TEMARIO

MODELADO DE DATOS UTILIZANDO EL MODELO RELACIONAL

Tal como dice el título de este capítulo, la idea fundamental es explicarte el proceso para crea un modelo de datos relacional. Específicamente utilizaremos las entidades y relaciones del modelo Entidad-Relación para crear posteriormente el modelo Relacional, mediante un conjunto de reglas de transformación.

En primer lugar te explicaremos un conjunto de conceptos técnicos que debes comprender. En segundo lugar describiremos las características principales del modelo Relacional para finalizar con un conjunto de directrices que te ayudarán a crear un modelo de datos relacional correcto.

Este capítulo dispone de un conjunto de ejercicios resueltos y otros propuestos para que practiques.

3.1 Introducción

Edgar Frank Codd definió las bases del modelo relacional a fines de los años 60. En el año 1970 publica el artículo "A Relational Model of data for Large Shared Data Banks" (que en Español es "Un modelo relacional de datos para grandes bancos de datos compartidos"). Actualmente se considera que ese es uno de los documentos más influyentes de toda la historia de la informática. Lo es porque en él se definieron las bases del llamado Modelo Relacional de Bases de Datos. Anteriormente el único modelo teórico estandarizado era el Codasyl que se utilizó masivamente en los años 70 como paradigma del modelo en red de bases de datos.

Codd se apoya en los trabajos de los matemáticos Cantor y Childs (cuya teoría de conjuntos es la verdadera base del modelo relacional, y es por ello el nombre debido al álgebra relacional). Según Codd los datos se agrupan en relaciones (actualmente llamadas tablas) que es un concepto que se refiere a la estructura que aglutina datos referidos a una misma entidad de forma independiente respecto a su almacenamiento físico.

Lo que Codd intentaba fundamentalmente es evitar que los usuarios de la base de datos tuvieran que verse obligados a aprender la estructura interna del sistema. Pretendía que los usuarios trabajaran de forma sencilla e independiente del funcionamiento físico de la base de datos en sí. Fue un enfoque revolucionario.

Aunque trabajaba para IBM, esta empresa no recibió de buen grado sus teorías (de hecho continuó trabajando en su modelo en red IMS). De hecho fueron otras empresas (en especial Oracle) las que implementaron sus teorías. Pocos años después el modelo se empezó a utilizar cada vez más, hasta finalmente ser el modelo de bases de datos más popular. Hoy en día casi todas las bases de datos siguen este modelo.

El modelo propuesto por Codd presentaba los siguientes objetivos:

- **Independencia física**. El modo en que se almacenan los datos no debe influir en su manipulación lógica y, por tanto, los usuarios que acceden a los datos no han de modificar sus programas (o sistemas) por cambios en el almacenamiento físico.
- **Independencia lógica**. Añadir, eliminar o modificar cualquier elemento de la base de datos no debe repercutir en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos.
- **Flexibilidad**. Poder ofrecer a cada usuario los datos de la forma más adecuada a su aplicación.
- **Uniformidad**. Las estructuras lógicas de los datos presentan un aspecto uniforme (tablas), lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
- **Sencillez**. Las características anteriores, así como unos lenguajes de usuario sencillos, producen el resultado de que el modelo de datos relacional es fácil de comprender y de utilizar.

3.2 Estructura de las Bases de Datos Relacionales

Según el modelo relacional (desde que Codd lo enunció) el elemento fundamental es lo que se conoce como relación, aunque más habitualmente se le llama tabla (o también array o matriz). Codd definió las relaciones utilizando un lenguaje matemático, pero se pueden asociar a la idea de tabla (de filas y columnas) ya que es más fácil de entender.
No hay que confundir la idea de relación según el modelo de Codd, con lo que significa una relación en el modelo Entidad/Relación de Chen. No tienen nada que ver.

3.2.1 Conceptos Básicos

Las relaciones constan de:

Atributos. Referido a cada propiedad de los datos que se almacenan en la relación (nombre, dni,...).

Tuplas. Referido a cada elemento de la relación. Por ejemplo si una relación almacena personas, una tupla representaría a una persona en concreto.

Puesto que una relación se representa como una tabla; podemos entender que las columnas de la tabla son los atributos; y las filas, las tuplas. La figura 1 representa la estructura de una relación según el modelo de Codd..

atributo 1	atributo 2	atributo 3	....	atributo n	
valor 1,1	valor 1,2	valor 1,3	....	valor 1,n	← tupla 1
valor 2,1	valor 2,2	valor 2,3	....	valor 2,n	← tupla 2
.....	.....	.....	....	.....	....
valor m,1	valor m,2	valor m,3	....	valor m,n	← tupla m

Figura 1. Representación de una relación en forma de tabla

En la figura podemos distinguir su nombre, un conjunto de columnas, denominadas atributos, que representan propiedades de la tabla y que también están caracterizadas por su nombre, y un conjunto de filas llamadas tuplas, que contienen los valores que toma cada uno de los atributos para cada elemento de la relación. La figura 2 presenta un ejemplo de relación.

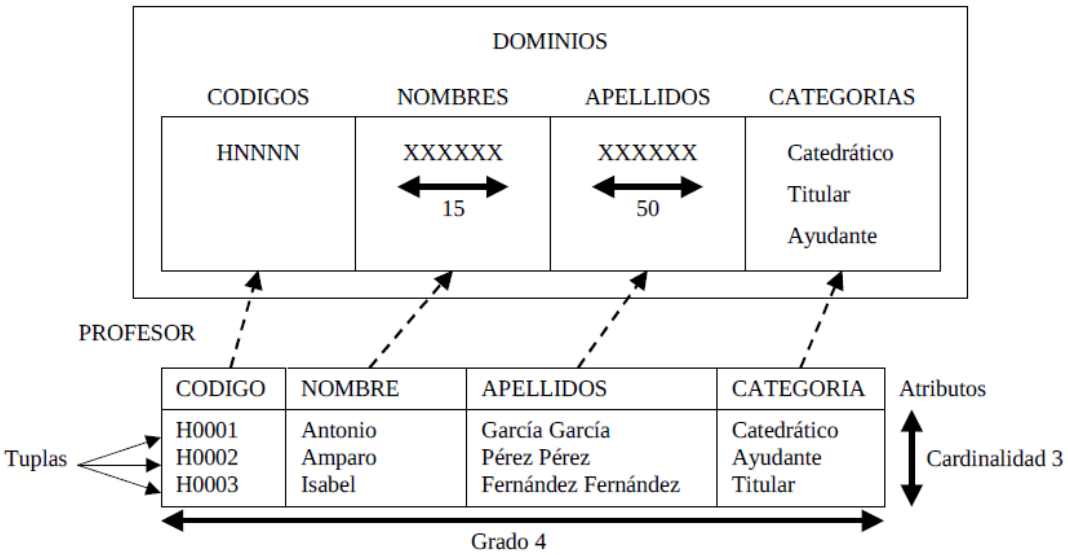

Figura 2. Representación de la relación Profesor.

