aquellas filas marcadas
       UPDATE tabla_Prov t
  SET lugar_M = (SELECT DISTINCT person_info
                     FROM lm d
                     WHERE t.nombre=d.name AND t.titulo=d.title
                           AND t.agno_Estreno=d.production_year
                           AND t.tipoDeObra=d.kind)
  WHERE persona_Vac = 0 AND blm = 1 AND NOT EXISTS (SELECT * FROM bipersonas
b WHERE b.name=nombre AND b.title=titulo AND b.production_year=agno_Estreno
AND b.kind=tipoDeObra);

DROP MATERIALIZED VIEW lm;
```

Unset

```
-- ACTUALIZAR FECHA DE NACIMIENTO
```

```
--  Marcar las filas donde existan datos de la fecha de nacimiento
      UPDATE tabla_Prov t
SET bfn = 1
WHERE persona_Vac = 0 AND EXISTS( SELECT * FROM datosdb.datospeliculas
                                  WHERE t.nombre=name AND t.titulo=title
                                        AND t.agno_Estreno=production_year
                                        AND t.tipoDeObra=kind
                                        AND info_context='birth date');

-- Almacenar información de esas filas
CREATE MATERIALIZED VIEW fn AS
SELECT DISTINCT name, title, production_year, kind, person_info
FROM datosdb.datospeliculas
WHERE info_context='birth date';

-- Actualizar la fecha de nacimiento en aquellas filas marcadas
      UPDATE tabla_Prov t
  SET fecha_Nac = (SELECT DISTINCT person_info
                   FROM fn d
                   WHERE t.nombre=d.name AND t.titulo=d.title
                         AND t.agno_Estreno=d.production_year
                         AND t.tipoDeObra=d.kind)
  WHERE persona_Vac = 0 AND bfn = 1 AND NOT EXISTS (SELECT * FROM bipersonas
b            WHERE b.name=nombre AND b.title=titulo AND
b.production_year=agno_Estreno AND b.kind=tipoDeObra);

DROP MATERIALIZED VIEW fn;

-- ACTUALIZAR FECHA DE MUERTE

--  Marcar las filas donde existan datos de la fecha de defunción
      UPDATE tabla_Prov t
SET bfm = 1
WHERE persona_Vac = 0 AND EXISTS( SELECT * FROM datosdb.datospeliculas
                                  WHERE t.nombre=name AND t.titulo=title
                                        AND t.agno_Estreno=production_year
                                        AND t.tipoDeObra=kind
                                        AND info_context='death date');
```

Unset

```
-- Almacenar información de esas filas
CREATE MATERIALIZED VIEW fm AS
SELECT DISTINCT name, title, production_year, kind, person_info
FROM datosdb.datospeliculas
WHERE info_context='death date';

-- Actualizar la fecha de defunción en aquellas filas marcadas
       UPDATE tabla_Prov t
  SET fecha_M = (SELECT DISTINCT person_info
                    FROM fm d
                    WHERE t.nombre=d.name AND t.titulo=d.title
                          AND t.agno_Estreno=d.production_year
                          AND t.tipoDeObra=d.kind)
  WHERE persona_Vac = 0 AND bfm = 1 AND NOT EXISTS (SELECT * FROM bipersonas
b WHERE b.name=nombre AND b.title=titulo AND
b.production_year=agno_Estreno AND b.kind=tipoDeObra);
DROP MATERIALIZED VIEW fm;

