Pauta Ayudantía 6

IIC2413, Bases de Datos · Benjamín Saavedra

Plataforma de streaming

Una plataforma de streaming quiere diseñar su base de datos. Se sabe lo siguiente:

- Cada **contenido** tiene un *id* único, un título y un año. Todo contenido es **exactamente una** de dos cosas: una **película** (que además tiene duración) o una **serie**.
- Cada contenido es producido por **exactamente un estudio** (nombre único, país). Un estudio puede producir muchos contenidos.
- Una película puede ser **remake** de otra película (su original). Una película original puede tener varios remakes.
- Una serie tiene **temporadas** numeradas 1, 2, 3, ... *dentro de cada serie* (la temporada 1 de dos series distintas son cosas distintas). Cada temporada tiene un año de estreno.
- Cada temporada tiene **episodios** numerados *dentro de su temporada*, con título y duración.
- Los **usuarios** (email único, nombre) tienen **perfiles**. El nombre de un perfil es único *sólo dentro de su cuenta* (dos usuarios pueden tener un perfil llamado "Niños").
- Se quiere registrar cada **visualización**: un perfil puede ver la misma película o el mismo episodio *muchas veces*. En cada visualización se guarda la fecha-hora de inicio y el minuto en que quedó. Un perfil no inicia dos visualizaciones en el mismo instante.
- Un perfil puede **calificar** un contenido (película o serie) con una nota de 1 a 5. Un perfil califica un mismo contenido *a lo más una vez* (si lo vuelve a calificar, se reemplaza la nota anterior). Se guarda la nota y la fecha en que se calificó.

(a) Dibuje un diagrama E/R para el dominio. Indique llaves, tipo de cada relación y referencias necesarias.

(b) Identifique las entidades débiles. Para cada una, indique sus relaciones de apoyo y su llave completa. ¿Cuál es la llave de un episodio?

(c) Una compañera propone modelar las visualizaciones como una relación Ve entre Perfil y Episodio, con atributos fechaHora y minuto. ¿Qué problema tiene esa propuesta? Proponga una alternativa. En cambio, ¿de qué elemento del diagrama son atributos la nota y la fecha de una calificación (de Perfil, de Contenido o de la relación entre ellos), y por qué la calificación sí puede modelarse como una relación?

(d) Cada llave y cada flecha de su diagrama expresa una dependencia funcional. Escriba las dependencias funcionales que expresa su diagrama para Contenido, Pelicula, Temporada, Visualizacion y Califica (considere que cada relación varios-a-uno se absorbe en la entidad del lado "varios"). Si cada una se tradujera a una tabla, ¿quedaría en BCNF? ¿Por qué cree que ocurre esto?