La figura 2 representa la relación PROFESOR, donde aparece la estructura del modelo relacional. En ella se puede observar el nombre de la relación (PROFESOR); los atributos (código, nombre, apellidos, categoría); los dominios (de donde los atributos toman sus valores, permitiéndose que varios atributos pueden tomar valores del mismo dominio); las tuplas (cada una de las cuales contiene los valores que toma el código, nombre, apellidos, etc.); el grado (numero de atributos) y la cardinalidad (numero de tuplas).

Cada una de las filas de la relación, se corresponde con la idea clásica de registro. Representa por tanto cada elemento individual de esa relación. Tiene que cumplir que:

- Cada tupla se debe corresponder con un elemento del mundo real.
- No puede haber dos tuplas iguales (con todos los valores iguales).

Dominio: Conviene aclarar en este punto el concepto de dominio expuesto en el párrafo anterior. Un dominio es un conjunto finito de valores homogéneos y atómicos caracterizado por un nombre. Contiene todos los posibles valores que puede tomar un determinado atributo. Dos atributos distintos pueden tener el mismo dominio.

Los valores son homogéneos porque son todos del mismo tipo y atómicos porque son indivisibles en el modelo, pues si se descomponen pierden la semántica asociada a los mismos. Así, si hablamos del dominio de nacionalidades, la nacionalidad "chilena" es atómica, pues si se descompone en sus letras pierde el sentido semántico, ya que las letras aisladas pierden el significado de una nacionalidad.

Un dominio en realidad es un conjunto finito de valores del mismo tipo. A los dominios se les asigna un nombre y así podemos referirnos a ese nombre en más de un atributo.

La forma de indicar el contenido de un dominio se puede hacer utilizando dos posibles técnicas:

- **Intensión**. Se define el nomino indicando la definición exacta de sus posibles valores. Por intensión se puede definir el dominio de edades de los trabajadores como: números enteros entre el 16 y el 65 (un trabajador sólo podría tener una edad entre 16 y 65 años).
- **Extensión**. Se indican algunos valores y se sobreentiende el resto gracias a que se autodefinen con los anteriores. Por ejemplo el dominio localidad se podría definir por extensión así: Palencia, Valladolid, Villamuriel de Cerrato,...

Además pueden ser:

- **Generales**. Los valores están comprendidos entre un máximo y un mínimo
- **Restringidos**. Sólo pueden tomar un conjunto de valores

Además, conviene resaltar que aunque una relación se represente en forma de tabla, tiene una serie de elementos que la distinguen de una tabla, ya que no se admiten filas duplicadas, las filas y columnas no están ordenadas y es plana, es decir, en el cruce de una fila y de una columna solo puede haber un valor.

Grado: Por otro lado, el Grado indica el tamaño de una relación en base al número de columnas (atributos) de la misma. Lógicamente cuanto mayor es el grado de una relación, mayor es su complejidad al manejarla.

Cardinalidad: Y la Cardinalidad es el Número de tuplas de una relación, o número de filas de una tabla.

La representación en forma de tabla ha conducido a que los usuarios utilicen el nombre de tabla para denominar las relaciones y, como consecuencia de ello, se llame filas a las tuplas y columnas a los atributos. En la figura 3 se tiene una comparación de la terminología de relación, tabla y fichero.

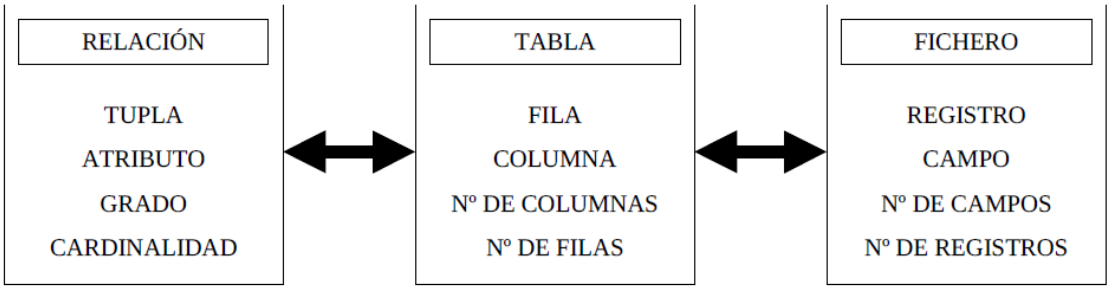

Figura 3. Comparación de la terminología de relación, tabla y archivo.

3.2.2 Claves

Hay atributos de la relación que nos servirán para identificar registros o datos externos. Es el caso de los atributos que son claves. Tenemos varias alternativas.

Clave candidata: Es el conjunto de atributos que identifican unívocamente cada tupla de la relación. Es decir columnas cuyos valores no se repiten en ninguna otra tupla de esa tabla. Toda tabla en el modelo relacional debe tener al menos una clave candidata (puede incluso haber más). Por ejemplo, en la relación EMPLEADO de la base de datos EMPRESA, podemos disponer de las siguientes claves cantidatas: RUT, EMAIL, CODIGO_EMP, ya que el Rut no se repite para ningún empleado, así también con su email y su código asignado.

Clave primaria: Clave candidata que se escoge como identificador de las tuplas. Se elige como primaria la candidata que identifique mejor a cada tupla en el contexto de la base de datos. Por ejemplo el atributo RUT sería clave candidata de una tabla de clientes, si esa tabla tiene un atributo de código de cliente, éste sería mejor candidato (y por lo tanto clave principal) porque es mejor identificador para ese contexto.

Clave alternativa: Cualquier clave candidata que no sea primaria.

Clave externa, ajena o secundaria: Son los datos de atributos de una tabla cuyos valores están relacionados con atributos de otra tabla.

Ejemplo 3.1

Por ejemplo, en la figura 4 se presentan dos relaciones (o tablas) del modelo relacional de la base de datos CAMPEONATO. La relación EQUIPO, contiene los nombres de los equipos y su número asignado, y la relación JUGADOR contiene el número del jugador, su apodo de jugador y número del equipo al que pertenece.

Para la relación EQUIPO tenemos dos claves alternativas, Equipo y nº Equipo, ya que los nombres de los equipos no se pueden repetir y su número tampoco. Por lo tanto es una forma de identificar el registro (o tupla) de manera única. En este caso la clave primaria será nº Equipo ya que es el mejor candidato.

Así también, en la relación JUGADOR, tenemos dos atributos que son claves alternativas, Nº Jugador y Jugador, ya que ambos son valores que no se repiten. En este caso se selecciona Nº Jugador como clave primaria.

Por último, el atributo Nº Equipo de la relación JUGADOR, es clave ajena o foránea, ya que es clave primaria en la relación EQUIPO. Sirve para relacionar el Jugador con el equipo al que pertenece.

EQUIPO

Equipo	Nº Equipo
Real Madrid	1
F.C. Barcelona	2
Athletic Bilbao	3

JUGADOR

Nº Jugador	Jugador	Nº Equipo
1	Karanka	3
2	Ronaldinho	2
3	Raul	1
4	Beckham	1

Figura 4. Ejemplo de clave primaria y clave ajena o foránea.