-- Añadir manualmente participaciones de personal que aparecen junto con
otra de mismo nombre en una misa obra
       INSERT INTO tabla_Prov VALUES (secT.NEXTVAL,'Gonzalez Sinde, Jose
Maria', 'm', 0, 'Madrid, Spain', 1, NULL, 0, '18 April 1968', 1, NULL, 0,
'actor', 'Pepito', 'Solos en la madrugada', '1978', 'movie', NULL, NULL,
NULL);
       INSERT INTO tabla_Prov VALUES (secT.NEXTVAL,'Gonzalez Sinde, Jose
Maria', 'm', 0, 'Burgos, Castilla y Leon, Spain', 1,'Madrid, Spain', 1,
'1941', 1, '20 December 1992', 1, 'writer', NULL, 'Solos en la madrugada',
'1978', 'movie', NULL, NULL, NULL);
       INSERT INTO tabla_Prov VALUES (secT.NEXTVAL,'Gonzalez Sinde, Jose
Maria', 'm', 0, 'Burgos, Castilla y Leon, Spain', 1,
                                  'Madrid, Spain', 1, '1941', 1, '20 December
1992', 1, 'producer', NULL, 'Solos en la madrugada', '1978', 'movie', NULL,
NULL, NULL);
       INSERT INTO tabla_Prov VALUES (secT.NEXTVAL,'Gonzalez Sinde, Jose
Maria', 'm', 0, 'Madrid, Spain', 1, NULL, 0, '18 April 1968', 1, NULL, 0,
'actor', 'Child', 'Asignatura pendiente', '1977', 'movie', NULL, NULL,
NULL);
```

Unset

```
	INSERT INTO tabla_Prov VALUES (secT.NEXTVAL,'Gonzalez Sinde, Jose
Maria', 'm', 0, 'Burgos, Castilla y Leon, Spain', 1,
                                'Madrid, Spain', 1, '1941', 1, '20 December
1992', 1, 'writer', NULL,'Asignatura pendiente', '1977', 'movie', NULL,
NULL, NULL);
	INSERT INTO tabla_Prov VALUES (secT.NEXTVAL,'Gonzalez Sinde, Jose
Maria', 'm', 0, 'Burgos, Castilla y Leon, Spain', 1,'Madrid, Spain', 1,
'1941', 1, '20 December 1992', 1, 'producer', NULL,'Asignatura pendiente',
'1977', 'movie', NULL, NULL, NULL);
	INSERT INTO tabla_Prov VALUES (secT.NEXTVAL,'Ponte, Laura', 'f', 0,
'Buenos Aires, Argentina', 1, NULL, 0, '1958', 1, NULL, 0,
'actress','Paula', 'La rosa de piedra', '1999', 'movie', NULL, NULL, NULL);
	INSERT INTO tabla_Prov VALUES (secT.NEXTVAL,'Ponte, Laura', 'f', 0,
'Vigo, Pontevedra, Spain', 1, NULL, 0, '9 June 1973',
                                1, NULL, 0, 'actress','Paula', 'La rosa de
piedra', '1999', 'movie', NULL, NULL, NULL);
	INSERT INTO tabla_Prov VALUES (secT.NEXTVAL,'Ulloa, Alejandro', 'm',
0, NULL, 0, 'Barcelona, Spain (complications from a fall)', 1,'22 October
1916', 1, '27 April 2004', 0, 'actor', NULL, 'Romanza final (Gayarre)',
'1986','movie', NULL, NULL, NULL);
	INSERT INTO tabla_Prov VALUES(secT.NEXTVAL,'Ulloa, Alejandro', 'm', 0,
'Madrid, Spain', 1, 'Madrid, Spain', 1, '14 September 1926', 1,
'14/05/2002', 1, 'cinematographer', NULL, 'Romanza final (Gayarre)', '1986',
'movie', NULL, NULL, NULL);

	INSERT INTO tabla_Prov VALUES(secT.NEXTVAL,'Nieto, Jose', 'm', 0,
'Madrid, Spain', 1, NULL, 0, '1 March 1942', 1, NULL, 0, 'composer',NULL,
'Hay que matar a B.', '1975', 'movie', NULL, NULL, NULL);

