TEMA 3: MODELO RELACIONAL Y NORMALIZACIÓN

David Bataller Signes

ÍNDICE

2.3 Primera forma normal

2.4 Preservación de información y dependencias en la normalización

2.5 Segunda forma normal

2.6 Tercera forma normal

2.7 Forma normal de Boyce-Codd

2.8 Cuarta forma normal

2.9 Quinta forma normal

2.10 Desnormalización.

1. Modelo relacional.

El modelo relacional es un modelo de datos basado en dos disciplinas matemáticas: la lógica de predicados y la teoría de conjuntos.

Quizás debido a este sólido fundamento teórico, que proporciona a este modelo una robustez excepcional, los SGBD relacionales (o SGBDR) son actualmente los que tienen una mayor implantación en el mercado.

El modelo relacional fue propuesto originalmente por Edgar Frank Codd en su trabajo “A Relational Model of Data for Large Shared Data Banks” ('Un modelo relacional de datos para grandes bancos de datos compartidos') en 1970, aunque no se implementó comercialmente hasta el final de la década.

1.1 Estructuración de los datos

El modelo relacional permite construir estructuras de datos para representar las diferentes informaciones del mundo real que tengan algún interés.

Las estructuras de datos construidas siguiendo el modelo relacional están formadas por conjuntos de relaciones.

Las relaciones pueden ser concebidas como representaciones tabulares de los datos.

Hay que precisar los extremos siguientes:

• Toda relación debe tener un nombre que la identifique unívocamente dentro de la base de datos.

• Cada fila está constituida por una tupla de datos relacionados entre sí, llamado también registro, que guarda los datos que nos interesa reflejar de un objeto concreto del mundo real.

• En cambio, cada columna contiene, en cada celda, datos de un mismo tipo, y se la puede llamar atributo o campo.

• Cada celda, o intersección entre fila y columna, puede almacenar un único valor.

Toda base de datos relacional está formada por un conjunto de relaciones.

Esta sencilla manera de visualizar la estructura de las bases de datos relacionales resulta muy comprensible para la mayoría de usuarios. Pero hay que profundizar en algunas características adicionales de las relaciones, por tal de poderlas distinguir claramente de los ficheros tradicionales.

1.1.1 Dominio

En cuanto al modelo relacional, un dominio consiste en un conjunto finito de valores indivisibles.

Los atributos sólo pueden tomar los valores que estén incluidos dentro del dominio respectivo. De lo contrario no son valores válidos, y un SGBD relacional no puede permitir el almacenamiento.

Debemos considerar dos tipologías de dominios:

• Dominios predefinidos. Son los tipos de datos que admita cada SGBD, como, por ejemplo (mencionados de manera genérica, ya que hay muchas especificidades en función de los diferentes sistemas gestores), las cadenas de caracteres, los números enteros, los números decimales, los datos de carácter cronológico, etc.

• Dominios definidos por los usuarios. Consisten en restricciones adicionales aplicadas sobre el dominio predefinido de algunos atributos, establecidas por los diseñadores y los administradores de bases de datos.

1.1.2 Esquema y extensión

Toda relación consta de un esquema (también llamado intensión de la relación) y de su extensión.

El esquema de una relación consiste en un nombre que la identifica unívocamente dentro de la base de datos, y en el conjunto de atributos que aquella contiene.

Es muy recomendable, para evitar confusiones en la implementación ulterior, seguir uniformemente una notación concreta a la hora de expresar los esquemas de las relaciones que forman una misma base de datos.

A continuación, se detallan las características de uno de los sistemas de notación más frecuentes:

• Hay que escribir el nombre de las relaciones con mayúsculas y preferiblemente en singular.

• Se debe escribir el nombre de los atributos comenzando con mayúscula y continuando con minúsculas, siempre que no se trate de siglas, ya que entonces es más conveniente dejar todas las letras en mayúsculas (como DNI). Para hacer los nombres compuestos más legibles, se puede encabezar cada palabra de las que forman el nombre del campo con una letra mayúscula (por ejemplo: FechaNacimiento, TelefonoParticular, etc.).