Nulos: Valor vacío. No obstante en las bases de datos se utiliza para diversos fines. En claves secundarias indican que el registro actual no está relacionado con ninguno. En otros atributos indica que la tupla en cuestión carece de dicho atributo: por ejemplo en una tabla de personas un valor nulo en el atributo teléfono indicaría que dicha persona no tiene teléfono.

Nota: Es importante indicar que el texto vacío ‘ ’, no significa lo mismo en un texto que el nulo; como tampoco el valor cero significa nulo.
Puesto que ese valor se utiliza continuamente, resulta imprescindible saber cómo actúa cuando se emplean operaciones lógicas sobre ese valor. Eso significa definir un tercer valor en la lógica booleana, además de los clásicos **verdadero** y **falso**.
Un valor nulo no es ni verdadero ni falso (se suele interpretar como un ***quizás***, o usando la aritmética clásica en valores lógicos, el 1 es verdadero, el 0 falso y el 0,5 nulo)..

3.3 Restricciones

Se trata condiciones de obligado cumplimiento por las tuplas de la base de datos. Las hay de varios tipos.

3.3.1 Inherentes

Son aquellas que no son determinadas por los usuarios, sino que son definidas por el hecho de que la base de datos sea relacional. Las más importantes son:

- No puede haber dos tuplas iguales
- El orden de las tuplas no es significativo
- El orden de los atributos no es significativo
- Cada atributo sólo puede tomar un valor en el dominio en el que está inscrito

3.3.2 Semántica

El modelo relacional permite a los usuario incorporar restricciones personales a los datos. Se comentan las diferentes reglas semánticas a continuación:

Clave principal (primary key): También llamada clave primaria. Marca uno o más atributos como identificadores de la tabla. De esa forma en esos atributos las filas de la tabla no podrán repetir valores ni tampoco dejarlos vacíos.

Unicidad (unique): Impide que los valores de los atributos marcados de esa forma, puedan repetirse. Esta restricción debe indicarse en todas las claves alternativas.
Al marcar una clave primaria se añade automáticamente sobre los atributos que forman la clave un criterio de unicidad.

Obligatoriedad (not null): Prohíbe que el atributo marcado de esta forma quede vacío (es decir impide que pueda contener el valor nulo, null).

Integridad referencial (foreign key): Sirve para indicar una clave externa (también llamada secundaria y foránea) sobre uno o más atributos. Los atributos marcados de esta forma sólo podrán contener valores que estén relacionados con la clave principal de la tabla que relacionan (llamada tabla principal). Dichos atributos sí podrán contener valores nulos. Es decir si hay una tabla de alquileres en la que cada fila es un alquiler, existirá un atributo cod_cliente que indicará el código del cliente y que estará relacionado con una tabla de clientes, en la que dicho atributo es la clave principal. De hecho no se podrá incluir un código que no esté en la tabla clientes; eso es lo que prohíbe la integridad referencial.

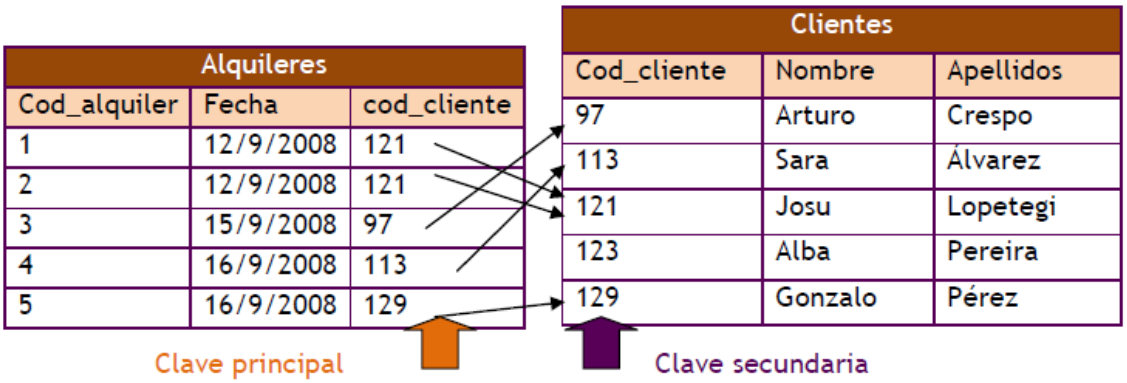

Figura 5. Ejemplo de tablas con integridad referencial.

Eso causa problemas en las operaciones de borrado y modificación de registros; ya que si se ejecutan esas operaciones sobre la tabla principal (si se modifica o borra un cliente) quedarán filas en la tabla secundaria con la clave externa haciendo referencia a un valor que ya no existe en la tabla principal.

Para solventar esta situación se puede hacer uso de estas opciones:

- Prohibir la operación (no action).
- Transmitir la operación en cascada (cascade). Es decir si se modifica o borra un cliente; también se modificarán o borrarán los alquileres relacionados con él.
- Colocar nulos (set null) Las referencias al cliente en la tabla de alquileres se colocan como nulos (es decir, alquileres sin cliente).
- Usar el valor por defecto (default). Se colocan un valor por defecto en las claves externas relacionadas. Este valor se indica al crear la tabla (opción default).

Regla de validación (check): Condición lógica que debe de cumplir un dato concreto para darlo por válido. Por ejemplo restringir el campo sueldo para que siempre sea mayor de 1000,

sería una regla de validación. También por ejemplo que la fecha de inicio sea mayor que la fecha final.

Disparadores o triggers: Se trata de pequeños programas grabados en la base de datos que se ejecutan automáticamente cuando se cumple una determinada condición. Sirven para realizar una serie de acciones cuando ocurre un determinado evento (cuando se añade una tupla, cuando se borra un dato, cuando un usuario abre una conexión…)
Los triggers permiten realizar restricciones muy potentes.

3.4 Las 12 reglas de Codd

Preocupado por los productos que decían ser sistemas gestores de bases de datos relacionales (SGBDR) sin serlo, Codd publica las 12 reglas que debe cumplir todo SGBD para ser considerado relacional. Estas reglas en la práctica las cumplen pocos sistemas relacionales. Las reglas son:

(1) Información: Toda la información de la base de datos (**metadatos**) debe estar representada explícitamente en el esquema lógico. Es decir, todos los datos están en las **tablas**.

(2) Acceso garantizado: Todo **dato** es accesible sabiendo el valor de su clave y el nombre de la columna o atributo que contiene el dato.

(3) Tratamiento sistemático de los valores nulos: El SGBD debe permitir el tratamiento adecuado de valores nulos. De ese modo el valor nulo se utiliza para representar la ausencia de información de un determinado registro en un atributo concreto.

(4) Catálogo en línea basado en el modelo relacional: Los metadatos deben de ser accesibles usando un esquema relacional. Es decir la forma de acceder a los metadatos es la misma que la de acceder a los datos.