	INSERT INTO tabla_Prov VALUES(secT.NEXTVAL,'Nieto, Jose', 'm', 0,
'Murcia, Murcia, Spain', 1, 'Matalascanas, Huelva, Andalucia', 1,
'03/05/2003', 1, '9 August 1982', 1, 'actor',NULL, 'Hay que matar a B.',
'1975', 'movie', NULL, NULL, NULL);

	INSERT INTO tabla_Prov VALUES(secT.NEXTVAL,'Closas, Alberto', 'm', 0,
'Barcelona, Cataluna, S', 1, NULL, 0, '1961', 1, NULL, 0, 'actor', NULL, 'La
familia, bien, gracias','1979', 'movie', NULL, NULL, NULL);
```

Unset

```
	INSERT INTO tabla_Prov VALUES(secT.NEXTVAL,'Closas, Alberto', 'm', 0,
'Barcelona, Barcelona, Catalonia, Spain', 1,'Madrid, Madrid, Spain (lung
cancer)', 1, '3 October 1921', 1, '19 September 1994', 1, 'actor','Carlos',
'La familia, bien, gracias', '1979', 'movie', NULL, NULL, NULL);
```

```
-- Crear vista materializada con la informacion detallada del personal y sus
identificadores artificiales
       CREATE SEQUENCE secP START WITH 1 INCREMENT BY 1;

       CREATE MATERIALIZED VIEW pers_Ext AS
       WITH
       pers AS(
              SELECT DISTINCT nombre, genero, lugar_Nac, lugar_M, fecha_Nac,
fecha_M
              FROM tabla_Prov)
              SELECT secP.NEXTVAL id_Personal, nombre, genero, lugar_Nac,
lugar_M, fecha_Nac, fecha_M
              FROM pers;
-- POBLACIÓN DE LAS TABLAS EN LA BASE DE DATOS
-- Poblacion del personal
       INSERT INTO personal
         WITH
       -- Personal con país de nacimiento
              pers_con_pais AS(
              SELECT DISTINCT id_Personal, nombre, genero,
REGEXP_SUBSTR(REGEXP_SUBSTR(lugar_Nac, '[^,](asterisco)$'), '[A-Za-z][A-Za-z
](*)[A-Za-z]') pais_Nac
              FROM pers_Ext WHERE lugar_Nac IS NOT NULL),
       -- Personal sin país de nacimiento
              pers_sin_pais AS(
              SELECT DISTINCT id_Personal, nombre, genero, lugar_Nac pais_Nac
              FROM pers_Ext WHERE lugar_Nac IS NULL)
         SELECT id_Personal, nombre, genero, pais_Nac
         FROM (SELECT * FROM pers_con_pais) UNION (SELECT * FROM
pers_sin_pais);

-- Poblacion de obras cinematográficas
       CREATE SEQUENCE secO START WITH 1 INCREMENT BY 1;
-- Población de películas o series que no han terminado de emitirse
INSERT INTO obraCinem
         WITH
       obra AS(
              SELECT DISTINCT title, production_year, kind
              FROM datosdb.datospeliculas
              WHERE (kind = 'movie' OR (kind = 'tv series' AND
SUBSTR(series_years,6,4)='????')) AND title IS NOT NULL AND production_year
IS NOT NULL)
         SELECT secO.NEXTVAL, title, production_year, kind, NULL
         FROM obra;
```

Unset

```
-- Población de series que han terminado de emitirse
INSERT INTO obraCinem
  WITH
       obra AS(
       SELECT DISTINCT title, production_year, kind, series_years
       FROM datosdb.datospeliculas
       WHERE (kind = 'tv series' AND SUBSTR(series_years,6,4)<>'????' AND
series_years IS NOT NULL) AND title IS NOT NULL AND production_year IS NOT
NULL)
  SELECT secO.NEXTVAL, title, production_year, kind,
TO_NUMBER(SUBSTR(series_years,6,4))
  FROM obra;
-- Población de género de obras cinematográficas
INSERT INTO genero
  WITH
obra_ext AS
       (SELECT * FROM datosdb.datospeliculas JOIN obraCinem ON(title = titulo
AND agno_Estreno = production_year AND kind = tipoDeObra AND
((((agno_fin_Emision = SUBSTR(series_years,6,4) AND agno_fin_Emision IS NOT
NULL) OR (agno_fin_Emision IS NULL AND SUBSTR(series_years,6,4) = '????'))
AND kind = 'tv series') OR (kind = 'movie'))))
  SELECT DISTINCT keyword, id_Obra FROM obra_ext WHERE keyword IS NOT NULL;