Los atributos de una relación son únicos dentro de esta. Su nombre no puede estar repetido dentro de una misma relación. Ahora bien, diferentes relaciones sí pueden contener atributos con el mismo nombre.

Por otra parte, hay que decir que los dominios de diferentes atributos de una misma relación pueden ser idénticos, aunque los campos respectivos almacenen los valores de diferentes propiedades del objeto (por ejemplo, sería perfectamente lógico que los atributos TelefonoFijo, TelefonoMovil y TelefonoTrabajo, a pesar de pertenecer a una misma relación, tuvieran el mismo dominio).

La extensión de una relación consiste en los valores de los datos almacenados en todas los tuplas que ésta contiene.

A veces, los atributos de las relaciones pueden no contener ningún valor o, dicho de otro modo, pueden contener valores nulos.

El grado de una relación depende del número de atributos que incluye su esquema.

La cardinalidad de una relación viene dada por el número de tuplas que forman la extensión.

Tabla 1.
ALUMNO

DNI	Nombre	Apellidos	Telefono
12345678A	Juan	Aaaa Bbbbb	654321987
23456789B	Pedro	Cccc Ddddd	632154987
34567890C	Antonio	Eeeee Fffff	698754321
01234567D	Jose	Ggggg Hhhh	Nulo

Ejemplo de grado de una relación
La relación con esquema ALUMNO (DNI, Nombre, Apellidos, Teléfono) de la tabla 1 es de grado 4, porque tiene cuatro atributos.

Ejemplo de cardinalidad
Si nos fijamos en la tabla 1.3, la cardinalidad de la relación ALUMNO es 4, porque su extensión contiene cuatro tuplas correspondientes a los cuatro alumnos que, de momento, hay matriculados.

También podemos ver un valor nulo en el atributo Telefono de la última entrada de la tabla.

1.1.3 Claves candidatas, clave primaria y claves alternativas

Con el fin de resultar útil, el almacenamiento de la información debe permitir la identificación de los datos. En el ámbito de las bases de datos relacionales, los tuplas de las relaciones se identifican mediante las llamadas superclaves.

Una superclave es un subconjunto de los atributos que forman el esquema de una relación que no es posible que haya más de una tupla en la extensión respectiva, con la misma combinación de valores en los atributos que forman parte del subconjunto mencionado.

Pero una superclave puede contener atributos innecesarios, que no contribuyen a la identificación inequívoca de los diferentes tuplas. Lo que habitualmente interesa es trabajar con superclaves mínimas, tales que ningún subconjunto propio sea capaz para sí solo de identificar los tuplas de la relación.

Por definición, ningún superclave mínima no puede admitir valores nulos en ninguno de sus atributos, porque si lo hiciera, no podría garantizar la identificación inequívoca de los tuplas que contuvieran algún valor nulo en algunos de los atributos de la superclave mínima en cuestión.

Por otra parte, hay que decir que en una misma relación puede pasar que haya más de una superclave mínima que permita distinguir los tuplas unívocamente entre ellos.

Se denominan claves candidatas todas las superclaves mínimas de una relación formadas por los atributos o conjuntos de atributos que permiten identificar las tuplas que contiene su extensión.

Pero, a la hora de implementar una BD, entre todas las claves candidatas de cada relación sólo se ha de elegir una.

Cuando hablamos de clave primaria nos referimos a la clave que, finalmente, el diseñador lógico de la base de datos elige para distinguir unívocamente cada tupla de una relación del resto.

Entonces, las claves candidatas no elegidas como clave primaria quedan presentes en la relación.

Cuando una relación ya tiene establecida una clave primaria, el resto de claves presentes en aquella, y que también podrían servir para identificar los diferentes tuplas de la extensión respectiva, se conocen como claves alternativas.

