BASES DE DATOS

Base de datos de obras cinematográficas

Grupo de tardes nº 13

CURSO 22/23

17 de abril de 2023

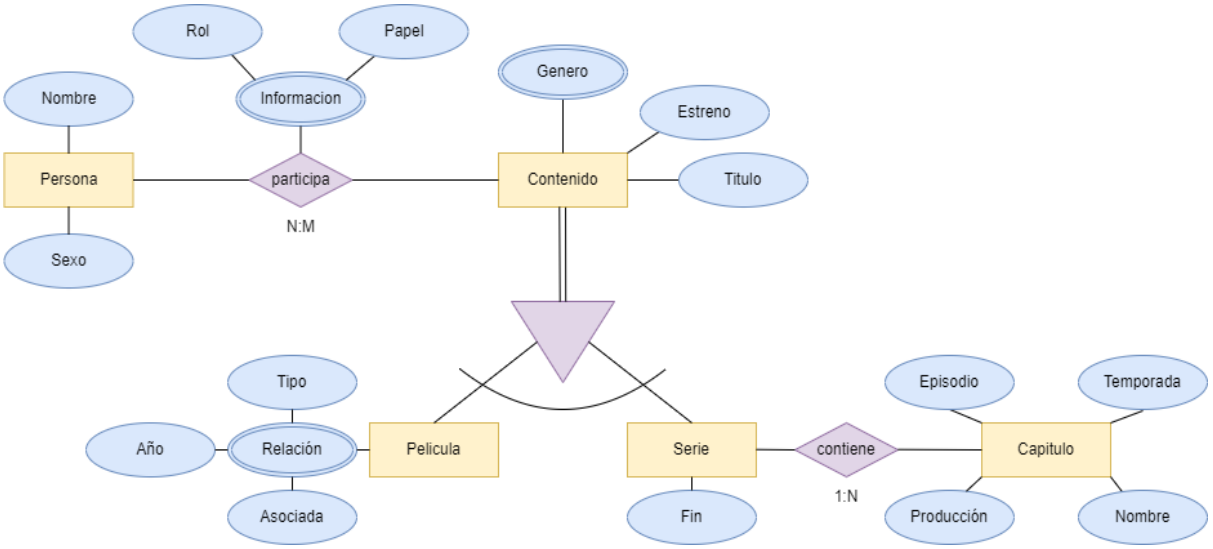
Hecho por LIZER BERNAD FERRANDO, 779035@unizar.es
GUILLERMO BAJO LABORDA, 842748@unizar.es
AXEL ISAAC PAZMIÑO ORTEGA, 817627@unizar.es

Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Parte I
Creación de una base de datos

Esquema E/R y restricciones:



En el esquema E/R aparecen un total de 5 entidades distintas: *Persona, Contenido* (que tiene cobertura total con *Pelicula* y *Serie*) y *Capitulo,* cada una con sus respectivos atributos. Se han creado los atributos considerados más relevantes y de mayor utilidad. Se distinguen de dos tipos: normales y multivaluados (que pueden tener sus propios atributos como es el caso). Para evitar que un contenido pueda ser simultáneamente *Pelicula* y *Serie* se ha utilizado la especialización disjunta. A su vez, existen dos relaciones entre las entidades, una N:M (*participa*) entre *Persona* y *Contenido* y una 1:N (*contiene*) entre *Serie* y *Capitulo,* ya que las series generalmente contienen capítulos.

A continuación se muestran alguna de las principales restricciones encontradas:

Contenido:

- La fecha de estreno debe de ser mayor que 1895 en *Peliculas* y que 1946 en *Series* (fecha de estreno de la primera película y serie). A su vez, ha de ser menor que 2024 en ambos casos.
- Un contenido no puede ser película y serie simultáneamente.

Pelicula:

- Una película no puede estar relacionada con una serie/capítulo.
- Una película no puede tener capítulos.
- Una película secuela de otra debe tener fecha de estreno posterior
- Una película precuela de otra debe tener fecha de estreno anterior

Capítulo:

- Todo capítulo ha de pertenecer a una serie registrada.
- Tanto *episodio* como *temporada* han de ser valores positivos.

Por otra parte, cabe destacar ciertas alternativas que se han planteado a la hora de hacer el diseño del esquema E/R:

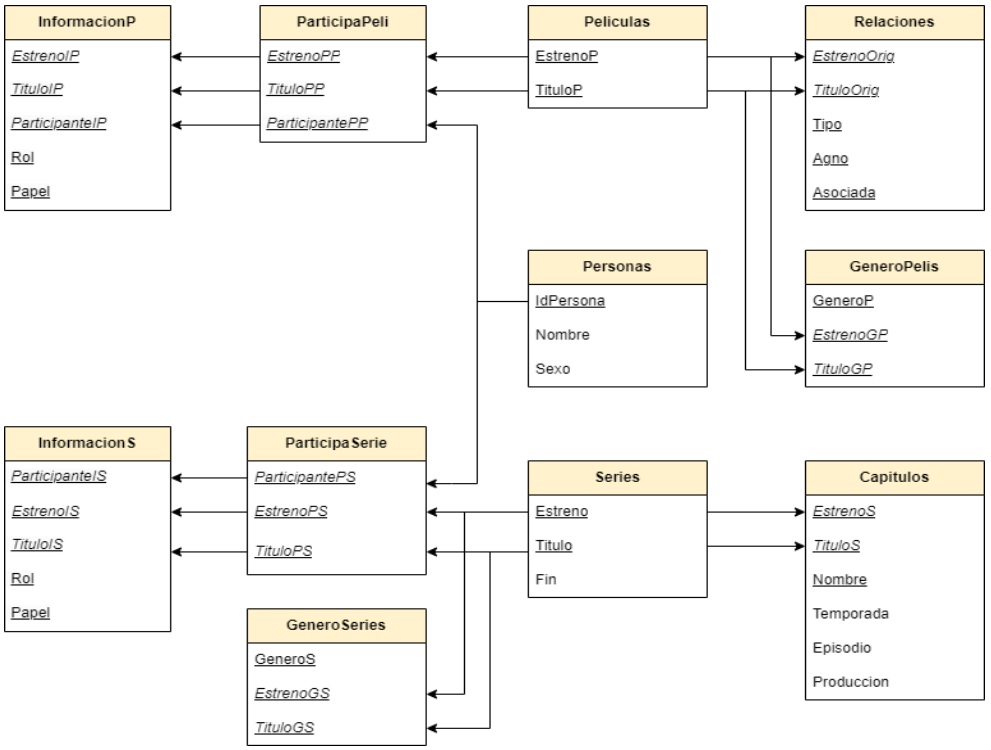
- El uso de 3 relaciones de participación entre *persona* y *contenido.*
- El uso de una relación N:M reflexiva para las relaciones de unas películas con otras.
- Almacenamiento de información de Capitulos en la propia tabla Series.