       -- Vista materializada para asociar los identificadores artificiales
con las obras y personal de la tabla original
CREATE MATERIALIZED VIEW fila_ext AS
  SELECT DISTINCT p.id_Personal, o.id_Obra, t.rol_Personal, t.personaje
  FROM tabla_Prov t JOIN obraCinem o ON(t.titulo=o.titulo AND o.agno_Estreno
= t.agno_Estreno AND t.tipoDeObra = o.tipoDeObra AND ((((o.agno_fin_Emision
= t.agno_fin_Emision AND o.agno_fin_Emision IS NOT NULL) OR
(o.agno_fin_Emision IS NULL AND t.agno_fin_Emision = '????')) AND
t.tipoDeObra = 'tv series') OR (t.tipoDeObra='movie')) OR (t.titulo_Serie =
o.titulo AND o.agno_Estreno = t.agno_Estreno_Serie AND t.tipoDeObra =
'episode' AND o.tipoDeObra = 'tv series')) JOIN pers_Ext p ON (p.nombre =
t.nombre AND (p.genero = t.genero OR (t.genero IS NULL AND p.genero IS
NULL)) AND ((p.lugar_Nac = t.lugar_Nac) OR (p.lugar_Nac IS NULL AND
t.lugar_Nac IS NULL)) AND ((p.lugar_M = t.lugar_M) OR (p.lugar_M IS NULL AND
t.lugar_M IS NULL)) AND ((p.fecha_Nac = t.fecha_Nac) OR (p.fecha_Nac IS NULL
AND t.fecha_Nac IS NULL)) AND ((p.fecha_M = t.fecha_M) OR (p.fecha_M IS NULL
AND t.fecha_M IS NULL)));
```

```
Unset
-- Insertar participaciones en películas (que no actuaciones)

       INSERT INTO participarEn
       SELECT DISTINCT id_Personal, id_Obra
       FROM fila_ext
       WHERE rol_Personal<>'actor' AND rol_Personal<>'actress';

-- Insertar actuaciones en películas
       INSERT INTO actuarEn
       SELECT DISTINCT id_Personal, id_Obra
       FROM fila_ext
       WHERE rol_Personal='actor' OR rol_Personal='actress';

-- Población de roles de participaciones
INSERT INTO rol
  SELECT rol_Personal, id_Personal, id_Obra
  FROM fila_ext
  WHERE rol_Personal<>'actor' AND rol_Personal<>'actress';

-- Población de personajes en películas
INSERT INTO personaje
       SELECT DISTINCT personaje, id_Personal, id_obra
       FROM fila_ext
       WHERE personaje IS NOT NULL AND (rol_Personal = 'actor' OR
rol_Personal = 'actress');

-- Vista materializada para asociar los identificadores artificiales con las
obras relacionadas de la tabla original
CREATE MATERIALIZED VIEW link_ext AS
  SELECT DISTINCT obSuj.id_Obra id_ObraSuj, obObj.id_Obra id_ObraObj, link
  FROM datosdb.datospeliculas db
       JOIN obraCinem obSuj ON(db.title = obSuj.titulo AND
obSuj.agno_Estreno = db.production_year AND db.kind = obSuj.tipoDeObra AND
db.kind = 'movie')
       JOIN obraCinem obObj ON(db.titlelink = obObj.titulo AND
obObj.agno_Estreno = db.productionyearlink AND obObj.tipoDeObra = 'movie');