Una forma de diferenciar los atributos que forman la clave primaria de las relaciones, los otros atributos del esquema respectivo, es ponerlos subrayados. por este motivo, normalmente se colocan juntos y antes de que el resto de atributos, dentro de el

esquema. Pero sólo se trata de una cuestión de elegancia, ya que el modelo relacional no se basa ni en el orden de los atributos del esquema, ni tampoco en el orden de las tuplas de la extensión de la relación.

Si no se dispone de un atributo que sea capaz de identificar los tuplas de la relación por sí solo, hay que buscar un subconjunto de atributos, tales que la combinación de los valores que adopten no se pueda repetir. Si esta posibilidad no existe, hay que añadir a la relación un atributo adicional que haga de identificador.

Por definición, el modelo relacional no admite tuplas repetidas, es decir, no permite la existencia de tuplas en una misma relación que tengan los mismos valores en cada uno de los atributos.

Ahora bien, las implementaciones concretas de los diferentes SGBD sí permiten esta posibilidad, siempre que no se establezca ninguna clave primaria en la relación con tuplas repetidas.

Esta permisividad a veces permite solucionar ciertas eventualidades, pero no debería ser la forma habitual de trabajar con BD relacionales.

1.1.4 Claves foráneas

Cualquier BD relacional, por pequeña que sea, contiene normalmente más de una relación. Para reflejar correctamente los vínculos existentes entre algunos objetos del mundo real, es necesario que los tuplas de las diferentes relaciones de una base de datos se puedan interrelacionar.

A veces, incluso puede ser necesario relacionar las tuplas de una relación con otras tuplas de la misma relación.

El mecanismo que ofrecen las BD relacionales para interrelacionar las relaciones que contienen se fundamenta en las llamadas claves foráneas.

Una clave foránea está constituida por un atributo, o por un conjunto de atributos, del esquema de una relación, que sirve para relacionar sus tuplas con las tuplas de otra relación de la base de datos (o con las tuplas de ella misma, en algunos casos).

Para conseguir conectar los tuplas de una relación con los de otra (o con sus propias), la clave foránea utilizada debe referenciar la clave primaria de la relación con la que se quiere relacionar.

Las diferentes combinaciones de valores de los atributos de toda clave foránea deben existir en la clave primaria a que se refieren, o bien deben ser valores nulos. De lo contrario, las referencias serían erróneas y, por tanto, los datos serían incorrectos.

Hay que prestar atención en las características siguientes de las claves foráneas:

• Toda clave foránea debe tener el mismo número de atributos que la clave primaria a la que hace referencia.

• Entre los atributos del esquema de una clave foránea y los de la clave primaria respectiva debe poder establecer una correspondencia (concretamente, una biyección).

• Los dominios de los atributos de toda clave foránea deben coincidir con los dominios de los atributos de la clave primaria respectiva (o, al menos, hay que sean compatibles dentro de un cierto rango).

Una relación puede contener más de una clave foránea, o bien no contener ninguna. Y, en sentido inverso, la clave primaria de una relación puede estar referenciada por una o más claves foráneas, o bien puede no estar referenciada por ninguna.

Finalmente, hay que decir que se puede dar el caso de que un mismo atributo forme parte tanto de la clave primaria de la relación como de alguna de sus claves foráneas.

La notación más habitual para designar las claves foráneas de las relaciones consiste a añadir esta circunstancia después de su esquema, incluyendo entre dos llaves el conjunto de atributos que forman la clave foránea de que se trate, precedido del adverbio DONDE, y seguido de la forma verbal REFERENCIA y de la relación a la que hace referencia. Si hay más de una clave foránea, se separan por comas y la última ha ir precedida de la conjunción Y.

Ejemplo de notación para designar claves foráneas

Las dos relaciones que se muestran en las tablas 2 y 3 se expresarán de la siguiente forma:

Tabla 2.