Respecto al primer punto, se ha considerado sustituir la relación N:M *participa* por tres relaciones distintas, todas N:M (*actúa*, *dirige* y *contribuye*), en cuyo caso la primera tendría un atributo *personaje* y la última un atributo *papel,* pero finalmente se has escogido la alternativa de añadir un atributo multivaluado a la relación *participa,* lo cual ahorra introducir tres relaciones muy similares.

Respecto al segundo punto, se ha planteado introducir una relación N:M reflexiva en la entidad *Peliculas* para almacenar la información de películas relacionadas. Sin embargo la decisión final ha sido hacerlo con el atributo multivaluado que se aprecia en el modelo E/R ya que este permite guardar relaciones de una película con otras películas que no se encuentran en la base mientras que la relación N:M solamente permite guardar relaciones con películas ya pobladas en la base.

Respecto al tercer punto, se ha planteado guardar la información de cada capítulo en la entidad *Serie.* Sin embargo al analizar los datos ha quedado patente que hay series que no contienen ningún capítulo, lo que supone un problema con el modelo propuesto ya que se pierde parte de la información solicitada (todas las series sin capítulos). Por consiguiente se ha decidido crear la entidad *Capitulo* que se puede observar en el diseño final del modelo E/R*,* que almacena la información de cada capítulo mientras que en *Serie* simplemente se almacena sólo la información general de la serie.

Una vez el diseño E/R ha sido considerado satisfactorio, se ha llevado a cabo el modelo relacional y la normalización del mismo. En el proceso de Normalización no ha sido necesario realizar apenas cambios pues el modelo ya cumple con las propiedades de las formas normales exigidas por el enunciado, con alguna excepción. Estas excepciones son los atributos multivaluados del modelo, los cuales violan la 1FN y por tanto han sido transformados con el método habitual en el modelo Relacional. Tras este pequeño cambio el Modelo Relacional está normalizado hasta la FNBC, y es el siguiente:



Sentencias SQL de creación de tablas:

La creación de las tablas de la base de datos se ha llevado a cabo mediante el fichero *crear_bd.sql*. El modelo cuenta con un total de 11 tablas, basadas en el modelo relacional normalizado propuesto anteriormente. También se han añadido checks que comprueban diversas restricciones establecidas. Así mismo, se dispone de un fichero *borrar_bd.sql* para borrar las tablas creadas en orden inverso al de creación.

```
CREATE TABLE Personas(
    idPersona    NUMBER(5) PRIMARY KEY,
    Nombre       VARCHAR(200) NOT NULL,
    Sexo         VARCHAR(1),
    CONSTRAINT ck_Sexo CHECK (Sexo IN ('f', 'm'))
);

CREATE TABLE Peliculas(
    EstrenoP      NUMBER(4),
    TituloP       VARCHAR(200),
    CONSTRAINT Peliculas_PK PRIMARY KEY (EstrenoP, TituloP),
    CONSTRAINT ck_Estreno CHECK (EstrenoP > 1895 AND EstrenoP < 2024)
);

CREATE TABLE GeneroPelis(
    GeneroP      VARCHAR(200),
    EstrenoGP    NUMBER(4),
    TituloGP     VARCHAR(200),
    FOREIGN KEY (EstrenoGP, TituloGP) REFERENCES Peliculas(EstrenoP,
```

```
    TituloP),
    CONSTRAINT GenerosP_PK PRIMARY KEY (GeneroP, EstrenoGP, TituloGP)
);

CREATE TABLE Relaciones(
    EstrenoOrig  NUMBER(4),
    TituloOrig   VARCHAR(200),
    Tipo         VARCHAR(200),
    Agno         NUMBER(4),
    Asociada     VARCHAR(200),
    FOREIGN KEY (EstrenoOrig, TituloOrig) REFERENCES Peliculas(EstrenoP,
    TituloP),
    CONSTRAINT Rels_PK PRIMARY KEY (EstrenoOrig, TituloOrig, Tipo, Agno,
    Asociada),
    CONSTRAINT ck_AgnoAsociada CHECK (Agno > 1895 AND Agno < 2024)
);

CREATE TABLE Series(
    Estreno      NUMBER(4),
    Titulo       VARCHAR(200),
    Fin          NUMBER(4),
    CONSTRAINT Series_PK PRIMARY KEY (Estreno, Titulo),
    CONSTRAINT ck_EstrenoS CHECK (Estreno > 1946 AND Estreno < 2024),
    CONSTRAINT ck_FinS CHECK (Fin > 1946 AND Fin < 2024 AND Fin >= Estreno)
);

CREATE TABLE GeneroSeries(
    GeneroS      VARCHAR(200),
    EstrenoGS    NUMBER(4),
    TituloGS     VARCHAR(200),
    FOREIGN KEY (EstrenoGS, TituloGS) REFERENCES Series(Estreno, Titulo),
    CONSTRAINT GenerosS_PK PRIMARY KEY (GeneroS, EstrenoGS, TituloGS)
);

CREATE TABLE Capitulos(
    EstrenoS     NUMBER(4),
    TituloS      VARCHAR(200),
    Nombre       VARCHAR(200) NOT NULL,
    Temporada    NUMBER(1),
    Episodio     NUMBER(2),
    Produccion   NUMBER(4) NOT NULL,
    CONSTRAINT Capitulos_PK PRIMARY KEY (EstrenoS, TituloS, Nombre),
    FOREIGN KEY (EstrenoS, TituloS) REFERENCES Series(Estreno, Titulo),
    CONSTRAINT ck_Capitulo CHECK (Temporada > 0 AND Episodio > 0)
);
```

```
CREATE TABLE ParticipaPeli(
    EstrenoPP  NUMBER(4),
    TituloPP   VARCHAR(200),
    ParticipantePP NUMBER(5),
    FOREIGN KEY (EstrenoPP, TituloPP) REFERENCES Peliculas(EstrenoP,
    TituloP),
    FOREIGN KEY (ParticipantePP) REFERENCES Personas(idPersona),
    CONSTRAINT ParticipaPeli_PK PRIMARY KEY (EstrenoPP, TituloPP,
    ParticipantePP)
);

CREATE TABLE InformacionP(
    EstrenoIP       NUMBER(4),
    TituloIP        VARCHAR(200),
    ParticipanteIP  NUMBER(5),
    Rol             VARCHAR(200),
    Papel           VARCHAR(200),
    FOREIGN KEY (EstrenoIP, TituloIP, ParticipanteIP) REFERENCES
    ParticipaPeli(EstrenoPP, TituloPP, ParticipantePP),
    CONSTRAINT InfP_PK PRIMARY KEY (EstrenoIP, TituloIP, ParticipanteIP,
    Rol, Papel)
);

CREATE TABLE ParticipaSerie(
    ParticipantePS  NUMBER(5),
    EstrenoPS       NUMBER(4),
    TituloPS        VARCHAR(200),
    FOREIGN KEY (EstrenoPS, TituloPS) REFERENCES Series(Estreno, Titulo),
    FOREIGN KEY (ParticipantePS) REFERENCES Personas(idPersona),
    CONSTRAINT ParticipaSerie_PK PRIMARY KEY (ParticipantePS, EstrenoPS,
    TituloPS)
);

CREATE TABLE InformacionS(
    ParticipanteIS  NUMBER(5),
    EstrenoIS       NUMBER(4),
    TituloIS        VARCHAR(200),
    Rol             VARCHAR(200),
    Papel           VARCHAR(200),
    FOREIGN KEY (ParticipanteIS, EstrenoIS, TituloIS) REFERENCES
    ParticipaSerie(ParticipantePS, EstrenoPS, TituloPS),
    CONSTRAINT InfS_PK PRIMARY KEY (ParticipanteIS, EstrenoIS, TituloIS,
    Rol, Papel)
);
```

Parte II
Introducción de datos y ejecución de consultas

Población de la base:

La población de la base ha sido un proceso relativamente sencillo que se ha llevado a cabo siguiendo el siguiente procedimiento:

Inicialmente se ha partido del recurso csv *DatosPeliculas* proporcionado en la plataforma Moodle. De este fichero se han seleccionado los campos deseados y se han eliminado los duplicados, para así, creando un archivo ctl cargar los datos en la tabla correspondiente. Este proceso se ha repetido un total de 11 veces, una por tabla.

Durante el proceso de población se han encontrado diversos problemas, los cuales han sido subsanados por completo con las correspondientes correcciones. Algunos de estos problemas han sido causados por el gran tamaño del excel proporcionado, que provocan que herramientas como la de borrado de duplicados borre más datos de los que debería. Esto se ha solucionado ordenando alfabéticamente las columnas de la hoja de cálculo que nos causan este problema ya que esto permite que el borrado de duplicados funcione con normalidad.

Otro problema encontrado durante la población se ha producido en las tablas InformaciónP e InformaciónS. El problema se produce debido a que en los datos de aquellos participantes de una película que no son actores el campo “Papel” aparece vacío. Esto genera un conflicto ya que “Papel” forma parte de la clave primaria de las tablas anteriormente mencionadas. Para subsanar este problema se ha decidido sustituir todos los valores NULL de “Papel” por la cadena ‘Ninguno’ evitando así el error que se produce al poblar las tablas.

Finalmente, de cara a tener datos más manejables, al poblar la tabla *Series* se decidió sustituir el atributo *Periodo* por los atributos *Fin* y *Estreno*. Como se apreciará posteriormente en la segunda consulta, esto facilita bastante el manejo de dicha información, al no tener que descomponer la cadena correspondiente al periodo de emisión en año de inicio y de fin de emisión. Tras los cambios, directamente se almacenan tanto el inicio como el fin de la emisión y fácilmente se puede acceder a ellos sin necesidad de convertir la cadena correspondiente al periodo en dos enteros.

