GRUPO T07:
PRÁCTICA 2
BASE DE DATOS
DE PELÍCULAS

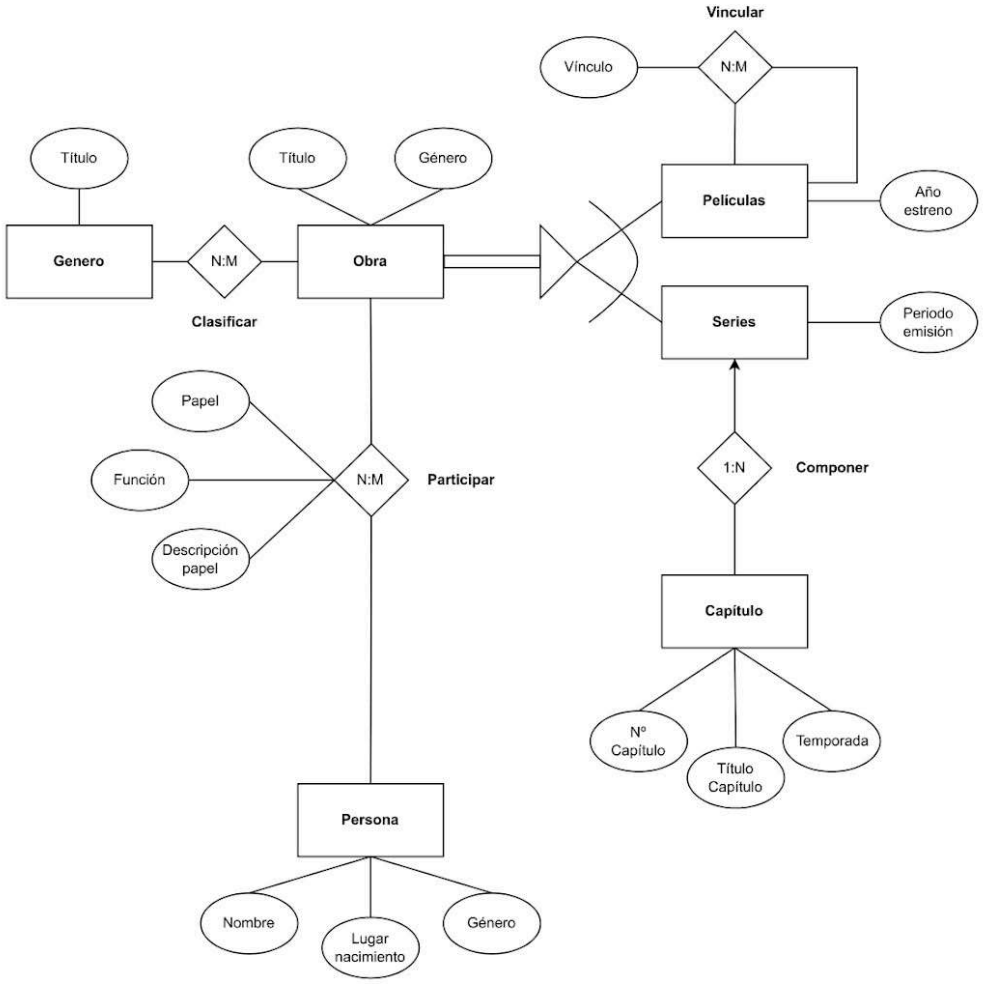
Antonio González Almela 143045@unizar.es
Alex Asensio Boj 874252@unizar.es

ÍNDICE

1. CREACIÓN DE UNA BASE DE DATOS

1.1. ESQUEMA E/R:

El esquema E/R propuesto es el siguiente:

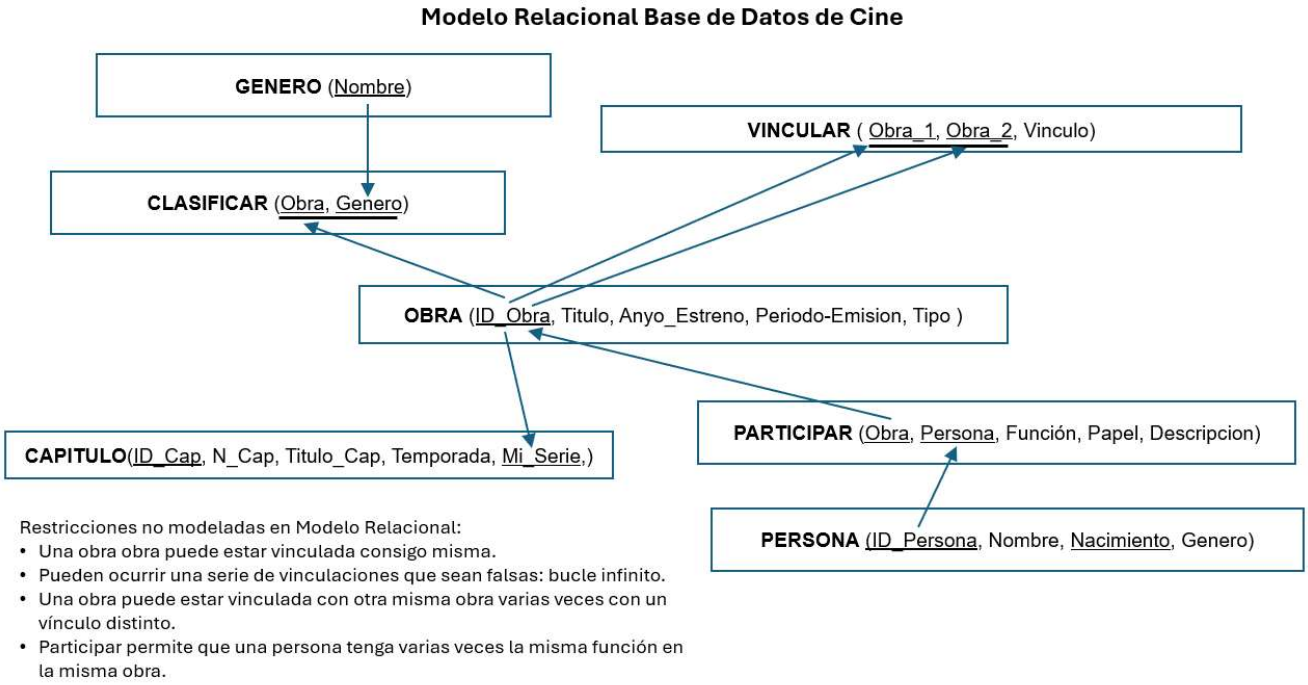


Restricciones NO modeladas por esquema E/R:

- Una obra puede estar vinculada consigo misma.
- Pueden ocurrir una serie de vinculaciones que sean falsas.
- Una obra puede estar vinculada con otra varias veces con un vínculo distinto.

1.2. ESQUEMA RELACIONAL

El esquema relacional correspondiente al esquema anterior es el siguiente:



NOTA: para la tabla PARTICIPAR no se ha definido clave única. Podríamos haber optado por incluir un contador generado por secuencia. Sin embargo, hemos decidido no usar este contador, ya que no resuelve el problema de que aparezcan personas que pueden desempeñar varias veces la misma función en una obra. Mediante un trigger, en la sección correspondiente, garantizaremos que la repetición en los atributos Obra, Persona, Funcion, conlleva que alguno de los atributos Papel o Descripcion es diferente en esta nueva tupla, en comparación con las existentes. Por ejemplo, una misma persona puede desempeñar varios papeles diferentes como actor, o un guionista escribir la historia original y también el guión de la obra.

NORMALIZACIÓN:

- Primera Forma Normal (1FN): No hay en las tablas propuestas ningún atributo multivaluado. Por tanto la base de datos está en 1FN.
- Segunda Forma Normal (2FN): no hay dependencias funcionales con parte de la clave. La tabla CLASIFICAR y la tabla VINCULAR tienen claves múltiples. En el caso de CLASIFICAR no hay relación parcial de nada con parte de la clave, la tabla no tiene más atributos. Para VINCULAR, conocida una obra o un vínculo no es posible deducir la otra obra vinculada, o conocidas dos obras no se puede deducir el vínculo. Por tanto también está en 2FN
- Tercera Forma Normal (3FN): No hay dependencias funcionales transitivas. La tabla OBRA tiene la asociación de atributos (Título, Año_Produccion) para película y (Titulo, Periodo_Emision) para series que son candidatas a clave primaria, por tanto no entran en el análisis de dependencias funcionales transitivas. En el resto de tablas no hay dependencias transitivas.
- FNBC: todas las dependencias funcionales dependen de la clave. Excepto en la tabla OBRA donde hay dos candidatas a clave primaria, no hay cualquier dependencia que no sea con la clave.