ALUMNO

DNI	Nombre	Apellidos	Telefono	CodigoAula	DNIDelegado
12345678A	Juan	Aaaa Bbbbb	654321987	110	12345678A
23456789B	Pedro	Cccc Ddddd	632154987	110	12345678A
34567890C	Antonio	Eeeee Fffff	698754321	201	34567890C
01234567D	Jose	Ggggg Hhhh	Nulo	201	34567890C

Tabla 3.

AULA

Codigo	Capacidad
110	55
205	70
103	63
201	42

ALUMNO(DNI, Nombre, Apellidos, Telefono, DNIDelegado, CodigoAula) DONDE {DNIDelegado} REFERENCIA ALUMNO y {CodigoAula} REFERENCIA AULA

AULA(Codigo, Capacidad)

1.1.5 Operaciones con relaciones

El modelo relacional permite realizar una serie de operaciones con los datos almacenados en las BD, las cuales tienen diferentes finalidades:

• Actualización. Estas operaciones realizan cambios en los tuplas que quedan reflejados en las relaciones que contienen las BD. Pueden ser de tres tipos:

- Inserción. Consiste en añadir uno o más tuplas nuevas a una relación determinada.
- Borrado. Consiste en eliminar uno o más tuplas nuevos de una relación determinada.
- Modificación. Consiste en cambiar el valor de uno o más atributos de una o más tuplas de una relación determinada.

• Consulta. Estas operaciones sólo hacen posible la obtención parametrizada de datos, sin que se vean alteradas las almacenadas en la BD.

La realización de estas operaciones implica el conocimiento previo de la estructura formada por las relaciones que sea necesario utilizar, es decir, los esquemas de las relaciones y las interrelaciones entre ellas, mediante las claves foráneas.

1.2 Reglas de integridad

Los valores que almacenan las BD deben reflejar en todo momento, de manera correcta, la porción de la realidad que queremos modelizar.

Llamamos integridad la propiedad de los datos que consiste en representar correctamente las situaciones del mundo real que modelizan.

Para que los datos sean íntegros, hay que garantizar que sean correctos, y también que estén enteras.

En atención al objetivo mencionado, pues, los datos deben cumplir ciertas condiciones, que podemos agrupar en dos tipologías diferentes:

• Restricciones de integridad del usuario. Son condiciones específicas de cada BD. Los SGBD deben permitir a los administradores establecer ciertas restricciones aplicables a casos concretos, y deben garantizar que se respeten durante la explotación habitual del sistema.

• Reglas de integridad del modelo. Son condiciones de carácter general que deben cumplir todas las BD que sigan el modelo relacional. No es necesario definirlas en implementar cada BD, porque se consideran preestablecidas.

Ejemplo de restricción de integridad del usuario.

Al dar de alta un nuevo alumno de la relación ALUMNO, que se muestra en la tabla 2, podríamos exigir al sistema que el validara, mediante el algoritmo correspondiente, si la letra introducida del NIF se corresponde con las cifras introducidas previamente, y que denegara la inserción en caso contrario, para no almacenar una situación en principio no admisible en el mundo real.

Ejemplo de regla de integridad del modelo.

Como la relación ALUMNO, que se muestra en la tabla 2, tiene definido el atributo DNI como clave primaria, el sistema validará automáticamente que no se introduzca más de un alumno con el mismo carné de identidad, ya que entonces la clave primaria no cumpliría su objetivo de garantizar la identificación inequívoca de cada tupla, diferenciándola del resto.

1.2.1 Unicidad de la clave primaria

El valor de una clave primaria, globalmente considerada, no puede estar repetido en más de una tupla de la misma relación, ya que entonces la clave primaria no estaría en condiciones de asegurar la identificación inequívoca de las diferentes tuplas.

En ningún momento, no puede haber dos o más tuplas con la misma combinación de valores en el conjunto de los atributos que forman la clave primaria de una relación.