Consultas:

1. **Listar el título de las películas de género familiar que sólo han sido interpretadas por actrices**

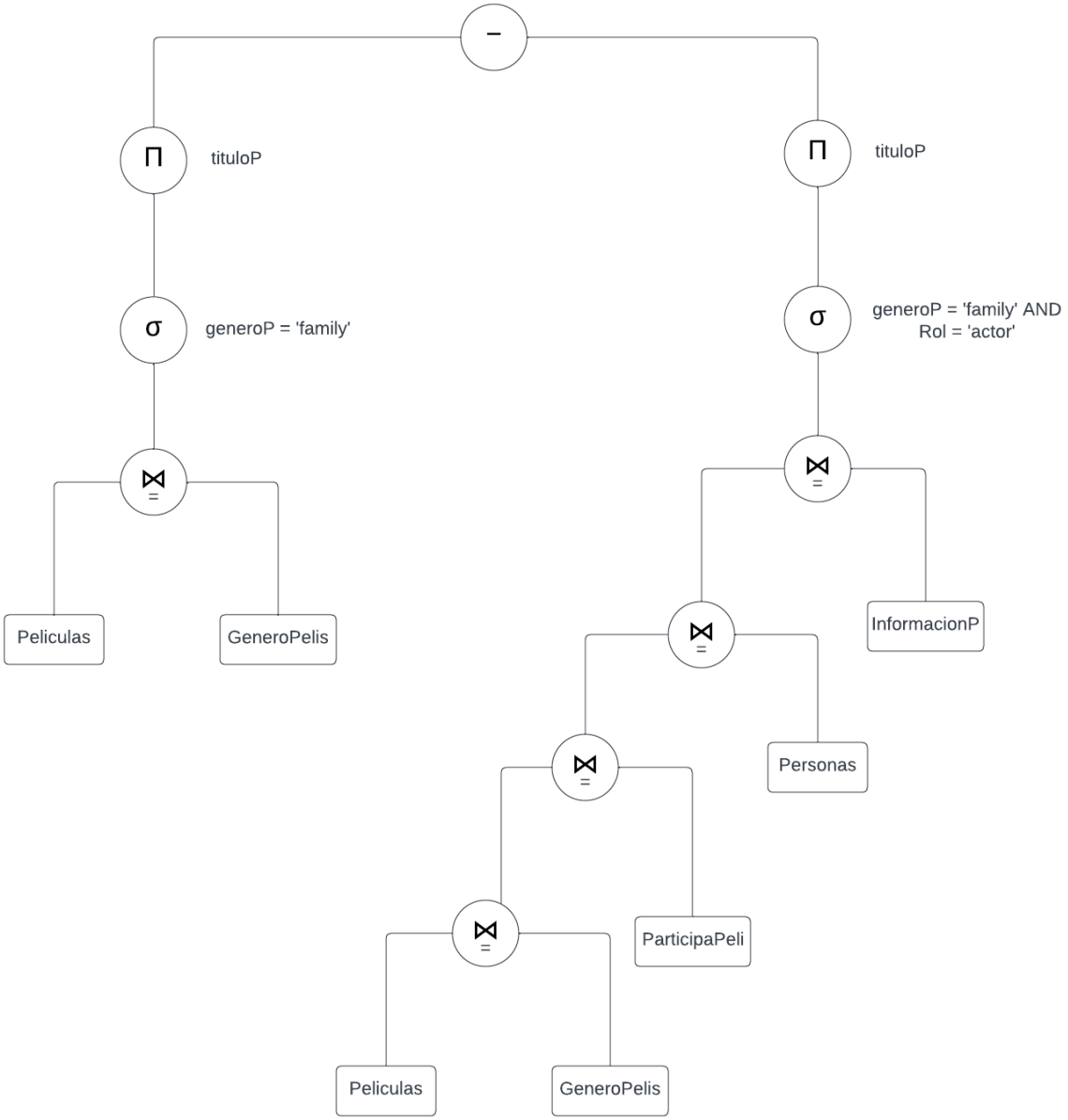
   En esta consulta se han realizado dos "subconsultas" importantes:
   La primera consiste en listar el título de las películas de género familiar.
   La segunda consiste en listar el título de las películas de género familiar que han sido interpretadas por al menos un actor varón.
   De esta forma el conjunto de películas obtenido en la segunda subconsulta siempre es un subconjunto del conjunto obtenido en la primera y la solución a nuestra consulta original radica precisamente en aquellos elementos de la primera consulta que no se encuentran en la segunda.
   Es por ello que se relacionan ambas subconsultas mediante la condición *NOT IN* en el *WHERE* de la primera subconsulta y se obtiene el resultado esperado.

```
SELECT DISTINCT TituloP
FROM Peliculas
    JOIN GeneroPelis
    ON EstrenoP = EstrenoGP
        AND TituloP = TituloGP
WHERE GeneroP = 'family'
    AND TituloP NOT IN (
        SELECT DISTINCT TituloP
        FROM Peliculas
            JOIN GeneroPelis
            ON EstrenoP = EstrenoGP
                AND TituloP = TituloGP
            JOIN ParticipaPeli
            ON EstrenoP = EstrenoPP
                AND TituloP = TituloPP
            JOIN Personas
            ON ParticipantePP = idPersona
            JOIN InformacionP
            ON EstrenoPP = EstrenoIP
                AND TituloPP = TituloIP
                AND ParticipantePP = ParticipanteIP
        WHERE GeneroP = 'family'
            AND Rol = 'actor'
    );
```

Salida:

```
TITULOP
------------------------------
The Language You Cry In
```

Árbol Sintáctico de la consulta 1:



2. **Listar el nombre de los directores que han dirigido al menos seis series distintas en la década de los 90**

Ésta consulta es bastante simple y lo único destacable son las cláusulas del WHERE donde se comprueba que las tuplas seleccionadas sean de directores y además se comprueba que la serie que ha dirigido esa persona se encuentre producida en los 90. Para comprobar esto último en realidad se comprueban cuatro posibles casos correctos mediante OR:

- La serie comenzó su producción entre 1990 y 1999.
- La serie terminó su producción entre 1990 y 1999.
- La serie comenzó su producción antes de 1990 y terminó después de 1999.
- La serie comenzó su producción antes de 1990 y aún no ha terminado.