1.3. CREACIÓN DE TABLAS

Para modelar la jerarquía de Obra con Película y Serie, hemos optado por llevar todos los atributos de Película y Serie a una sola tabla, donde agregamos un atributo Tipo con valores 'P' para Película y 'S' para Serie. Al tomar esta decisión los candidatos a clave primaria de la tabla OBRA: (Título, Año_Produccion) y (Titulo, Periodo_Emision) no pueden ser implantados, ya que Año_Produccion o Periodo-Emision tomarán valor NULL en alguna ocasión, lo que les imposibilita para ser definidos como clave primaria.

Con la creación de tablas hemos incluido ciertas comprobaciones sobre los datos que serán introducidos:

- En la tabla OBRA comprobamos que Tipo toma solo los valores 'P' o 'S', esto garantiza la implantación de la jerarquía como cobertura total y especialización disjunta: A = (B ∪ C) y (B ∩ C) = Ø.
- Mediante un CHECK comprobamos que en VINCULAR no haya auto vinculaciones.
- En tabla PERSONA comprueba que el atributo Genero toma siempre uno de los valores: 'f' (femenino), 'm' (masculino) ó 'o' (no definido).
- VINCULAR garantiza mediante un CHECK que las dos obras introducidas en una tupla de esta tabla son siempre diferentes.

- **Tabla OBRA:**

```
CREATE TABLE obra (
   id_obra            NUMBER(12) CONSTRAINT obra_pk PRIMARY KEY,
   titulo             VARCHAR2(150) NOT NULL,
   anyo_estreno       NUMBER(5),
   periodo_emision    VARCHAR2(9),
   tipo               VARCHAR2(1)
                      CONSTRAINT tipo_chk CHECK ( tipo IN ( 'P', 'S' ) ) NOT NULL);

CREATE SEQUENCE obr_id_obra_seq START WITH 1 INCREMENT BY 1;
CREATE OR REPLACE TRIGGER obr_id_obra_trg BEFORE
   INSERT ON obra
   FOR EACH ROW
   WHEN ( new.id_obra IS NULL )
BEGIN
   :new.id_obra := obr_id_obra_seq.nextval;
END;
```

- **Tabla GENERO:**

```
CREATE TABLE genero (
   nombre  VARCHAR2(25) CONSTRAINT genero_pk PRIMARY KEY);
```

- **Tabla CLASIFICAR**:

```
CREATE TABLE clasificar (
   obra          NUMBER(12) NOT NULL,
   genero        VARCHAR(25) NOT NULL,
   CONSTRAINT clasificar_pk PRIMARY KEY (obra, genero),
   CONSTRAINT clasificar_obra_fk FOREIGN KEY (obra) REFERENCES obra(id_obra),
```

CONSTRAINT clasificar_genero_fk FOREIGN KEY (genero) REFERENCES genero(nombre));

- **Tabla PERSONA:**

```
CREATE TABLE persona (
   id_persona  NUMBER(12) CONSTRAINT persona_pk PRIMARY KEY,
   nombre      VARCHAR2(60) NOT NULL,
   nacimiento  VARCHAR2(60),
   genero      VARCHAR2(1) DEFAULT 'O' NOT NULL,
   CONSTRAINT genero_chk CHECK ( genero IN ( 'f', 'm', 'o' ) ) );

CREATE SEQUENCE per_id_persona_seq START WITH 1 INCREMENT BY 1;
CREATE OR REPLACE TRIGGER per_id_persona_trg BEFORE
   INSERT ON persona
   FOR EACH ROW
   WHEN ( new.id_persona IS NULL )
BEGIN
   :new.id_persona := per_id_persona_seq.nextval;
END;
```