(5) Sublenguaje de datos completo: Al menos debe de existir un lenguaje que permita el manejo completo de la base de datos. Este lenguaje, por lo tanto, debe permitir realizar cualquier operación sobre la misma.

(6) Actualización de vistas: El SGBD debe encargarse de que las vistas muestren la última información. No son válidas vistas que muestren datos que no están al día.

(7) Inserciones, modificaciones y eliminaciones de dato nivel: Cualquier operación de modificación debe actuar sobre conjuntos de filas o registros, nunca deben actuar registro a registro.

(8) Independencia física: Los datos deben de ser accesibles desde la lógica de la base de datos aún cuando se modifique el almacenamiento. La forma de acceder a los datos no varía porque el esquema físico de la base de datos, cambie.

(9) Independencia lógica: Los programas no deben verse afectados por cambios en las tablas. Que las tablas cambien no implica que cambien los programas.

(10) Independencia de integridad: Las reglas de integridad deben almacenarse en la base de datos (en el diccionario de datos), no en los programas de aplicación.

(11) Independencia de la distribución: El sublenguaje de datos debe permitir que sus instrucciones funcionen igualmente en una base de datos distribuida que en una que no lo es.

(12) No subversión: Si el SGBD posee un lenguaje procedimental que permita crear bucles de recorrido fila a fila, éste no puede utilizarse para incumplir o evitar las reglas relacionales anteriores. Especialmente la regla 7 no puede ser incumplida por ningún lenguaje del SGBD.

3.5 Paso de entidad/relación al modelo relacional

3.5.1 Transformación de las entidades fuertes

En principio las entidades fuertes del modelo Entidad Relación son transformados al modelo relacional siguiendo estas instrucciones:

- **Entidades**. Las entidades pasan a ser tablas
- **Atributos**. Los atributos pasan a ser columnas o atributos de la tabla.
- **Identificadores principales**. Pasan a ser claves primarias
- **Identificadores candidatos**. Pasan a ser claves candidatas.

Esto hace que la transformación se produzca según este ejemplo:

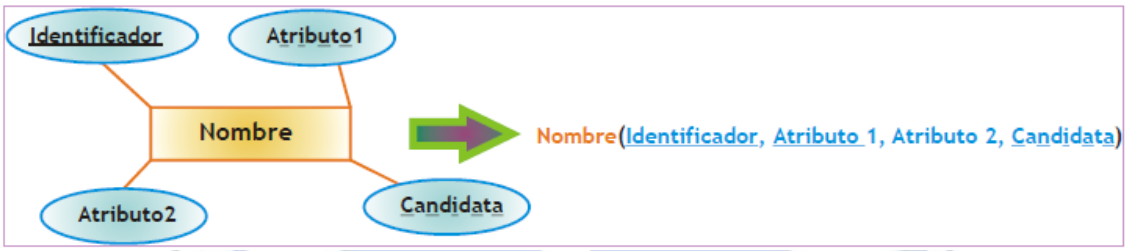

Figura 6. Ejemplo Transformación de una entidad fuerte al esquema relacional.

3.5.2 Transformación de relaciones

La idea inicial es transformar cada relación del modelo conceptual en una tabla en el modelo relacional. Pero hay que tener en cuenta todos los casos de acuerdo a la cardinalidad asignada a cada relación.

1. **Relaciones muchos a muchos:**

En las relaciones con cardinalidad muchos a muchos (N:M), la relación se transforma en una tabla cuyos atributos son: los atributos de la relación original y las claves principales de las entidades relacionadas (que pasarán a ser claves foráneas en la nueva tabla). La clave principal de la nueva tabla la forman todas las claves foráneas.

La siguiente figura muestra las tablas obtenidas desde el modelo ER para una relación con cardinalidad N:M.

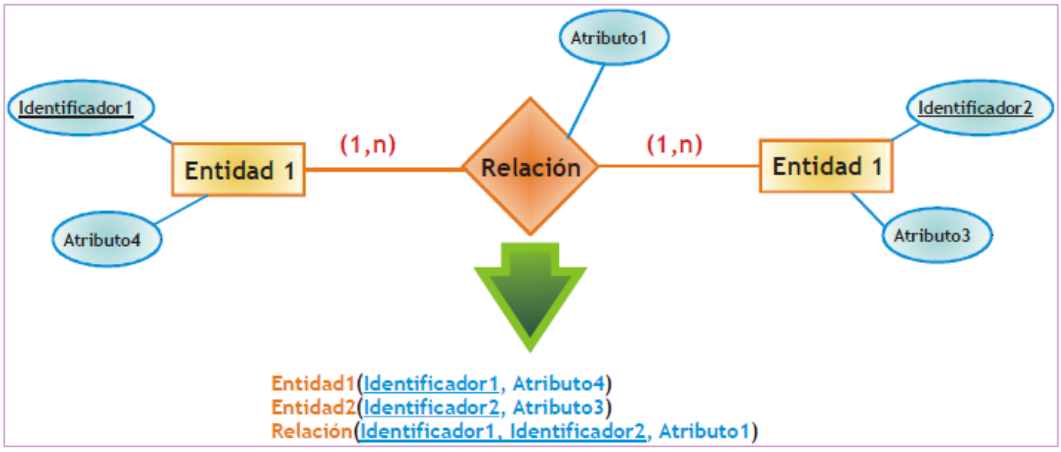

Figura 7. Ejemplo Transformación del modelo ER al modelo relacional cuando se tiene una relación con cardinalidad N:M

Ejemplo 3.2

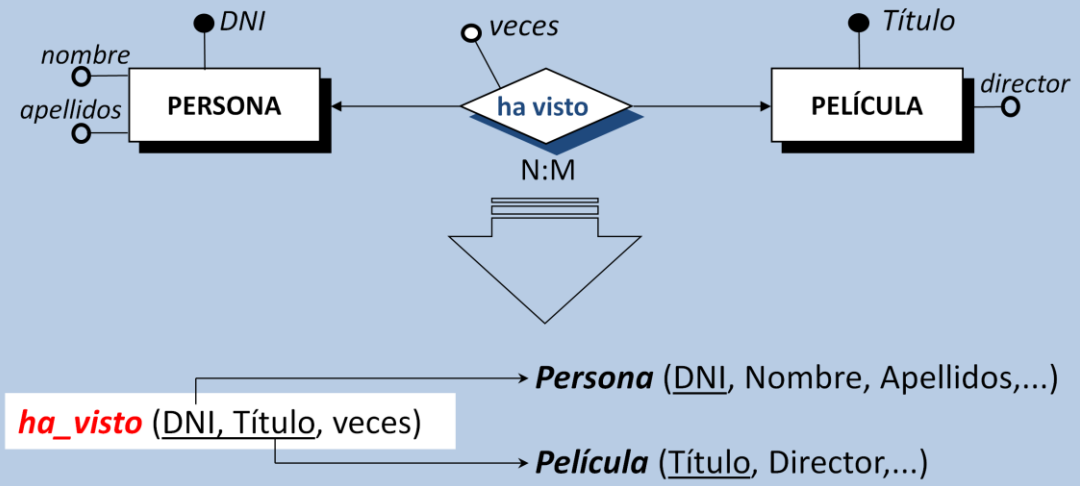

Figura 8. Ejemplo de transformación al modelo relacional cuando la cardinalidad es N:M

Observamos en la figura 8 que la relación ha_visto se transforma en tabla, conteniendo los atributos DNI, Título (claves foráneas) y veces (que proviene desde la relación original).
La clave principal estará compuesta por las claves foráneas en conjunto.