Todos los casos mencionados anteriormente implican que una parte del periodo de producción de la serie se ha llevado a cabo durante los noventa.

Finalmente una vez están todas las tuplas correctas seleccionadas se agrupan por el nombre del director y se cuenta cuántas agrupaciones tienen al menos 6 series distintas.

```
SELECT Nombre
FROM Personas
    JOIN ParticipaSerie
    ON idPersona = ParticipantePS
    JOIN Series
    ON Estreno = EstrenoPS
        AND Titulo = TituloPS
    JOIN InformacionS
    ON EstrenoPS = EstrenoIS
        AND TituloPS = TituloIS
        AND ParticipantePS = ParticipanteIS
WHERE Rol = 'director'
    AND ((Estreno >= 1990
            AND Estreno <= 1999
        )
        OR (Fin >= 1990
            AND Fin <= 1999
        )
        OR (Estreno < 1990
            AND (Fin > 1999
                OR Fin IS NULL
            )
        )
    )
GROUP BY Nombre
HAVING COUNT(DISTINCT Titulo) >= 6;
```

Salida:

NOMBRE

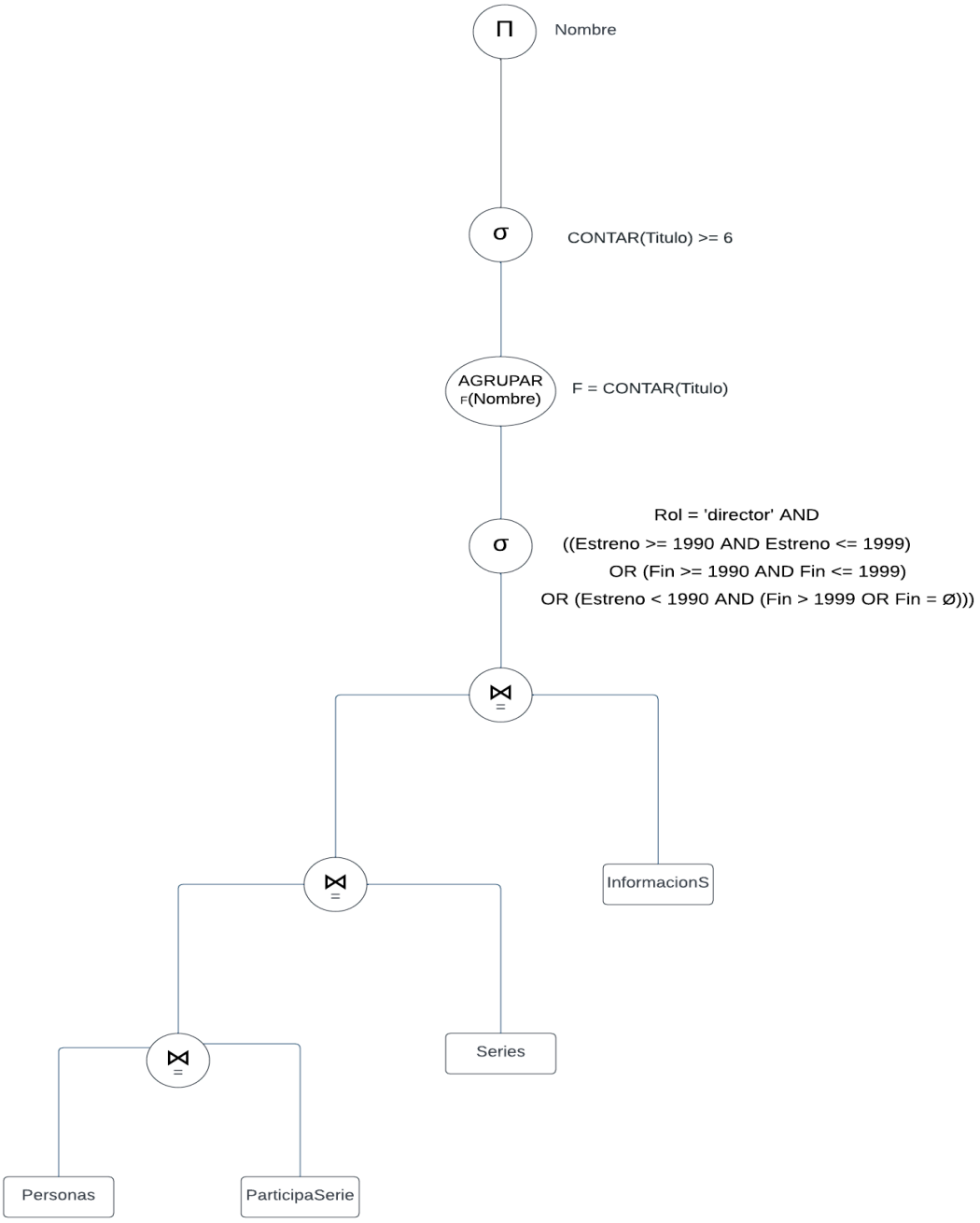
Plaza, José Antonio

Alvarez, Juan Luis

Garcia-Loureda, Ruben

Lara, Orestes

Árbol sintáctico de la consulta 2:



3. **Obtener el número de personajes distintos que han sido interpretados por al menos cuatro actores o actrices distintos y que aparecen en películas de alguna saga (es decir, que son secuelas/precuelas unas de otras)**

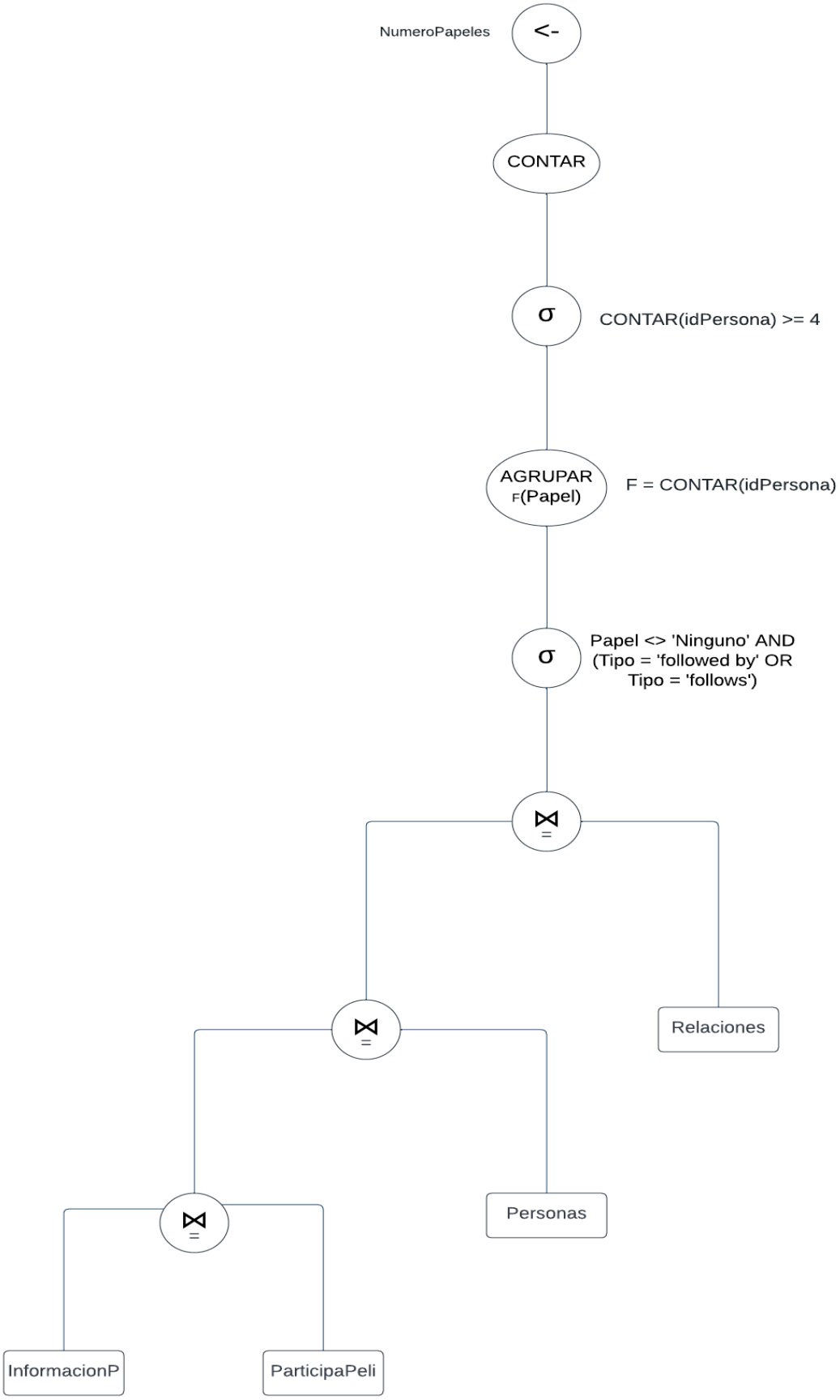
   En esta consulta se seleccionan aquellas filas que contienen personajes que aparecen en películas con una relación de secuela o precuela. Después se agrupan los personajes y se seleccionan aquellos grupos en los que el personaje es interpretado por al menos cuatro actores distintos.
   Finalmente se cuenta el número de grupos que han sido seleccionados y se muestra como la solución *NumeroPapeles*.

```
SELECT COUNT(COUNT(Papel)) AS NumeroPapeles
FROM InformacionP
    JOIN ParticipaPeli
    ON EstrenoPP = EstrenoIP
        AND TituloPP = TituloIP
        AND ParticipanteIP = ParticipantePP
    JOIN Personas
    ON ParticipantePP = IdPersona
    JOIN Relaciones
    ON EstrenoIP = EstrenoOrig
        AND TituloIP = TituloOrig
WHERE Papel <> 'Ninguno'
    AND (Tipo = 'followed by'
        OR Tipo = 'follows'
    )
GROUP BY Papel
HAVING COUNT(DISTINCT IdPersona) >= 4;
```

Salida:

```
NUMEROPAPELES
-------------------------
25
```

Árbol sintáctico de la consulta 3:



Parte III
Diseño físico

Análisis de rendimiento de las consultas:

Se han obtenido informes y estadísticas sobre la ejecución de las consultas para analizar el coste de ejecución de sentencias SQL y plantear posibles mejoras sobre las mismas.