- **Tabla VINCULAR:**

```
CREATE TABLE vincular (
   obra_1      NUMBER(12) NOT NULL ,
   obra_2      NUMBER(12) NOT NULL,
   vinculo     VARCHAR2(60)
CONSTRAINT vinculo_chk CHECK ( vinculo IN ( 'precuela', 'remake', 'secuela' )) NOT NULL,
CONSTRAINT vincular_pk PRIMARY KEY (obra_1, obra_2, vinculo),
CONSTRAINT vincular_obra_fk1 FOREIGN KEY ( obra_1 ) REFERENCES obra ( id_obra ),
CONSTRAINT vincular_obra_fk2 FOREIGN KEY ( obra_2 ) REFERENCES obra ( id_obra ),
CONSTRAINT vinvular_chk CHECK ( obra_1 <> obra_2 ));
```

- **Tabla PARTICIPAR:**

```
CREATE TABLE capitulo (
   id_cap      NUMBER(12) CONSTRAINT capitulo_pk PRIMARY KEY,
   n_cap       NUMBER(12) ,
   titulo_cap  VARCHAR2(150) NOT NULL,
   temporada   VARCHAR2(40),
   mi_serie    NUMBER(12) REFERENCES obra ( id_obra ) NOT NULL);

CREATE SEQUENCE cap_id_cap_seq START WITH 1 INCREMENT BY 1;
CREATE OR REPLACE TRIGGER cap_id_cap_trg BEFORE
   INSERT ON capitulo
   FOR EACH ROW
   WHEN ( new.id_cap IS NULL )
BEGIN
   :new.id_cap := cap_id_cap_seq.nextval;
END;
```

2. INTRODUCCIÓN DE DATOS Y EJECUCIÓN DE CONSULTAS

2.1. POBLAR LA BASE DE DATOS

- **Tabla PERSONA y PARTICIPAR. Estrategia Seguida.**

En los datos de partida (DATOSDB.DATOSPELICULAS) el atributo PERSON_INFO es multivaluado. Hemos decidido usar los datos de esta tabla como están, es decir no cambiarlos a una segunda tabla donde cada uno de los diferentes valores de INFO_CONTEXT, aparezca como un atributo (columna) diferente.

También hemos encontrado personas diferentes que tienen el mismo nombre. Por ejemplo 'Noriega, Eduardo', que corresponde con un actor Mexicano nacido en 1916 y también con otro actor Español nacido en 1973.

Denominamos a un nombre (campo NAME) como REPETIDO cuando para el mismo valor de NAME se encuentran valores diferentes en PERSON_INFO para el mismo INFO_CONTEXT.

Vamos a realizar el poblado de las tablas PERSONA y PARTICIPAR en dos fases. En una primera fase (A), cargamos todos los nombres (NAME en DATOSPELICULAS) no nulos y que no estén repetidos, según el criterio de repetido explicado anteriormente.

Fase B: usando herramienta externa (excel) hemos filtrado los datos del campo PERSON_INFO para que cada persona tenga un nombre diferente, como resultado creamos la tabla DATOSPELEXTRA, que contiene los datos que había en DATOSDB.DATOSPELICULAS para las personas cuyo nombre presentaba repetición.

En el poblado de la tabla PERSONA usamos una vista (NACIDOS) que colecta toda la información de lugar de nacimiento de una persona. Además nos apoyamos en una tabla auxiliar (DIRECTORIO) que contiene toda la información de PERSON_INFO con su asociación a INFO_CONTEXT para todos los distintos nombres que aparecen en DATOSPELICULAS. De la tabla DIRECTORIO se obtienen los nombres de persona que están repetidos. Hemos encontrado 37 nombres repetidos y que han sido filtrados en excel dando como resultado DATOSPELEXTRA.csv.

El fichero de excel DATOSPELEXTRA.csv es cargado en la base de datos en la tabla DATOSPELEXTRA usando sqlldr2 con el fichero de control 'datosDatospelextra.ctl'. Esta nueva tabla será usada para poblar PERSONA y PARTICIPAR. La primera tabla poblada es PERSONA, PARTICIPAR solo puede hacerse después de existir PERSONA y OBRA.