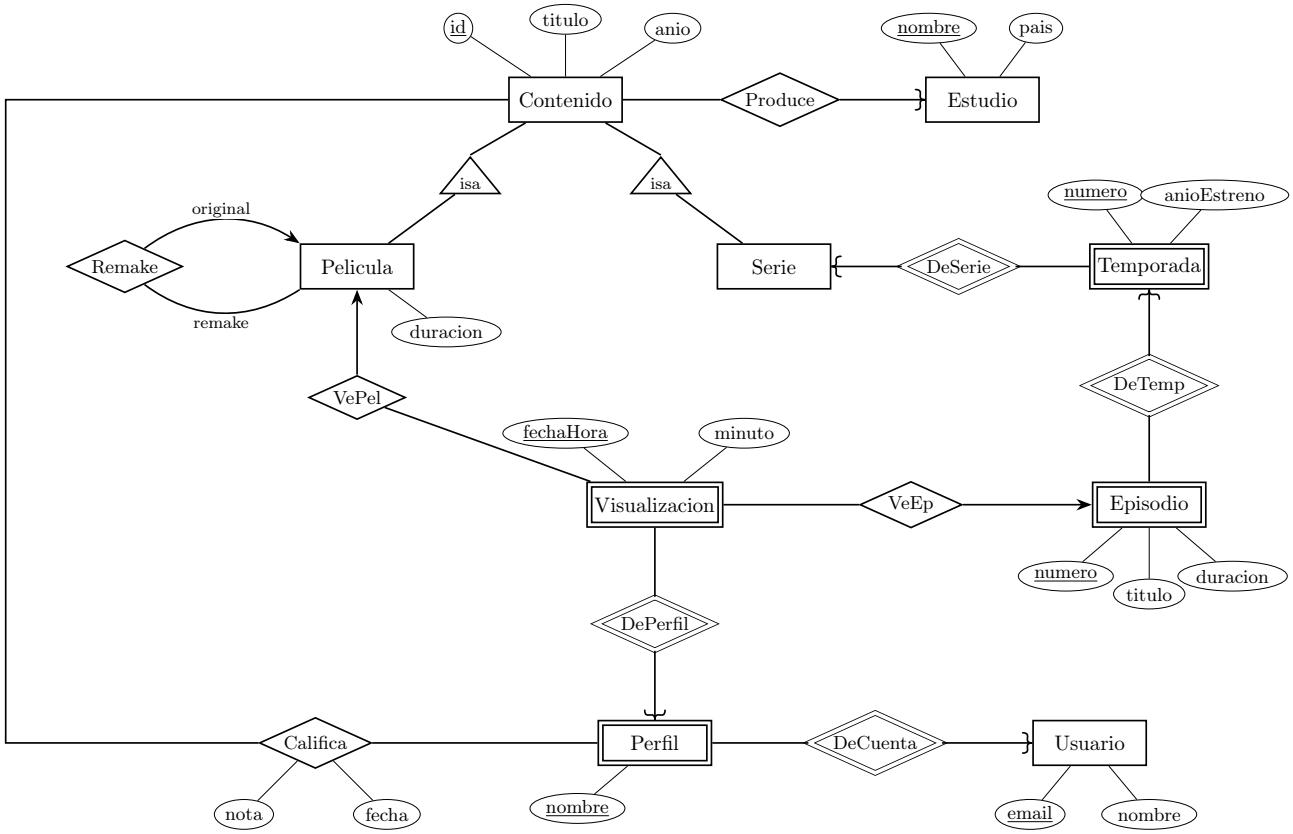
(e) Traduzca la jerarquía Película / Serie `isa` Contenido con las **tres** estrategias (E/R, OO y nulos). ¿Cuántas tablas genera cada una? ¿Cambia algo el hecho de que todo contenido sea exactamente película o serie? ¿Qué estrategia prefiere si las consultas más frecuentes son (i) "títulos de todo el contenido de un estudio" y (ii) "título y duración de las películas de 2020"?

(f) Usando la estrategia E/R para la jerarquía, escriba los `CREATE TABLE` de todo el esquema, con `PRIMARY KEY` y `FOREIGN KEY`. Elija y justifique un `ON DELETE` para cada llave foránea. En particular, ¿qué debería ocurrir con las temporadas y los episodios al borrar una serie?

(g) Un compañero propone usar `ON DELETE SET NULL` en la llave foránea de **Temporada** hacia **Serie** "para no perder las temporadas". ¿Qué ocurre? ¿Y si se usara `SET NULL` en la llave foránea de **Contenido** hacia **Estudio**?

Solución.

(a) Un diagrama posible:



- Produce: varios-a-uno de Contenido a Estudio, con referencia necesaria ("exactamente un estudio").
- Remake: relación de Pelicula consigo misma, con **roles** *remake* y *original*. Es varios-a-uno hacia el rol *original* (cada remake tiene a lo más un original). No es referencia necesaria, porque la mayoría de las películas no son remakes.
- DeSerie, DeTemp, DeCuenta, DePerfil: relaciones de apoyo (varios-a-uno con referencia necesaria).
- VePel y VeEp: varios-a-uno con flecha **normal** (no curva). Cada visualización es de una película *o* de un episodio: una visualización de película no tiene episodio, y viceversa, así que ninguna de las dos puede exigir "exactamente uno". Hacen falta dos relaciones porque Pelicula y Episodio no tienen una superclase común hacia la cual apuntar con una sola relación. La restricción "exactamente una de las dos" **no se puede dibujar** en el diagrama E/R; se anota aparte (en SQL se puede imponer con un `CHECK`).
- Califica: varios-a-varios entre Perfil y Contenido, con los atributos nota y fecha en la *relación* (ver (c)).