A continuación para cada consulta se mostrará los informes obtenidos mediante los comandos *EXPLAIN PLAN FOR <consulta SQL>* y *SELECT PLAN_TABLE_OUTPUT FROM TABLE(DBMS_XPLAN.DISPLAY()).*

Consulta 1

Id	Operación	Name	Filas	Bytes	Coste (%CPU)	Tiempo
0	SELECT STATEMENT		2	612	212 (2)	00:00:01
1	HASH UNIQUE		2	612	212 (2)	00:00:01
2	HASH JOIN ANTI		2	612	211 (2)	00:00:01
3	INDEX RANGE SCAN	GENEROSP_ PK	215	43860	4 (0)	00:00:01
4	VIEW		2835	282K	207 (2)	00:00:01
5	NESTED LOOPS SEMI		2835	1201K	207 (2)	00:00:01
6	TABLE ACCESS FULL		37899	8031K	205 (1)	00:00:01
7	INDEX UNIQUE SCAN		16	3472	0 (0)	00:00:01

siendo para cada id los siguientes predicados en la operación:

2 - access("TITULOGP"="TITULOP")
3 - access("GENEROPELIS"."GENEROP"='family')
6 - filter("INFORMACIONP"."ROL"='actor')
7 - access("GENEROPELIS"."GENEROP"='family' AND "ESTRENOGP"="ESTRENOIP" AND "TITULOGP"="TITULOIP")

Consulta 2

Id	Operación	Name	Filas	Bytes	Coste (%CPU)	Tiempo
0	SELECT STATEMENT		763	152K	93 (2)	00:00:01
1	HASH GROUP BY		763	152K	93 (2)	00:00:01
2	VIEW	VM_NWVW_1	763	152K	93 (2)	00:00:01
3	HASH GROUP BY		763	352K	93 (2)	00:00:01
4	HASH JOIN		763	352K	92 (0)	00:00:01
5	HASH JOIN SEMI		763	266K	24 (0)	00:00:01
6	TABLE ACCESS FULL	INFORMACIONS	763	171K	19 (0)	00:00:01
7	TABLE ACCESS FULL	SERIES	1502	187K	5 (0)	00:00:01
8	TABLE ACCESS FULL	PERSONAS	35988	4041K	68 (0)	00:00:01

siendo para cada id los siguientes predicados en la operación:

1 - filter(COUNT("$vm_col_1")>=6)
4 - access("IDPERSONA"="PARTICIPANTEIS")
5 - access("ESTRENO"="ESTRENOIS" AND "TITULO"="TITULOIS")
6 - filter("INFORMACIONS"."ROL"='director' AND "ESTRENOIS">1946 AND
"ESTRENOIS"<2024)
7 - filter("SERIES"."ESTRENO"<1990 AND ("SERIES"."FIN" IS NULL OR
"SERIES"."FIN">1999) OR "SERIES"."ESTRENO">=1990 AND "SERIES"."EST
RENO"<=1999 OR "SERIES"."FIN">=1990 AND "SERIES"."FIN"<=1999)

Consulta 3

Id	Operación	Name	Filas	Bytes	Coste (%CPU)	Tiempo
0	SELECT STATEMENT		1	128	119 (1)	00:00:01
1	SORT AGGREGATE		1	128	119 (1)	00:00:01
2	FILTER				119 (1)	00:00:01
3	HASH GROUP BY		1	128	119 (1)	00:00:01
4	VIEW	VW_DAG_0	126	16128	119 (1)	00:00:01
5	HASH GROUP BY		126	57960	119 (1)	00:00:01
6	NESTED LOOPS		126	57960	118 (0)	00:00:01
7	NESTED LOOPS		126	56322	118 (0)	00:00:01
8	TABLE ACCESS FULL		115	24955	3 (0)	00:00:01
9	INDEX RANGE SCAN		1	230	1 (0)	00:00:01
10	INDEX UNIQUE SCAN		1	13	0 (0)	00:00:01

siendo para cada id los siguientes predicados en la operación:

2 - filter(COUNT("ITEM_1")>=4)
8 - filter("RELACIONES"."TIPO"='followed by' OR "RELACIONES"."TIPO"='follows')
9 - access("INFORMACIONP"."ESTRENOIP"="ESTRENOORIG" AND
"INFORMACIONP"."TITULOIP"="TITULOORIG")
filter("INFORMACIONP"."PAPEL"<>'Ninguno')
10 - access("PARTICIPANTEIP"="IDPERSONA")

Respecto a la consulta 1, se ha probado a crear una vista para reducir los costes de tiempo y espacio, pero al ejecutar de nuevo los comandos para obtener los informes, se ha visto que la mejora en los aspectos mencionados ha sido mínima.

Para la consulta 2 se decidió por crear un índice en el rol de participaS, para hacer más eficiente la búsqueda, pero al hacer pruebas ejecutando la misma consulta pero con el nuevo índice tampoco se distinguen mejoras significativas.

Como último para la consulta 3 decidimos no implementar ningún cambio en cuanto al diseño físico, queda planteado al igual que en las consultas anteriores, guardar el "join" de cada consulta y gestionarlo con triggers para ahorrar tiempo y espacio en la consulta.

Triggers:

A continuación se van a listar todas las restricciones que Oracle no puede verificar con la estructura de tablas definidas ni con las restricciones de integridad de los CREATE TABLE:

- Quien participa de actor en una película y/o serie y es una persona de género masculino, su rol debe ser 'actor'.
- Quien participa de actor en una película y/o serie y es una persona de género femenino, su rol debe ser 'actress'.
- Si añadimos un capítulo de una serie, en caso de especificar el número de episodio y la temporada, no pueden ser los mismos de alguno ya existente.