-- Población de ralaciones entre películas
       INSERT INTO RelacionPeli
  SELECT id_ObraSuj, id_ObraObj, 'remake' FROM link_ext WHERE link = 'remake
of';
       INSERT INTO RelacionPeli
  SELECT id_ObraSuj, id_ObraObj, 'secuela' FROM link_ext WHERE link =
'follows';

-- Población  de capítulos
       CREATE SEQUENCE secC START WITH 1 INCREMENT BY 1;
INSERT INTO capitulo
```

```
  WITH
       cap AS(
       SELECT DISTINCT title, id_obra, season_nr, episode_nr FROM
datosdb.datospeliculas JOIN obraCinem ON (serie_title = titulo AND
agno_Estreno = serie_prod_year AND kind = 'episode' AND tipoDeObra = 'tv
series'))
  SELECT secC.NEXTVAL, title, id_obra, season_nr, episode_nr
         FROM cap;

-- Eliminar elementos auxiliares
       DROP MATERIALIZED VIEW fila_ext;
       DROP MATERIALIZED VIEW link_ext;
       DROP MATERIALIZED VIEW pers_Ext;
       DROP MATERIALIZED VIEW bipersonas;
       DROP TABLE tabla_Prov;
       DROP SEQUENCE secT;

       COMMIT;
```

# Anexo 3. Consultas adicionales

En este anexo se pueden encontrar algunas de las consultas más relevantes llevadas a cabo a lo largo del proyecto:

- Número de fallos transitividad a la hora de chequear relaciones entre secuelas
- Media de personal por año
- Media de obras por año
- Media de actuaciones por año
- Media de participaciones por año
- Número de personal sin género
- Nombre de personal referenciando a más de una persona en una obra y la obra correspondiente
- Número de nombres sin género asociado
- Número de nombres asociados a actores pero que no tienen género
- Número de veces que se indica que hay información en el campo de contexto pero no la hay
- Nombres asociados a más de una persona por tener distintos lugares de nacimiento
- Nombres de personal sin lugar de nacimiento
- Nombres de actores sin personajes en una obra
- Número total de obras agrupadas por tipo
- Número total de series con mismo título que capitulos
- Número total de series con mismo título que películas
- Número total de capítulos con mismo título que películas
- Series con mismo título y año que películas
- Número de obras con mismo título pero distinto año de estreno
- Número de obras con más de un personal
- Series con capítulos
- Series con mismo título y año que películas
- Películas con mismo título y año que capítulos
- Número de nombres de personal que han participado en con distintos roles en una misma obra
- Nombres especificados en capítulos pero no en series
- Nombres especificados en capítulos y series
- Capítulos con año de estreno distinto al de la serie a la que pertenecen
- Capítulos sin número de temporada o episodio
- Número de capítulos sin referencias a series
- Series sin periodos de emisión

```
Unset
-- Número de fallos de transitividad
SELECT COUNT(*) "Fallos de transitividad"
FROM relacionPeli r1, relacionPeli r2
WHERE r1.id_Obra_Sujeto = r2.id_obra_Objeto AND r1.tipoRel = r2.tipoRel
      AND (r1.tipoRel = 'secuela') AND NOT EXISTS (SELECT * FROM
relacionPeli r WHERE r1.id_Obra_Objeto = r.id_Obra_Objeto AND
r2.id_Obra_Sujeto = r.id_Obra_Sujeto AND r.tipoRel = 'secuela');
```

```
Unset
-- Media de nuevo personal por año
WITH
       PT AS(
SELECT DISTINCT id_Personal, id_Obra
       FROM (SELECT * FROM actuarEn) UNION (select * FROM participarEn)),
       PP AS(
       SELECT id_Personal, MIN(agno_Estreno) pp
       FROM PT Natural JOIN obraCinem
       GROUP BY id_Personal),
       P  AS(
       SELECT COUNT(id_Personal) id, pp agno
       FROM PP
       GROUP BY pp
       ORDER BY pp asc)
SELECT AVG(id) FROM P;