2. Relaciones de orden n:

Las relaciones ternarias, cuaternarias y n-arias que unen más de dos relaciones se transforman en una tabla que contiene los atributos de la relación más los identificadores de las entidades relacionadas. La clave la forman todas las claves externas:

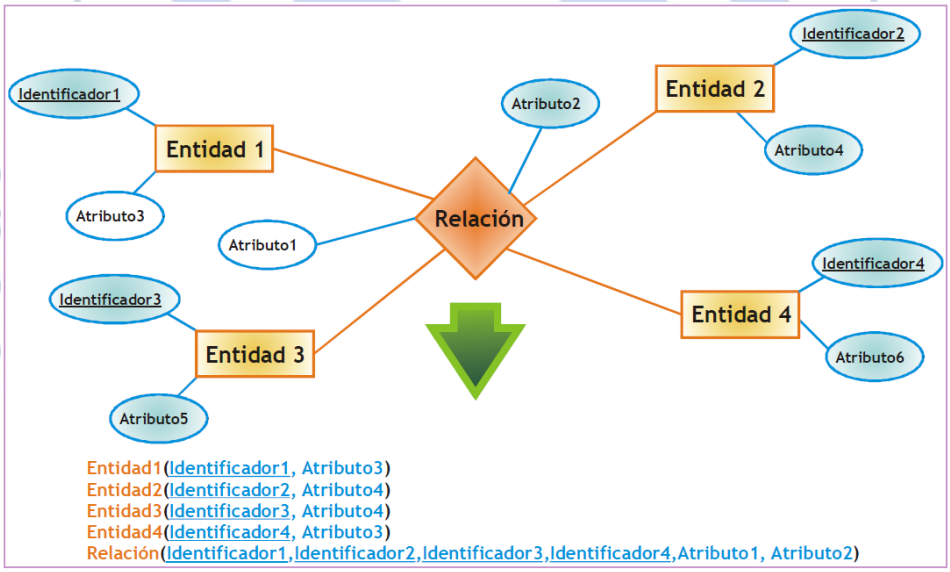

Figura 9. Ejemplo Transformación al modelo relacional cuando se tiene una relación con orden n

Ejemplo 3.3

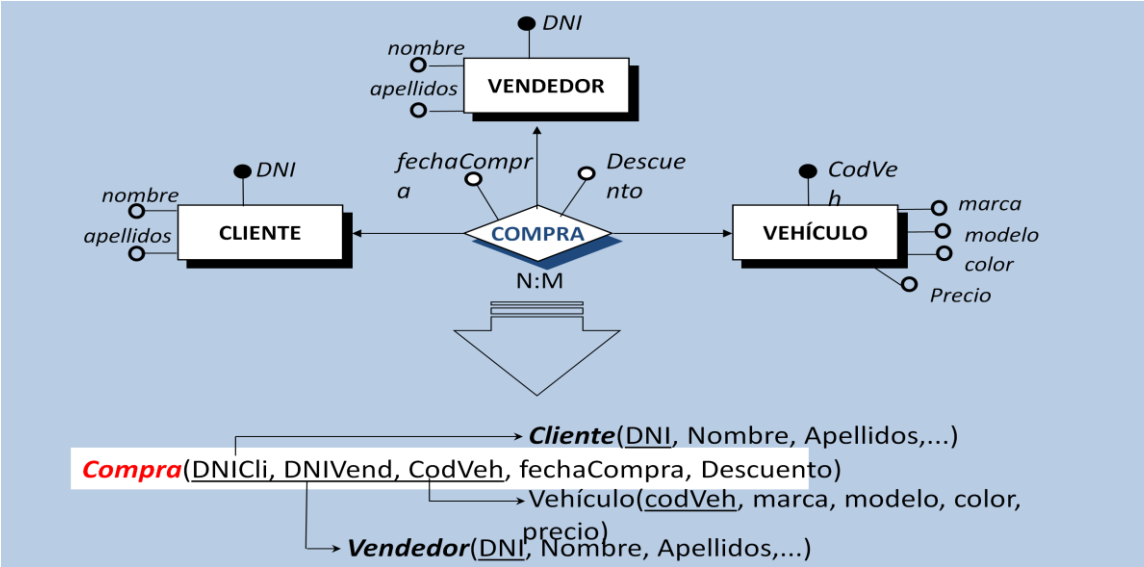

Figura 10. Ejemplo de transformación al modelo relacional cuando existe una relación ternaria.

Observamos en la figura 10 que la relación Compra se transforma en tabla, conteniendo los atributos DNICli, DNIVend, CodVeh (claves foráneas), fechaCompra y Descuento (que provienen desde la relación original).
La clave principal estará compuesta por las claves foráneas en conjunto.

3. Relaciones uno a muchos:

En este tipo de relaciones existen 2 posibilidades:

- Propagar la clave de la entidad que interviene con cardinalidad 1.

En tal caso, la tabla del lado varios (tabla relacionada) incluye como clave externa1 el identificador de la entidad del lado uno (tabla principal).

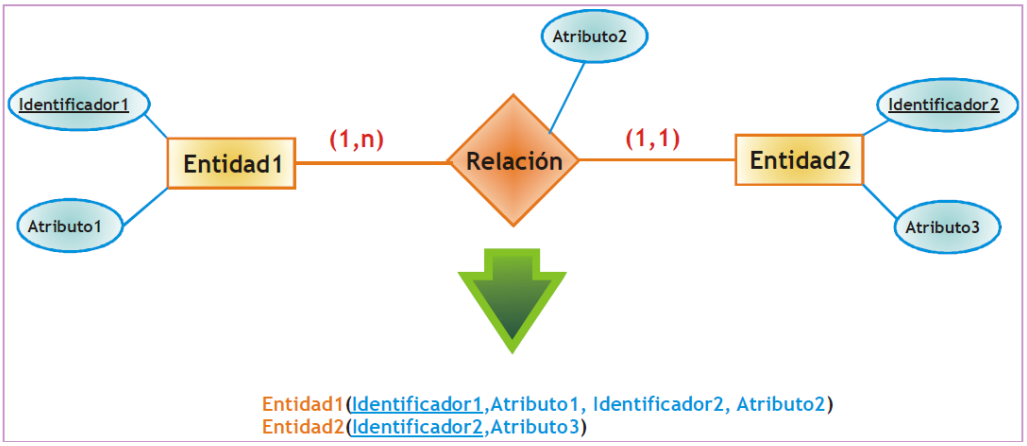

Figura 11. Ejemplo Transformación cuando propagamos la clave de la entidad que interviene con cardinalidad 1 hacia la que interviene con n