- **Tabla OBRA.**

Poblamos en dos fases. Primero cargamos los datos de películas (diferenciadas con tipo='P') y después los de series de tv (diferenciadas con tipo='S'). Usando consulta sobre DATOSPELICULAS.

- **Tabla GENERO, CLASIFICAR y CAPITULO.**

Las tres tablas se pueblas usando consulta sobre DATOSPELICULAS, solo en CAPITULO usamos una vista que proporciona el ID_OBRA de la serie con la que está relacionado el capítulo.

- **Tabla PARTICIPAR**

Seguimos la estrategia descrita en PERSONA: dos fases. Primero los nombres no repetidos y después los repetidos, en este caso obteniendo su información de DATOSPELEXTRA.

- **Tabla VINCULAR**

De los diferentes valores de la columna LINK hemos decidido:

  - 'remake of': corresponde con un remake en nuestra BD.
  - 'follows': corresponde con una secuela en nuestra BD. Donde A es secuela de B y por tanto el año de estreno de A es posterior al de B.
  - 'followed by': corresponde con una precuela en nuestra BD. Donde A es precuela de B y por tanto el año de estreno de A es posterior al de B.

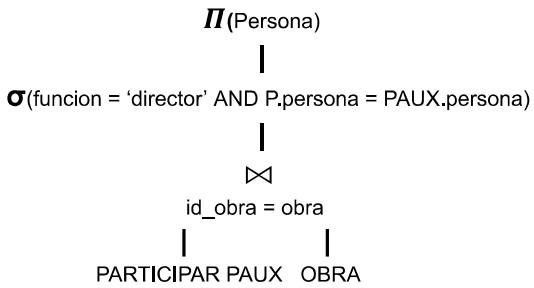
2.2. CONSULTAS SQL

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

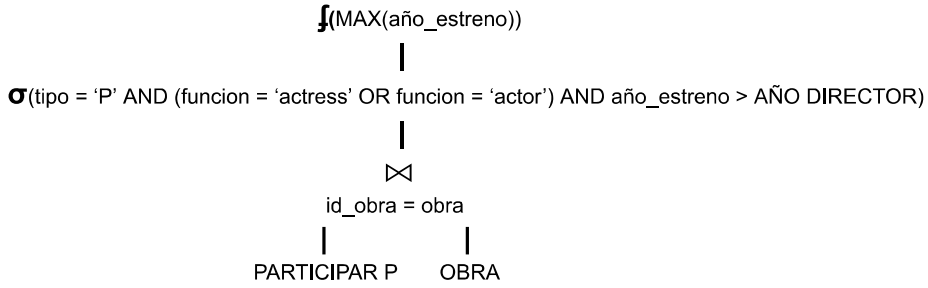
```
CREATE OR REPLACE VIEW DIRECTORES AS
SELECT distinct persona -- Id de personas que han sido directores y en la última actor/actriz
FROM PARTICIPAR P, OBRA O
WHERE O.id_obra = P.obra AND --JOIN
        O.tipo = 'P' AND (P.funcion = 'actress' OR P.funcion = 'actor')
        AND O.año_estreno > (
                SELECT max(OAux.año_estreno) -- Último año en el que una persona fue director
                FROM PARTICIPAR PAux, OBRA OAux
                WHERE PAux.obra = OAux.id_obra --JOIN
                 AND PAux.funcion = 'director' AND P.persona = PAux.persona
        );

                SELECT P.nombre -- Pone nombre a los ID's
                FROM DIRECTORES D, PERSONA P
                WHERE P.id_persona = D.persona;
```

Álgebra Relacional:

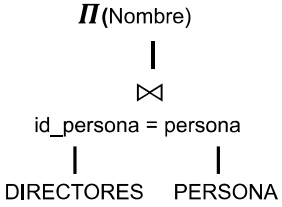
AÑO DIRECTOR:



DIRECTORES:



PRINCIPAL:



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

```
WITH PeliculasPorSaga AS (
SELECT obra_1 AS id_saga, COUNT(obra_2) AS cantidad_peliculas
FROM VINCULAR
GROUP BY obra_1
),
SagaMasLarga AS (
SELECT id_saga
FROM PeliculasPorSaga
WHERE cantidad_peliculas = (SELECT MAX(cantidad_peliculas) FROM PeliculasPorSaga)
),
PeliculasSagaMax AS (
SELECT DISTINCT obra_2 AS id_pelicula
FROM VINCULAR
WHERE obra_1 IN (SELECT id_saga FROM SagaMasLarga)
)
SELECT DISTINCT titulo
FROM OBRA
WHERE id_obra IN (SELECT id_pelicula FROM PeliculasSagaMax)
OR id_obra IN (SELECT id_saga FROM SagaMasLarga);
SELECT v.obra_1 AS id_saga, o.titulo, COUNT(v.obra_2) AS cantidad_peliculas
FROM VINCULAR v, OBRA o
where v.obra_1 = o.ID_OBRA
GROUP BY v.obra_1, o.titulo;
```

Álgebra Relacional:

PELICULASPORSAGA:

Π(obra_1 AS id_saga, obra_1$\int$COUNT(obra_2) AS cantidad_peliculas)

|

VINCULAR

SAGAMASLARGA:

Π(id_saga)

|

σ(cantidad_peliculas = $\int$MAX(cantidad_peliculas)

|

PELICULASPORSAGA

|

PELICULASPORSAGA

PELICULASSAGAMAX:

$\int$(DISTINCT(obra_2) AS id_pelicula)

|

σ(obra_1 IN (Π(id_saga))

|

SAGAMASLARGA

|

VINCULAR

TÍTULOS:

$\int$(DISTINCT(titulo))

|

σ(id_obra IN (Π(id_saga)) OR id_obra IN (Π(id_pelicula)))

| |

SAGAMASLARGA PELICULASSAGAMAX

PRINCIPAL:

Π(obra_1 AS id_saga, titulo, (obra_1, titulo)$\int$COUNT(obra_2) AS cantidad_peliculas)

|

$\bowtie$

obra_1 = id_obra

| |

VINCULAR OBRA

2.2.3. Listar las parejas actor-actriz de distinta nacionalidad que siempre han trabajado juntos en películas estrenadas 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.

```
WITH Actores AS (
   SELECT p.id_persona AS personaID, p.nombre AS
nombre, ip.pais AS paisNacimiento, o.id_obra AS obraID
   FROM participar pa
   JOIN obra o ON pa.obra = o.id_obra
   JOIN persona p ON pa.persona = p.id_persona
   JOIN iden_pais ip ON p.nacimiento = ip.nacimiento
   WHERE pa.funcion IN ('actor')
      AND o.tipo = 'P'
      AND o.anyo_estreno BETWEEN 1980 AND 2010
),
Actrices AS (
   SELECT p.id_persona AS personaID, p.nombre AS
nombre, ip.pais AS paisNacimiento, o.id_obra AS obraID
   FROM participar pa
   JOIN obra o ON pa.obra = o.id_obra
   JOIN persona p ON pa.persona = p.id_persona
   JOIN iden_pais ip ON p.nacimiento = ip.nacimiento
   WHERE pa.funcion IN ('actress')
      AND o.tipo = 'P'
      AND o.anyo_estreno BETWEEN 1980 AND 2010
)
SELECT a.nombre AS actor_nombre, a.paisNacimiento
AS actor_pais, b.nombre AS actriz_nombre,
b.paisNacimiento AS actriz_pais,
   COUNT(*) AS NumPelis,
   (SELECT COUNT(*) FROM Actores ac WHERE
ac.personaID = a.personaID) AS TotalPelisActor,
   (SELECT COUNT(*) FROM Actrices act WHERE
act.personaID = b.personaID) AS TotalPelisActriz
FROM Actores a
JOIN Actrices b ON a.obraID = b.obraID AND
a.paisNacimiento != b.paisNacimiento
GROUP BY a.nombre,  b.nombre,  a.personaID,
b.personaID, a.paisNacimiento, b.paisNacimiento
HAVING
   COUNT(*) = (SELECT COUNT(*) FROM Actores ac
WHERE ac.personaID = a.personaID)
   AND COUNT(*) = (SELECT COUNT(*) FROM Actrices
act WHERE act.personaID = b.personaID)
ORDER BY
   NumPelis DESC;
```