-- Media de nuevas obras por año
WITH
       c AS(
       SELECT COUNT(id_Obra) id, agno_Estreno agno
       FROM obraCinem GROUP BY agno_Estreno ORDER BY  agno_Estreno asc)
SELECT AVG(id)
FROM c;

-- Media de nuevas actuaciones por año
WITH
       contAct AS(
       SELECT COUNT(*) cont, agno_Estreno
       FROM obraCinem NATURAL JOIN actuarEn
       GROUP BY agno_Estreno)
SELECT AVG(cont)
FROM contAct;

-- Media de nuevas participaciones por año
WITH
       contPart AS(
       SELECT COUNT(*) cont, agno_Estreno
       FROM obraCinem NATURAL JOIN participarEn
       GROUP BY agno_Estreno)
SELECT AVG(cont)
FROM contPart;

-- Nombre de personal referenciado a más de una persona en una obra y la
obra
SELECT DISTINCT d1.title, d1.production_year, d1.kind, d1.name
FROM datosdb.datospeliculas d1
```

```
JOIN datosdb.datospeliculas d2 ON(d1.title = d2.title AND
d1.production_year=d2.production_year AND d1.name=d2.name AND d1.kind =
d2.kind AND d1.gender = d2.gender)
WHERE d1.info_context=d2.info_context AND  d1.person_info<>d2.person_info;



-- Número de nombres sin género asociado
SELECT count(DISTINCT name) "Personal sin datos de genero" FROM
datosdb.datospeliculas
WHERE name IS NOT NULL AND gender IS NULL;

-- Número total de nombres asociados a actores y que no tienen género
WITH
       actores AS(
       SELECT DISTINCT name FROM datosdb.datospeliculas db
       WHERE db.role = 'actor' OR db.role = 'actress')
       SELECT count(*) "Actores sin datos de genero" FROM actores N
       WHERE NOT EXISTS (SELECT * FROM datosdb.datospeliculas db WHERE
db.name = N.name AND db.gender IS NOT NULL);
```

Unset

```
-- Número de veces que se indica que hay información de contexto pero no hay
datos
SELECT COUNT(*) "Filas con contexto pero sin texto"
FROM datosdb.datospeliculas
WHERE info_context IS NOT NULL AND person_info IS NULL;

-- Lista de nombres asociados a más de una persona por tener distinto lugar
de nacimiento
WITH
       personal AS(
       SELECT DISTINCT name, person_info
       FROM datosdb.datospeliculas
       WHERE info_context = 'birth notes')
       SELECT COUNT(*) "Persona no identificable por nombre"
       FROM personal p1 JOIN personal p2 ON p1.name = p2.name
       WHERE p1.person_info < p2.person_info;

-- Nombres de personal sin lugar de nacimiento
WITH
       personal AS(
```

```
       SELECT DISTINCT name
       FROM datosdb.datospeliculas)
SELECT COUNT(*) "Personal sin lugar de nacimiento"
FROM personal p
WHERE NOT EXISTS (SELECT * FROM datosdb.datospeliculas db
                    WHERE p.name = db.name AND info_context = 'birth notes'
                    AND person_info IS NOT NULL);

-- Nombre de actores sin personaje en una obra
with
       Part AS(
       SELECT DISTINCT name, title, production_year py
       FROM datosdb.datospeliculas
       WHERE role = 'actor' OR role = 'actress')
SELECT COUNT(*) "Participaciones de un actor en pelicula sin personaje
especificado" FROM Part p
WHERE NOT EXISTS(SELECT * FROM datosdb.datospeliculas db
             WHERE p.title = db.title AND py = production_year
             AND  p.name = db.name AND (role = 'actor' OR role = 'actress')
             AND role_name IS NOT NULL);