Ejemplo 3.4

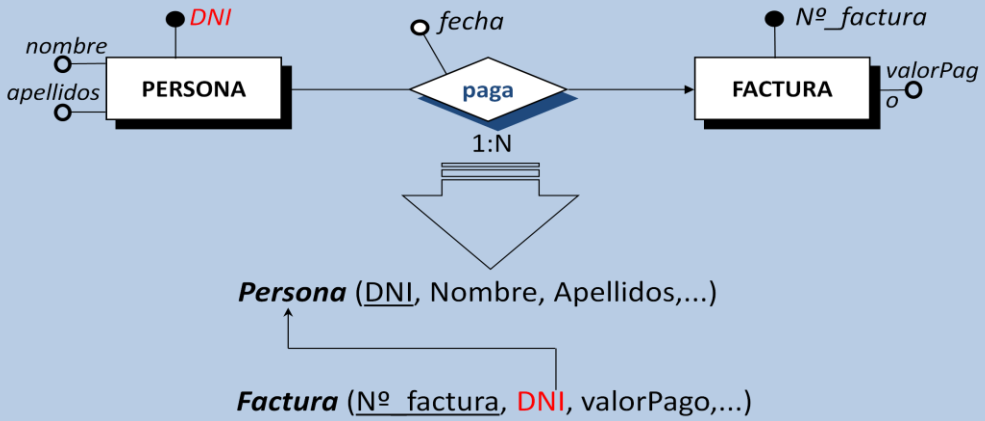

Figura 12. Ejemplo de transformación al modelo relacional cuando se propaga la clave.

Observamos en la figura 12 que la tabla Factura contiene el atributo DNI que proviene desde la tabla Persona y es por tanto clave foráneo.

- Transformar en una nueva relación.

En tal caso, se crea una nueva tabla al igual que en el caso de relaciones con cardinalidad N:M. Pero Su clave sería la clave de la entidad que interviene en la interrelación con N ocurrencias.

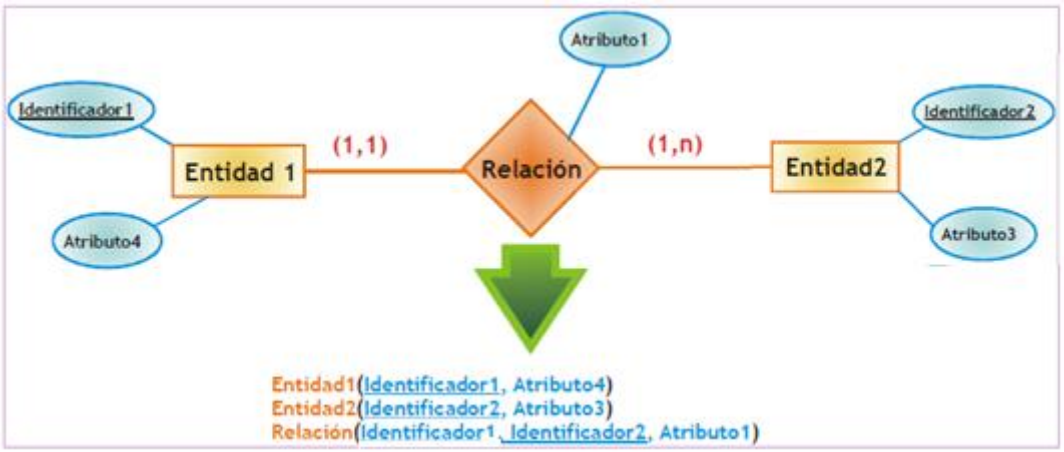

Figura 13. Ejemplo cuando creamos una tabla a partir de una relación con cardinalidad 1:N

Ejemplo 3.5

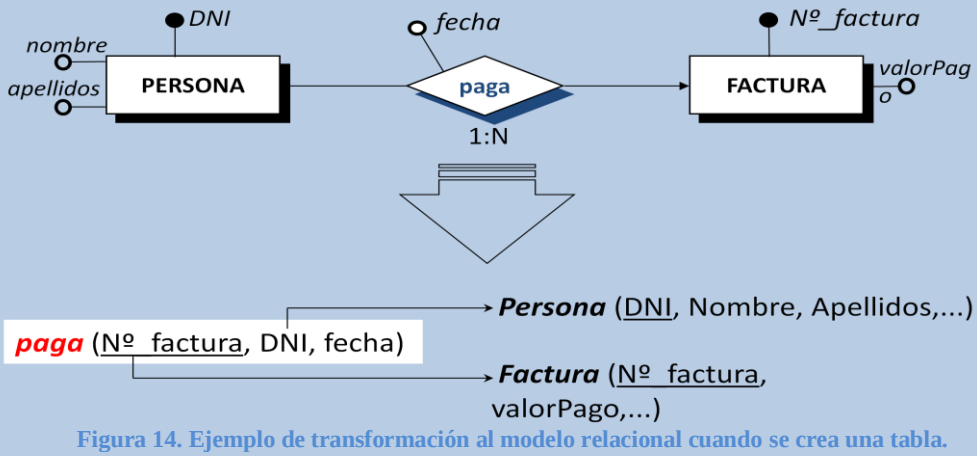

Figura 14. Ejemplo de transformación al modelo relacional cuando se crea una tabla.

Observamos en la figura 14 que la tabla Paga se crea a partir de la relación, y esta contiene el Nºfactura que proviene de la tabla Factura y participa como clave principal (también es clave foránea), además el atributo DNI que proviene desde la tabla Persona y el atributo fecha que proviene de la relación original.

¿Qué hacer con las cardinalidades mínimas cero?

Este es un caso particular que debe ser considerado a la hora de escoger propagar la clave de una de las entidades.

Revisemos la siguiente figura para aclarar los problemas que se presentan cuando no se considera el problema de mínimos cero.

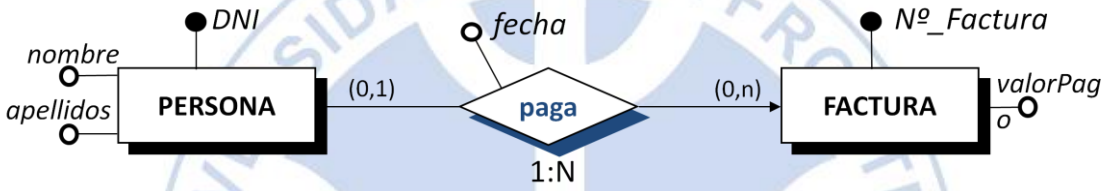

Figura 15. Ejemplo de relación con mínimos cero

- **(0,1):** la entidad que interviene con una ocurrencia es opcional. En este caso una Factura puede ser pagada por una persona o por ninguna.