Álgebra Relacional:

Actores / Actrices:

Π(P.id_persona AS personaID, P.nombre AS nombre, IP.pais AS paisNacimiento, O.id_obra As obraID)

|

σ(PA.funcion = ‘actor’ / ‘actress’ AND O.tipo = ‘P’ AND O.anyo_estreno BETWEEN 1980 AND 2010)

|

$\bowtie$

PA.obra = O.id_obra

| |

$\bowtie$ OBRA

PA.persona = P.id_persona

| |

$\bowtie$ PARTICIPAR PA

P.nacimiento = ip.nacimiento

| |

PERSONA P IDEN_PAIS IP

Principal:

Π(A.nombre AS actor_nombre, A.paisNacimiento AS actor_pais, B.nombre AS actriz_nombre, B.paisNacimiento AS actriz_pais, (A.nombre, B.nombre, A.personaID, B.personaID, A.paisNacimiento, B.paisNacimiento)$\int$COUNT(*) AS NumPelis, Π($\int$COUNT(*) σ(AC.personaID = A.personaID)) AS TotalPelisActor, Π($\int$COUNT(*) σ(AC.personaID = A.personaID) AS TotalPelisActor)))

|

σ(A.paisNacimiento != B.paisNacimiento)

|

$\bowtie$

A.obraID = B.obraID

| |

ACTORES A ACTRICES B

3. DISEÑO FÍSICO

3.1. Análisis de rendimiento

- **Consulta 1:**

  Al realizar el xplain for sobre la consulta 1 se nos muestra que la mayor demora se produce al filtrar la tabla OBRA entre películas y series. Una solución posible a esto sería dividir la tabla OBRA en 2. Una de PELICULA y otra SERIE. Esto minimizaría el coste de esta consulta en concreto, pero aumentaría enormemente la complejidad de la base de datos. Habría dos relaciones PARTICIPAR, lo que se traduciría en dos tablas. Esto perjudicaría a aquellas consultas que necesitan saber la participación de una persona tanto en películas como en series debido a que habría que realizar un Join entre estas dos tablas.

- **Consulta 2:**

  Al obtener los resultados del xplain for sobre la consulta 2 vemos que lo que aumenta el tiempo de ejecución de esta es la realización de un join entre la tabla VINCULAR y OBRA, para obtener los nombres de las películas. Para evitar este join alteramos la tabla VINCULAR añadiendole los campos titulo_obra_1 y titulo_obra_2.

```
ALTER TABLE VINCULAR
ADD titulo_obra_1 VARCHAR(150)
ADD titulo_obra_2 VARCHAR(150);

UPDATE vincular v
SET (v.titulo_obra_1, v.titulo_obra_2) = (
   SELECT o1.titulo, o2.titulo
   FROM obra o1
   JOIN obra o2 ON v.obra_1 = o1.id_obra AND v.obra_2 = o2.id_obra
);
```

  El rendimiento ahora parece ser ligeramente mejor. Pero habría que añadir un trigger para que los campos de titulo fuesen correctos. Además en la consulta nueva se emplea un UNION, que podría empeorar el resultado con volúmenes de datos más grandes.

- **Consulta 3:**

3.2. Restricciones y triggers propuestos

3.2.1. Comprobar que el año de estreno de la precuela es posterior a la otra

```
CREATE OR REPLACE TRIGGER verificar_año_precuela
BEFORE INSERT ON VINCULAR
FOR EACH ROW
DECLARE
   año_precuela INTEGER;
   año_original INTEGER;
   MAL_AÑO EXCEPTION;
BEGIN
   SELECT O1.año_estreno INTO año_precuela
   FROM OBRA O1
   WHERE O1.id_obra = :new.obra_1;

   SELECT O2.año_estreno INTO año_original
   FROM OBRA O2
   WHERE O2.id_obra = :new.obra_2;

   IF año_original > año_precuela THEN
      RAISE MAL_AÑO;
   END IF;
END;
```

Si se intenta insertar una tupla en vincular de la forma (A, B, ‘precuela’). Siendo A y B id’s de obras correspondientes a la obra que es precuela y con la que tiene esta relación, respectivamente. Si sucede que el A se estrenó antes que B se lanza una excepción, ya que una precuela no se puede estrenar antes que la obra de la cual es precuela.