(b) Las entidades débiles son las siguientes:

Entidad débil	Apoyo	Llave completa
Temporada	DeSerie → Serie	(idSerie, numero)
Episodio	DeTemp → Temporada	(idSerie, numTemporada, numero)
Perfil	DeCuenta → Usuario	(email, nombre)
Visualizacion	DePerfil → Perfil	(email, nombrePerfil, fechaHora)

La llave de Episodio se obtiene *en cadena*. Su atributo numero no basta, así que toma la llave de Temporada, que a su vez es débil y toma la llave de Serie. Serie no tiene llave propia: la hereda de Contenido (id). Por eso la llave de un episodio es (id de la serie, número de temporada, número de episodio).

(c) Para ver el problema conviene recordar qué es una relación en el modelo E/R: un **conjunto** de combinaciones de entidades. En Ve(Perfil, Episodio), cada elemento es un par (perfil, episodio), y como se trata de un conjunto, **el mismo par no puede aparecer dos veces**. Pasa lo mismo que con el conjunto $\{1, 2, 2\}$, que en realidad es $\{1, 2\}$.

Esto se ve también al pasar la relación a tabla. La tabla de una relación varios-a-varios tiene como llave primaria las llaves de las entidades que participan, y los atributos de la relación (fechaHora, minuto) quedan fuera de la llave:

```
Ve(email, nombrePerfil, idSerie, numTemporada, numEpisodio, fechaHora, minuto)
PRIMARY KEY (email, nombrePerfil, idSerie, numTemporada, numEpisodio)
```

Ahora pensemos en un caso concreto. El perfil "Ana" de `ana@mail.com` ve el lunes el episodio 1 de la temporada 1 de la serie 7 y queda en el minuto 20; el martes lo vuelve a ver y lo termina:

email	nombrePerfil	idSerie	numTemp	numEp	fechaHora	minuto
ana@mail.com	Ana	7	1	1	lunes 21:00	20
ana@mail.com	Ana	7	1	1	martes 22:00	45

Las dos filas tienen **la misma llave primaria**, así que el motor rechaza la segunda. Con esta propuesta sólo se podría guardar *una* visualización por cada par (perfil, episodio), cuando el enunciado dice que un perfil puede ver lo mismo muchas veces. Lo mismo pasaría con una relación análoga entre Perfil y Pelicula para registrar las películas vistas: tendría el mismo problema.

El fondo del problema es que la fecha, justo lo que distingue una visualización de otra, no forma parte de su identidad. La solución es convertir la visualización en una **entidad**, de modo que fechaHora sí pueda ser parte de la llave. Por sí sola no basta, porque dos perfiles distintos pueden empezar a ver algo a la misma hora, así que la llave se completa con el perfil. Por eso Visualizacion queda como entidad **débil**, con relación de apoyo hacia Perfil y llave (email, nombrePerfil, fechaHora). Con esa llave, las dos filas de Ana son distintas (lunes $\neq$ martes) y ambas caben. Qué se vio se conecta aparte, con VePel hacia Pelicula y VeEp hacia Episodio. Otra alternativa válida, aunque más artificial, es agregar una entidad Instante y usar una relación ternaria Ve(Perfil, Episodio, Instante).

La calificación. Para decidir dónde van la nota y la fecha, hay que preguntarse si el dato describe a una entidad *por sí sola* o a la *combinación* de entidades. Un test útil: fijar una entidad, cambiar la otra, y ver si el valor puede cambiar.

- Se fija el perfil y se cambia el contenido: el mismo perfil le pone 5 a una película y 2 a otra. La nota **no** es un atributo de Perfil (un perfil no tiene "una" nota).
- Se fija el contenido y se cambia el perfil: la misma película recibe 5 de un perfil y 3 de otro. La nota **no** es un atributo de Contenido.
- La nota depende del **par** (perfil, contenido): es un **atributo de la relación** Califica. Con la fecha pasa lo mismo.

Ojo: la "nota promedio de un contenido" sí describe al contenido, pero es un dato *derivado* (se calcula con `AVG` sobre **Califica**). Guardarlo aparte sería redundante.