Resultado:

- si se decide propagar, la clave foránea propagada será opcional.
  Al propagar, la tabla queda así Factura (NºFactura, DNI, valorPago, fecha).
  En caso que la factura no la pague nadie, el atributo DNI sería NULL o bien le asignamos un valor por defecto.
- si se decide crear una nueva tabla, habrá identificadores de la entidad opcional que no aparezcan en ninguna ocurrencia de esta relación.
  Los datos de la nueva tabla serían Paga(NºFactura, DNI, fecha). Otra vez el DNI estaría vacío ya que existe la posibilidad que nadie pague esta factura, por lo que se le debe dar el valor NULL o uno por defecto.

- **(0,n):** la entidad que interviene con varias ocurrencias es opcional. En este caso decimos que hay personas que no paga ninguna factura o bien paga varias.

Resultado: no hay que tomar ninguna medida especial

- si se propaga la clave, habrá ocurrencias de la clave de identificación propagada que no aparezcan en ninguna tupla de la otra tabla. En este caso el DNI del cliente que no tiene facturas no aparecerá en la tabla Factura.
- si se crea una nueva tabla, habrá identificadores de la entidad opcional que no aparezcan en ninguna ocurrencia de esta tabla. En este caso, el DNI de los clientes que no tienen facturas por pagar no aparecerán registrados en la tabla Paga.

¿Qué hacer con las cardinalidades mínimas uno?

Nuevamente revisemos la siguiente figura para aclarar qué ocurre en este caso.

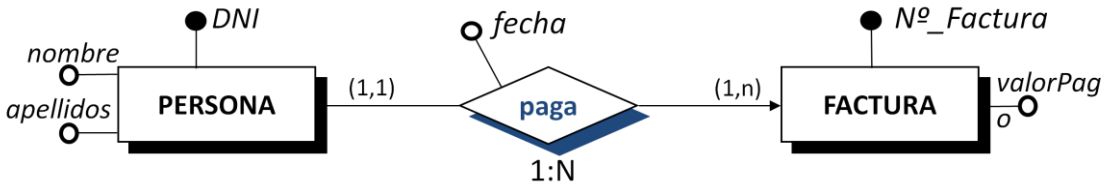

Figura 16. Ejemplo de relación con mínimos uno

- **(1,1):** la entidad interviene con una y solo una ocurrencia. Es decir, una factura es pagada por sólo una persona, y además todas se pagan.

Resultado:

- si se propaga, la clave foránea tomará siempre un valor (obligatoriedad). Esto es porque el DNI de la persona es único para una determinada factura.
- si se crea una nueva tabla, en ella debería haber una tupla por cada ocurrencia de la tabla que interviene con N. Es decir, todas las facturas aparecerán con una persona (DNI) que la paga.

- **(1,n):** la entidad interviene con una o varias ocurrencias. Esto quiere decir que una persona puede pagar mínimo una factura o muchas.

Resultado: no se toma ninguna medida especial.

- si se propaga, toda valor de la clave propagada debería aparecer en alguna ocurrencia de la otra relación. Esto se debe a que todas las personas registradas debieran tener al menos una factura vinculada.
- si se crea una nueva tabla, en ella debería haber al menos una tupla por cada ocurrencia de la relación que interviene con una ocurrencia. Esto es debido a que todas las personas pagan al menos una factura y todas las facturas son pagadas por una persna en particular.

¿Cuál de las opciones es más conveniente?

- En general, es preferible propagar la clave
- Se creará una nueva tabla si:
  a) la interrelación tiene caracterización propia (atributos propios)
  b) se prevé que la interrelación podría ser N:M en el futuro
  c) la cardinalidad mínima de la interrelación para la entidad que propaga es 0 (opcional), y en la otra entidad el número de ocurrencias no relacionadas es elevado (la clave ajena sería NULL)

4. Relaciones uno a uno:

Se puede considerar un caso particular de las anteriores, y por tanto las soluciones anteriores son válidas también en este caso.

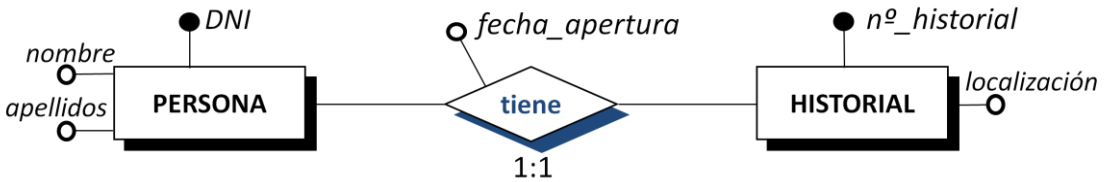

Figura 17. Ejemplo de relación con cardinalidad 1:1

Las soluciones aplicables son:

a) crear una nueva tabla
b) propagar una de las claves
c) propagar las claves de las dos entidades (mutuamente)
d) fusionar ambas entidades (interrelacionadas) en una sola tabla

a) Crear una nueva tabla

La Justificación es similar al caso 1:N
a) si las cardinalidades mínimas son cero (ambas), esto evitará valores nulos en las claves foráneas y mantendrá la simetría natural (entidades mantienen su independencia en relaciones separadas)
b) si la interrelación tiene caracterización propia (atributos) o
c) si se prevé que posteriormente puedan variarse las cardinalidades

Ejemplo 3.6

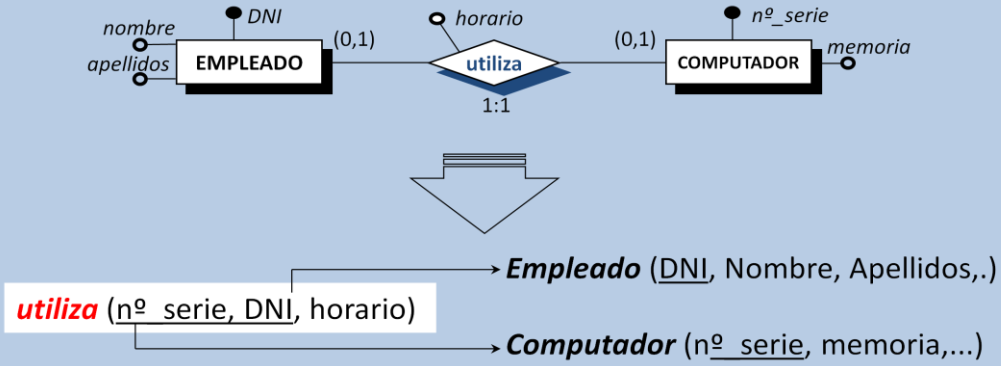

Figura 18. Ejemplo de transformación al modelo relacional cuando se crea una tabla.

Observamos en la figura 18 que la tabla Utiliza se crea a partir de la relación, y esta contiene el nºserie que proviene de la tabla Computador y participa como parte de la clave principal (también es clave foránea), además el atributo DNI que proviene desde la tabla Empleado (también es parte de la clave primaria) y el atributo horario que proviene de la relación original.