3.2.2. Mantener la integridad de la tabla PARTICIPAR, ya que no tiene una clave primaria.

```
CREATE OR REPLACE TRIGGER verifica_participar
BEFORE INSERT ON participar
FOR EACH ROW
DECLARE
   v_count NUMBER;
BEGIN
   SELECT COUNT(*)
   INTO v_count
   FROM participar
   WHERE obra = :NEW.obra
     AND persona = :NEW.persona
     AND funcion = :NEW.funcion
     AND ((:NEW.papel IS NULL AND descripcion = :NEW.descripcion) OR
         (:NEW.descripcion IS NULL AND papel = :NEW.papel));
   IF v_count > 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'Existe otra fila con el mismo
valor de obra, persona, y funcion. Esta nueva fila debe tener papel o
descripcion diferente.');
   END IF;
END;
```

Como no es posible garantizar que las columnas papel y descripción no sean nulas, nos imposibilita definir una clave primaria. Pretendemos impedir que la misma persona pueda tener dos veces la misma función sin que exista ninguna diferencia en los valores de papel o de descripción. Es decir una persona podrá aparecer varias veces con la misma función, pero bien es un actor que interpreta diferentes papeles, o bien desempeña cualquier otra función, pero siempre con un valor diferente en la columna descripción.

Mediante este trigger se verifica si ya existe una otra fila con exactamente los mismos valores que se pretende introducir ahora, en ese caso lanza un error.

3.2.3. Una obra puede estar vinculada con otra misma obra varias veces con un vínculo distinto.

Esta es una restricción que ni en el esquema E/R ni el modelo relacional resuelven. La tabla vincular tiene una clave primaria compuesta por las columnas obra_1 y obra_2. De esta forma una misma obra (A) no puede vincularse con otra segunda obra (B) mas que una vez. La siguiente vez que se intente relacionar, aunque sea con un vínculo diferente, se violará la clave primaria.

Aún así siempre se podría incluir en orden diferente, es decir la segunda vez obra_1=B y obra_2=A. De esta manera no se viola la clave primaria, pero es una situación que no se puede permitir. El trigger del apartado 3.2.2 también resuelve esta restricción no modelada, ya que usa las fechas de estreno y en este caso no permitirá la introducción.

3.2.4. Evitar que haya tuplas repetidas en PARTICIPAR

```
CREATE OR REPLACE TRIGGER integridad_participar
BEFORE INSERT ON PARTICIPAR
FOR EACH ROW
DECLARE
   contador INTEGER;
   EXISTE EXCEPTION;
BEGIN
   SELECT COUNT(*) INTO contador
   FROM PARTICIPAR
   WHERE :new.persona = persona AND :new.obra = obra AND
      :new.funcion = funcion AND
      ((:new.papel IS NULL AND papel IS NULL) OR :new.papel = papel) AND
      ((:new.descripcion IS NULL AND descripcion IS NULL) OR
:new.descripcion = descripcion);

   IF contador > 0 THEN
      RAISE EXISTE;
   END IF;
END;
```

Este trigger se emplea para evitar que haya tuplas repetidas en la tabla PARTICIPAR. Idealmente se habría hecho con un índice, definido en la propia creación de la tabla. Pero esto no ha sido posible ya que los atributos papel y descripción pueden ser nulos.
El proceso del trigger consiste en comprobar si ya existe esa tupla. Si esto sucede se lanza una excepción y se evita esta inserción.

ANEXO I

Apartado	Antonio	Alex
1.1	2.5	
1.2	1.5	
1.3	2	
2.1	10	
2.2	2	
3.1	0.5	
3.2	1.5	
TOTAL HORAS	20	