-- Número total de obras agrupadas por tipo (títulos)
WITH
       obra AS(
       SELECT title, production_year, kind
       FROM datosdb.datospeliculas
       WHERE title IS NOT NULL AND production_year IS NOT NULL AND kind is
not NULL)
SELECT COUNT(*) "Numero total de obra del tipo", kind
FROM datosdb.datospeliculas
GROUP BY kind;

-- Numero total de series con mismo titulo que capitulos
WITH
       serie AS(
       SELECT DISTINCT title tser FROM datosdb.datospeliculas
       WHERE kind = 'tv series'),
       capitulo AS(
       SELECT DISTINCT title tcap FROM datosdb.datospeliculas
       WHERE kind = 'episode')
SELECT DISTINCT tcap "capitulos con mismo titulo que una serie"
FROM serie, capitulo
WHERE tser = tcap;

-- Numero total de series con mismo titulo que peliculas
WITH
       serie AS(
       SELECT DISTINCT title tser FROM datosdb.datospeliculas
```

```
       WHERE kind = 'tv series'),
       pelicula AS(
       SELECT DISTINCT title tpel FROM datosdb.datospeliculas
       WHERE kind = 'movie')
SELECT count(DISTINCT tpel) "peliculas con mismo titulo que una serie"
FROM serie, pelicula
WHERE tser = tpel;

-- Numero total de capitulos con mismo titulo que peliculas
WITH
       capitulo AS(
       SELECT DISTINCT title tcap, serie_title FROM datosdb.datospeliculas
       WHERE kind = 'episode'),
       pelicula AS(
       SELECT DISTINCT title tpel FROM datosdb.datospeliculas
       WHERE kind = 'movie')
SELECT DISTINCT tpel "peliculas con mismo titulo que un capitulo"
FROM capitulo, pelicula WHERE tcap = tpel;
```

```
Unset
-- Series con mismo titulo y agno que peliculas
WITH
       serie AS(
       SELECT DISTINCT title tser, production_year aSer FROM
datosdb.datospeliculas
       WHERE kind = 'tv series'),
       pelicula AS(
       SELECT DISTINCT title tpel, production_year aPel FROM
datosdb.datospeliculas
       WHERE kind = 'movie')
SELECT tpel "peliculas con mismo titulo y agno que una serie"
FROM serie, pelicula
WHERE tser = tpel AND aSer = aPel;

-- Número de obras con mismo título pero distinto año de estreno
WITH
       obra AS(
       SELECT DISTINCT title tit, production_year agno, kind k
```

```
       FROM datosdb.datospeliculas)
SELECT DISTINCT COUNT(DISTINCT *) "Obras con mismo título pero distinto
agno"
FROM datosdb.datospeliculas db, obra
WHERE tit = title AND k = kind AND agno < production_year;

-- Número de obras con más de un personal
SELECT COUNT(COUNT(DISTINCT name)) "Obras con mas de un personal"
FROM datosdb.datospeliculas
GROUP BY title, production_year, kind
HAVING COUNT(DISTINCT name) >= 1;

-- Series con capitulos
WITH
       seriesconcap AS(
       SELECT DISTINCT title, production_year
       FROM datosdb.datospeliculas d1
       WHERE kind = 'tv series' AND
       EXISTS (SELECT * FROM datosdb.datospeliculas d2
              WHERE d1.title = d2.serie_title
              AND d1.production_year = d2.serie_prod_year
              AND d2.kind = 'episode'))
SELECT COUNT(*) "Series con capitulos" FROM seriesconcap;


-- Series con mismo titulo y año que peliculas
WITH
       serie AS(
       SELECT DISTINCT title tser, production_year aSer FROM
datosdb.datospeliculas
       WHERE kind = 'tv series'),
       pelicula AS(
       SELECT DISTINCT title tpel, production_year aPel FROM
datosdb.datospeliculas
       WHERE kind = 'movie')
SELECT tpel "peliculas con mismo titulo y agno que una serie"
FROM serie, pelicula
WHERE tser = tpel AND aSer = aPel;