Para estas restricciones se han implementado triggers.

integridadGeneroPelis: Se crea la variable sexp para poder almacenar el género de la persona que participa en la película, como actor o actriz y comprobar la consistencia.
En caso de no cumplirse cualquiera de las 2 primeras restricciones descritas anteriormente no se permitirá la inserción de la tupla, lanzando su respectivo mensaje de error.

```
CREATE TRIGGER integridadGeneroPelis
BEFORE INSERT ON informacionP
FOR EACH ROW
DECLARE sexp VARCHAR(1);
BEGIN
    SELECT sexo INTO sexp
    FROM personas
    WHERE (personas.idpersona=:NEW.participanteip);
     -- Si pretende introducir una persona con rol 'actor' pero es mujer, dispara el
trigger.
    IF (:NEW.rol = 'actor') AND (sexp = 'f') THEN
         RAISE_APPLICATION_ERROR(-20000,'Si quien participa en la pelicula de actor
su genero es femenino, su rol debe ser "actress": violada');
    END IF;

     -- Si pretende introducir una persona con rol 'actress' pero es hombre, dispara
el trigger.
    IF (:NEW.rol = 'actress') AND (sexP = 'm') THEN
         RAISE_APPLICATION_ERROR(-20001,'Si quien participa en la pelicula de actor
su genero es masculino, su rol debe ser "actor": violada');
    END IF;
END;
/
```

integridadGeneroSeries: Mismo comportamiento que el anterior trigger pero para series, con la variable sexs en vez de sexp.

```
CREATE TRIGGER integridadGeneroSeries
BEFORE INSERT ON informacions
FOR EACH ROW
DECLARE sexs VARCHAR(1);
BEGIN
    SELECT sexo INTO sexs
    FROM personas
    WHERE (personas.idpersona = :NEW.participanteis);
     -- Si pretende introducir una persona con rol 'actor' pero es mujer, dispara el
trigger.
    IF (:NEW.rol = 'actor') AND (sexs = 'f') THEN
         RAISE_APPLICATION_ERROR(-20000,'Si quien participa en la serie de actor su
genero es femenino, su rol debe ser "actress": violada');
    END IF;

     -- Si pretende introducir una persona con rol 'actress' pero es hombre, dispara
el trigger.
    IF (:NEW.rol = 'actress') AND (sexs = 'm') THEN
         RAISE_APPLICATION_ERROR(-20001,'Si quien participa en la serie de actor su
genero es masculino, su rol debe ser "actor": violada');
    END IF;
END;
/
```

integridadGeneroSeries: Se disparará el trigger, en caso de que no sea null la temporada y episodio. Se crean las variables temp y ep para guardar la temporada y episodio que sea igual que la de la nueva tupla a añadir, en caso de que exista. Si así fuera, se lanza un mensaje de error y se aborta la ejecución del insert.

```
CREATE TRIGGER integridadCapitulos
BEFORE INSERT ON capitulos
FOR EACH ROW
-- El trigger se dispara solo con Capitulos con los valores de temporada y episodio
asignados (no nulos)
WHEN ((NEW.temporada IS NOT NULL) AND (NEW.episodio IS NOT NULL))
DECLARE temp NUMBER(1);
ep NUMBER(2);
BEGIN
    SELECT temporada, episodio INTO temp, ep
    FROM capitulos
     WHERE (capitulos.titulos=:NEW.titulos) AND (capitulos.estrenos=:NEW.estrenos )
AND (capitulos.temporada=:NEW.temporada) AND (capitulos.episodio=:NEW.episodio);
    IF (temp IS NOT NULL) AND (ep IS NOT NULL) THEN
          RAISE_APPLICATION_ERROR(-20002,'Si se especifica temporada y episodio al
añadir capitulo, este no puede tener los mismos que otro capitulo de la misma
serie: violada');
    END IF;
```

```
END;
/
```

Coordinación del grupo

Para la realización de esta segunda base de datos han sido necesarias 6 horas de trabajo conjunto en los horarios de prácticas y alrededor de 12 y 15 horas de trabajo individual por cada miembro del equipo. Respecto al reparto de trabajo, se ha procurado evitar realizar distinciones para que todos los miembros del grupo participaran en todos los apartados por igual, de forma que se tienen varios puntos de vista y se puede debatir sobre las decisiones tomadas a lo largo del desarrollo de la base de datos. En cambio, al igual que en la práctica anterior, ciertos apartados han sido llevados a cabo en mayor medida por un integrante u otro de forma voluntaria más que por asignación de trabajo. En definitiva, se podría decir que todos los integrantes del equipo han participado de forma bastante equitativa en la elaboración del proyecto. Por tanto no ha habido problemas de organización y se ha podido desarrollar la base de datos con relativa fluidez.

Respecto al trabajo en sí, ha sido completado con bastante más soltura que la primera base de datos puesto que los integrantes del grupo conocen mejor las diversas herramientas disponibles. El diagrama E/R, así como el modelo relacional y la normalización del mismo se pudo llevar a cabo en prácticamente la primera sesión de prácticas. La creación de tablas se completó con bastante prontitud puesto que los errores en la creación de alguna tabla ya se trataron en la primera base y fueron rápidamente corregidos. La población fue la parte en la que se descubrieron la mayoría de problemas del diagrama entidad relación original y se propusieron ideas para subsanarlos. Una vez poblados los datos, las consultas y los triggers se completaron con relativa facilidad, así como la optimización y posibles mejoras a tener en cuenta.