¿Por qué relación y no entidad? Una relación es un *conjunto*: cada par (perfil, contenido) aparece a lo más una vez. Eso calza con la calificación, porque un perfil califica un contenido a lo más una vez (volver a calificar reemplaza la nota). La visualización, en cambio, puede repetirse para el mismo par, y por eso tuvo que ser una entidad. Si el enunciado pidiera guardar el *historial* de todas las calificaciones, la calificación también tendría que ser una entidad débil, identificada por el perfil, el contenido y la fecha.

(d) Las dependencias salen directamente del diagrama: la llave de cada entidad determina sus atributos, y cada relación varios-a-uno absorbida agrega al lado derecho la llave de la entidad apuntada.

- Contenido: $\mathsf{id} \to \mathsf{titulo}, \mathsf{anio}, \mathsf{estudio}$. La parte $\mathsf{id} \to \mathsf{estudio}$ viene de la flecha de Produce.
- Pelicula: $\mathsf{id} \to \mathsf{duracion}, \mathsf{idOriginal}$. La parte $\mathsf{id} \to \mathsf{idOriginal}$ viene de la flecha de Remake.
- Temporada: $\mathsf{idSerie}, \mathsf{numero} \to \mathsf{anioEstreno}$ (su llave incluye la de la serie, porque es débil).
- Visualizacion:

  $$\mathsf{email}, \mathsf{nombrePerfil}, \mathsf{fechaHora} \to \mathsf{minuto}, \mathsf{idPelicula}, \mathsf{idSerie}, \mathsf{numTemporada}, \mathsf{numEpisodio}.$$

- Califica: $\mathsf{email}, \mathsf{nombrePerfil}, \mathsf{idContenido} \to \mathsf{nota}, \mathsf{fecha}$ (los atributos de la relación dependen del par completo).

En todas, el único lado izquierdo es la llave de la tabla, así que **todas quedarían en BCNF**. No es casualidad: al traducir un diagrama E/R bien hecho, cada tabla describe *una sola cosa* (una entidad o una relación), y las dependencias que el diagrama puede expresar tienen siempre como lado izquierdo la llave de esa cosa. En otras palabras, un buen diseño E/R ya hace el trabajo de la normalización.

Hay dos matices. Primero, esto vale para las dependencias *que el diagrama dibuja*. Si el dominio tiene otras (por ejemplo, "no hay dos contenidos con el mismo título y año", es decir $\mathsf{titulo}, \mathsf{anio} \to \mathsf{id}$), hay que agregarlas y revisar de nuevo. En ese caso aparece una segunda llave ($\mathsf{titulo}, \mathsf{anio}$) y la tabla sigue en BCNF; en SQL se declara con `UNIQUE (titulo, anio)`. Segundo, vale sólo si la traducción es correcta: si alguien juntara, por ejemplo, visualizaciones y episodios en una sola tabla, aparecería $\mathsf{idSerie}, \mathsf{numTemporada}, \mathsf{numEpisodio} \to \mathsf{titulo}$, cuyo lado izquierdo no es llave, y la tabla dejaría de estar en BCNF.

(e) Atributos: Contenido(*id*, *titulo*, *anio*) más la llave del estudio absorbida (estudio); Pelicula agrega duracion y el idOriginal de la relación Remake absorbida; Serie no agrega atributos.

- **Estilo E/R (3 tablas)**: **Contenido**(id, titulo, anio, estudio), **Pelicula**(id, duracion, idOriginal), **Serie**(id).
- **Estilo OO**: en general, una tabla por cada subconjunto de {Pelicula, Serie} junto a la raíz, es decir, $2^2 = 4$ tablas: sólo-Contenido, Contenido+Película, Contenido+Serie y Contenido+Película+Serie. Pero en este dominio todo contenido es *exactamente* una de las dos: nunca hay un contenido que sea sólo Contenido, ni uno que sea película y serie a la vez. Esas dos tablas estarían siempre vacías y basta con **2 tablas**: **Pelicula**(id, titulo, anio, estudio, duracion, idOriginal), **Serie**(id, titulo, anio, estudio).
- **Con nulos (1 tabla)**: **Contenido**(id, titulo, anio, estudio, duracion, idOriginal). Las series tienen duracion e idOriginal en `NULL`. *Ojo*: si una película puede tener duración desconocida, ya no se distingue de una serie; en ese caso conviene agregar un atributo discriminador tipo.