-- Peliculas con mismo titulo y agno que capitulos
WITH
       capitulo AS(
       SELECT DISTINCT title tcap, production_year aCap FROM
datosdb.datospeliculas
       WHERE kind = 'episode'),
       pelicula AS(
       SELECT DISTINCT title tpel, production_year aPel FROM
datosdb.datospeliculas
```

```
      WHERE kind = 'movie')
SELECT COUNT(DISTINCT tpel) "peliculas con mismo titulo y agno que un
capitulo"
FROM capitulo, pelicula
WHERE tcap = tpel AND aCap = aPel;
```

Unset

```
-- Número de nombres personal que ha participado con distintos roles en una
misma obra
SELECT COUNT(DISTINCT db1.name) "Numero de personal que ha participado en
mas de un rol en misma peli"
FROM datosdb.datospeliculas db1, datosdb.datospeliculas db2
WHERE db1.name = db2.name AND db1.role <> db2.role
      AND NOT (db1.title = db2.title AND db1.production_year =
db2.production_year);

-- nombres especificados en capitulos pero no en series
SELECT COUNT(DISTINCT name) "Nombres especificado en caps y no series"
FROM datosdb.datospeliculas d1
WHERE kind = 'episode' AND
      NOT EXISTS (SELECT * FROM datosdb.datospeliculas d2
                  WHERE d2.title = d1.serie_title
                  AND d2.production_year = d1.serie_prod_year
                  AND d2.name = d1.name AND kind = 'serie TV');


-- nombres especificados en capitulos y series
SELECT COUNT(DISTINCT name) "Personal esp en caps y series"
FROM datosdb.datospeliculas d1
WHERE kind = 'episode' AND
      EXISTS (SELECT * FROM datosdb.datospeliculas d2
                  WHERE d2.title = d1.serie_title
                  AND d2.production_year = d1.serie_prod_year
                  AND d2.name = d1.name);

-- Capítulos con año de estreno distinto al de la serie a la que pertenecen
WITH
  cap_serie AS(
      SELECT DISTINCT t1.title, t1.production_year
      FROM datosdb.datospeliculas t1
            JOIN datosdb.datospeliculas t2 ON (t1.serie_title = t2.title
                                          AND t1.serie_prod_year =
t2.production_year)
      WHERE t1.kind = 'episode' AND t2.kind = 'tv series')
SELECT COUNT(*)
```

```sql
FROM cap_serie;

-- capitulos sin numero de temporada o de episodio
WITH
       cap AS(
       SELECT DISTINCT title t, production_year py
       FROM datosdb.datospeliculas
       WHERE kind = 'episode')
SELECT  COUNT(*)
FROM cap "capitulos sin numero de temporada o de episodio"
WHERE NOT EXISTS(SELECT * FROM datosdb.datospeliculas db
             WHERE t = title AND py = production_year
             AND (season_nr IS NOT NULL AND episode_nr IS NOT NULL));

-- Número de capitulos sin referencias a series
WITH
       cap AS(
       SELECT title t, production_year py
       FROM datosdb.datospeliculas
       WHERE kind = 'episode')
SELECT count(*)
FROM cap
WHERE NOT EXISTS(SELECT * FROM datosdb.datospeliculas
             WHERE t = title AND py = production_year
                    AND serie_title IS NOT NULL
                    AND serie_prod_year IS NOT NULL);

-- Series sin periodos de emision
WITH
       serie AS(
       SELECT title t, production_year py
       FROM datosdb.datospeliculas
       WHERE kind = 'tv series')
SELECT COUNT(*)
FROM serie s
WHERE NOT EXISTS (SELECT * FROM datosdb.datospeliculas db
                    WHERE t = title AND py = production_year
                    AND series_years IS NOT NULL);
```