Los SGBD relacionales deben garantizar la regla de unicidad de la clave primaria en todas las inserciones de nuevos tuplas, y también en todas las modificaciones que afecten el valor de alguno de los atributos que formen parte de la clave primaria.

1.2.2 Entidad de la clave primaria

Las claves primarias sirven para diferenciar cada tupla de una relación del resto de tuplas de la misma relación. Para garantizar la consecución de este fin, es necesario que los atributos que forman parte de una clave primaria no puedan tener valor nulo ya que, si se admitiera esta posibilidad, las tuplas con valores nulos en la clave primaria no se podrían distinguir de algunos otros.

Ningún atributo que forme parte de una clave primaria no puede contener más valores nulos en ninguna tupla.

Los SGBD relacionales deben garantizar la regla de entidad de la clave primaria en todas las inserciones de nuevos tuplas, así como en todas las modificaciones que afecten al valor de alguno de los atributos que formen parte de la clave primaria.

1.2.3 Integridad referencial

El modelo relacional no admite, por definición, que la combinación de valores de los atributos que forman una clave foránea no esté presente en la clave primaria correspondiente, ya que ello implicaría una conexión incorrecta.

La integridad referencial implica que, para cualquier tupla, la combinación de valores que adopta el conjunto de los atributos que forman la clave foránea de la relación o bien debe estar presente en la clave primaria a la que hace referencia, o bien debe estar constituida exclusivamente por valores nulos (si los atributos implicados admiten esta posibilidad, y así se ha estipulado en definir sus propiedades).

Los SGBD relacionales deberán hacer las comprobaciones pertinentes, de manera automática, para garantizar la integridad referencial, cuando se produzcan dos tipos de operaciones con relaciones que tengan claves foráneas:

• Inserciones de nuevas tuplas.

• Modificaciones que afecten atributos que formen parte de cualquier clave foránea.

Por otra parte, los SGBD relacionales también deberán validar la corrección de otros dos tipos de operaciones con relaciones que tengan la clave primaria referenciada desde alguna clave foránea:

• Borrado de tuplas.

• Modificaciones que afecten atributos que formen parte de la clave primaria.

Para garantizar la integridad referencial en estos dos últimos tipos de operación, se puede seguir alguna de las tres políticas: restricción, actualización en cascada y anulación.

Ejemplo de violación de la integridad referencial.

Continuamos especulando con las relaciones ALUMNO y AULA (reflejadas en la tabla 2 y tabla 3 respectivamente).
La tupla que contiene los datos de Jose tiene un valor 201 en el atributo que forma la clave foránea que hace referencia a la relación AULA.
Si quisiéramos actualizar con el valor 316, por ejemplo, el sistema no nos debería dejar hacer, porque este valor no está presente en la clave primaria de ninguna tupla

de la relación AULA y, por tanto, esta operación contravendría la regla de integridad referencial.

Política de restricción

La política de restricción consiste en prohibir la operación de actualización de que se trate:

• En caso de borrado, no permitirá eliminar una tupla si su clave primaria está referenciada desde alguna clave foránea.

• En caso de modificación, no permitirá alterar el valor de ninguno de los atributos que forman la clave primaria de una tupla, si ésta está referenciada desde alguna clave foránea.

Ejemplos de restricciones

Consideramos una vez más las relaciones ALUMNO y AULA.
Aplicando la restricción tanto en caso de borrado como de modificación, estas operaciones no serán posibles con el aula 110 de la relación AULA porque hay alumnos matriculados que deben asistir a clase dentro de este espacio y, por tanto, la referencian desde la clave foránea las tuplas que los representan. Sí sería posible, en cambio, borrar el aula 103, porque no está referenciada desde la relación ALUMNO.