Con respecto a las consultas:

	E/R	OO	Nulos
(i) títulos de un estudio	1 tabla	`UNION` de 2 tablas	1 tabla
(ii) título y duración de películas 2020	`JOIN` de 2 tablas	1 tabla	1 tabla (+ filtro de tipo)

La estrategia con nulos responde ambas consultas con una sola tabla, a cambio de espacio desperdiciado. También **pierde integridad**: por ejemplo, **Temporada** tendría que referenciar a **Contenido** y no habría forma de impedir con una llave foránea que una película tenga temporadas. En OO, en cambio, no se puede garantizar con una `PRIMARY KEY` que un id no se repita entre **Pelicula** y **Serie**. La estrategia E/R es la más "limpia" en integridad y la que usaremos en (f).

(f) Un esquema posible:

```
CREATE TABLE Estudio ( nombre VARCHAR(50), pais VARCHAR(50),
    PRIMARY KEY (nombre) );

CREATE TABLE Contenido ( id INT, titulo VARCHAR(100), anio INT,
    estudio VARCHAR(50) NOT NULL,                -- referencia necesaria
    PRIMARY KEY (id),
    FOREIGN KEY (estudio) REFERENCES Estudio(nombre) ON DELETE NO ACTION );

CREATE TABLE Pelicula ( id INT, duracion INT, idOriginal INT,
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES Contenido(id) ON DELETE CASCADE,
    FOREIGN KEY (idOriginal) REFERENCES Pelicula(id) ON DELETE SET NULL );

CREATE TABLE Serie ( id INT,
    PRIMARY KEY (id),
    FOREIGN KEY (id) REFERENCES Contenido(id) ON DELETE CASCADE );

CREATE TABLE Temporada ( idSerie INT, numero INT, anioEstreno INT,
    PRIMARY KEY (idSerie, numero),
    FOREIGN KEY (idSerie) REFERENCES Serie(id) ON DELETE CASCADE );

CREATE TABLE Episodio ( idSerie INT, numTemporada INT, numero INT,
    titulo VARCHAR(100), duracion INT,
    PRIMARY KEY (idSerie, numTemporada, numero),
    FOREIGN KEY (idSerie, numTemporada)
        REFERENCES Temporada(idSerie, numero) ON DELETE CASCADE );

CREATE TABLE Usuario ( email VARCHAR(100), nombre VARCHAR(50),
    PRIMARY KEY (email) );

CREATE TABLE Perfil ( email VARCHAR(100), nombre VARCHAR(30),
    PRIMARY KEY (email, nombre),
    FOREIGN KEY (email) REFERENCES Usuario(email) ON DELETE CASCADE );

CREATE TABLE Visualizacion ( email VARCHAR(100), nombrePerfil VARCHAR(30),
    fechaHora TIMESTAMP, minuto INT,
    idPelicula INT,                              -- VePel (absorbida)
    idSerie INT, numTemporada INT, numEpisodio INT, -- VeEp (absorbida)
    PRIMARY KEY (email, nombrePerfil, fechaHora),
    FOREIGN KEY (email, nombrePerfil)
        REFERENCES Perfil(email, nombre) ON DELETE CASCADE,
    FOREIGN KEY (idPelicula) REFERENCES Pelicula(id) ON DELETE CASCADE,
    FOREIGN KEY (idSerie, numTemporada, numEpisodio)
        REFERENCES Episodio(idSerie, numTemporada, numero) ON DELETE CASCADE );

CREATE TABLE Califica ( email VARCHAR(100), nombrePerfil VARCHAR(30),
    idContenido INT,
    nota INT, fecha DATE,                        -- atributos de la relación
    PRIMARY KEY (email, nombrePerfil, idContenido),
    FOREIGN KEY (email, nombrePerfil)
        REFERENCES Perfil(email, nombre) ON DELETE CASCADE,
    FOREIGN KEY (idContenido) REFERENCES Contenido(id) ON DELETE CASCADE );
```

Justificación de cada decisión:

- **Subclases** (**Pelicula**, **Serie**): su id es a la vez `PRIMARY KEY` y llave foránea hacia **Contenido**. Si se borra el contenido, también debe desaparecer su parte de subclase, así que corresponde `CASCADE`.
- **Entidades débiles** (**Temporada**, **Episodio**, **Perfil**, **Visualizacion**): no existen sin su entidad dueña, así que corresponde `CASCADE`. Al borrar una serie se borran en cadena sus temporadas y, por la llave foránea de **Episodio**, también sus episodios (y las visualizaciones de esos episodios).
- **Remake**: si se borra la película original, el remake sigue existiendo; sólo se pierde el vínculo, así que corresponde `SET NULL`. Se puede porque idOriginal no es parte de la llave y admite nulos.
- **Estudio**: `NO ACTION`. No queremos que al borrar un estudio desaparezca silenciosamente todo su catálogo, y `SET NULL` es imposible (ver (g)).
- **Visualización** $\rightarrow$ **contenido**: aquí se puede discutir. `CASCADE` borra el historial del contenido eliminado; `NO ACTION` impide borrar contenido que alguien vio. Ambas se pueden justificar.
- La restricción "exactamente una de idPelicula o idSerie no es nula" no se captura con PK/FK. Se puede agregar (fuera de lo visto en clases) con un `CHECK ((idPelicula IS NULL) <> (idSerie IS NULL))`.
- **Califica**: es varios-a-varios, así que tiene tabla propia. Su llave son las llaves de **Perfil** y **Contenido** (un perfil califica un contenido a lo más una vez); nota y fecha *no* son parte de la llave. `CASCADE` en ambas: una calificación no tiene sentido sin su perfil ni sin su contenido.

(g) Recordemos primero qué hace `ON DELETE SET NULL`: cuando se borra una fila referenciada (por ejemplo, una serie), el motor busca las filas que la apuntan con la llave foránea (sus temporadas) y, en vez de borrarlas, **pone en `NULL` la columna de la llave foránea**. La tabla de temporadas, con la propuesta del compañero, sería:

```
Temporada(idSerie, numero, anioEstreno)
PRIMARY KEY (idSerie, numero)
FOREIGN KEY (idSerie) REFERENCES Serie(id) ON DELETE SET NULL    -- la propuesta
```

Supongamos que la serie 7 tiene dos temporadas y ejecutamos `DELETE FROM Serie WHERE id = 7`. El motor intentaría dejar la tabla así:

idSerie	numero	anioEstreno
~~7~~ → `NULL`	1	2020
~~7~~ → `NULL`	2	2021

Pero idSerie es **parte de la llave primaria**, y una llave primaria **nunca puede tener** `NULL`. Como la operación violaría esa regla, el motor la rechaza y **el `DELETE` completo falla**: no se borra la serie ni se modifica ninguna temporada. El compañero quería "no perder las temporadas", pero lo que consigue es que la serie no se pueda borrar; en la práctica, se comporta igual que `NO ACTION`.

Y aunque el motor lo permitiera, tampoco tendría sentido en el dominio. Una fila $(\texttt{NULL}, 1, 2020)$ sería una "temporada 1 de ninguna serie", y podría haber varias iguales, una por cada serie borrada. Temporada es débil justamente porque su número *sólo identifica algo junto con su serie*: sin la serie, la temporada pierde su identidad. Por eso lo correcto es `ON DELETE CASCADE`: si la serie se va, sus temporadas (y, en cadena, sus episodios) también.

Con **Contenido** → **Estudio** ocurre lo mismo, pero por otra razón. La columna estudio *no* es parte de la llave, pero la declaramos `NOT NULL`, porque el enunciado dice que todo contenido tiene *exactamente un* estudio (flecha curva). Al borrar un estudio, `SET NULL` intentaría poner `NULL` en una columna `NOT NULL`, y el `DELETE` falla igual.

En resumen, `SET NULL` sólo funciona si la columna **admite nulos**, es decir, si no es parte de la llave primaria ni es `NOT NULL`. Por eso nunca sirve para la llave foránea de una **entidad débil** hacia su dueño (queda dentro de la PK) ni para una **referencia necesaria** (flecha curva, `NOT NULL`). Donde sí tiene sentido es en **Pelicula**.idOriginal: si se borra la película original, el remake sigue existiendo y simplemente queda "sin original".