Nota: observar la pérdida de eficiencia, ya que muchos consultas implican combinar dos relaciones, e incluso hay consultas que implican combinar las tres relaciones.

b) Propagar una clave

Si una de las cardinalidades mínimas es cero y la otra no (será 1,1), conviene propagar la clave de esta última (la obligatoria).

Ejemplo 3.7

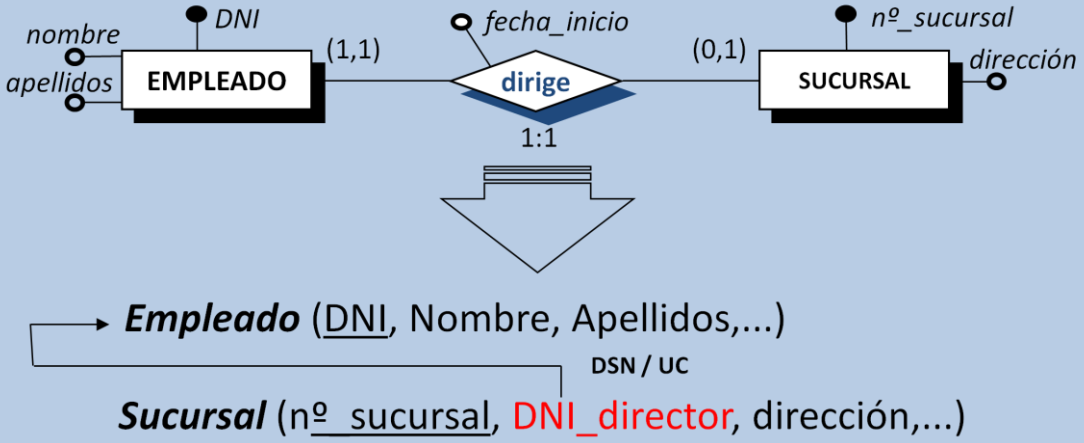

Figura 19. Ejemplo de transformación al modelo relacional cuando se propaga una clave.

Observamos en la figura 19 que la tabla Sucursal contiene el DNI_director que es clave foránea. Esta decisión se ha tomado ya que todas las sucursales tienen un director. No ocurre lo mismo al revés (Empleado con clave foránea) ya que no todos los empleados son directores de una sucursal.

Revisemos algunas ventajas y desventajas de este proceso.

- Inconvenientes:
  - se pierde la simetría
  - consultas a la información de la entidad que interviene con (1,1) suponen combinación natural (por ejemplo, empleados que no dirigen sucursal)
- Ventajas:
  - no pierde semántica (sobre la cardinalidad mínima 1)
  - se evitan valores nulos
  - algunas consultas no precisan combinación de relaciones

Si hay relaciones con atributos, se debe considerar lo siguiente:

- si se crea una nueva tabla, esos atributos se incluyen en esta tabla.
- si se propaga una clave, los atributos acompañan a la clave.
- si se propagan ambas claves, los atributos se incluyen en una de las entidades interrelacionadas.
- si se fusionan en una tabla, esta también contendrá esos atributos.

5. Relaciones recursivas

Las relaciones recursivas se tratan de la misma forma que las otras, sólo que un mismo atributo puede figurar dos veces en una tabla como resultado de la transformación (por eso es interesante indicar el rol en el nombre del atributo.

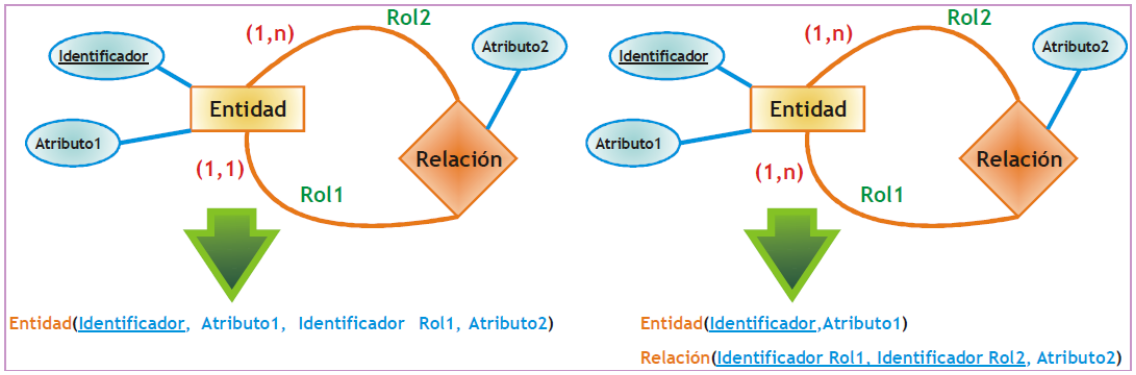

Figura 20. Ejemplo de transformación cuando hay relaciones recursivas

3.5.3 Transformación de las entidades débiles

Toda entidad débil incorpora una relación implícita con una entidad fuerte. Esta relación no necesita incorporarse como tabla en el modelo relacional (al tratarse de una relación 1:N), bastará con añadir como atributo y clave foránea en la entidad débil, el identificador de la entidad fuerte.

En ocasiones el identificador de la entidad débil tiene como parte de su identificador al identificador de la entidad fuerte (por ejemplo si para identificar líneas de factura utilizamos el número de línea y el número de factura, clave de la entidad factura). En esos casos no hace falta añadir de nuevo como clave foránea el identificador de la entidad fuerte (imagen de la derecha).

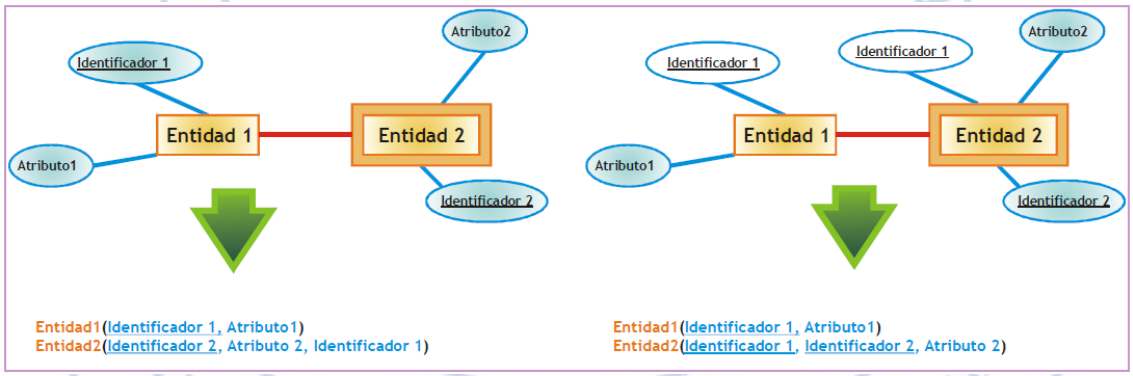

Figura 21. Ejemplo de transformación cuando existen entidades débiles