Actualización en cascada

La política de actualización en cascada consiste en permitir la operación de actualización de que se trate sobre una tupla determinada, pero disponiendo al mismo tiempo una serie de operaciones compensatorias que propaguen en cascada las actualizaciones necesarias para que se mantenga la integridad referencial de las tuplas que referencian, desde los atributos que forman la clave foránea, la tupla objeto de actualización:

• En caso de borrado, se eliminarán todas las tuplas que hagan referencia a la tupla borrada.

• En caso de modificación, los valores de los atributos que formen parte de la clave foránea de las tuplas que hagan referencia a la tupla modificada alterarán para continuar coincidiendo con los nuevos valores de la clave primaria de la tupla al que hacen referencia.

Ejemplos de actualización en cascada

Volvemos a tomar como punto de partida de los ejemplos las relaciones ALUMNO y AULA.

Si aplicamos la actualización en cascada borrando la tupla <201, 42> de la relación AULA, también se borrarán las dos tuplas de la relación ALUMNO que hacen referencia desde la clave foránea respectiva (CodigoAula).
En cambio, si aplicamos la actualización en cascada modificando la tupla <201, 42> de la relación AULA, cambiando el valor de su clave primaria por otro, tales como 203, las dos tuplas de la relación ALUMNO que hacen referencia actualizarán en cascada el valor del atributo CodigoAula de 201 a 203, a fin de mantener la conexión correcta entre las tuplas de ambas relaciones.

Política de anulación

La política de anulación consiste en permitir la operación de actualización de que se trate en una tupla determinada, pero disponiendo al mismo tiempo una serie de operaciones compensatorias que pongan valores nulos en todos los atributos que formen parte de las claves foráneas de las tuplas que hagan referencia a la tupla objeto de actualización:

• En caso de borrado, los atributos de la clave foránea de las tuplas que hagan referencia a la tupla borrada pasarán a tener valor nulo, y no indicarán ningún tipo de conexión.

• En caso de modificación, los atributos de la clave foránea de las tuplas que hagan referencia a la tupla modificada pasarán a tener valor nulo, y no indicarán ningún tipo de conexión.

La política de anulación sólo puede aplicarse si los atributos de las claves foráneas implicadas admiten los valores nulos.

Ejemplo de anulación

Tomamos una vez más como punto de partida de los ejemplos las relaciones ALUMNO y AULA.
Aplicando la política de anulación, tanto si borramos la tupla <110, 55> de la relación AULA, como si sólo cambiamos el valor del atributo de su clave primaria (Código) por otro (como, por ejemplo, 105), la tupla de la relación ALUMNO que hace referencia actualizará el valor del atributo CodigoAula de 110 a valor nulo, para evitar una conexión incorrecta entre las tuplas de ambas relaciones.

Selección de la política a seguir

Será el diseñador de cada BD quien escogerá la política más adecuada que se debe seguir en cada caso concreto. Como orientación, conviene saber que las opciones más frecuentes, siempre que no haya que hacer consideraciones adicionales, son las siguientes:

• En caso de borrado, normalmente se opta por la restricción.

• En caso de modificación, el más habitual es optar por la actualización en cascada.

La política de anulación es mucho menos frecuente, y se pone en práctica cuando se quieren conservar ciertos datos, aunque hayan perdido la conexión que tenían antes, a veces con la esperanza de que los puedan recuperar más adelante.

1.2.4 Integridad del dominio

La regla de integridad del dominio implica que todos los valores no nulos que contienen los atributos de las relaciones de cualquier BD deben pertenecer a los respectivos dominios declarados para los atributos en cuestión.

Esta condición es aplicable tanto a los dominios predefinidos, así como con respecto a los dominios definidos por el usuario.

La regla de integridad del dominio también conlleva que los operadores que es posible aplicar sobre los valores dependen de las propiedades de los respectivos atributos que los almacenan.

Ejemplos de integridad de dominio

Si en la relación con esquema AULA (Código, Capacidad) definimos el dominio de el atributo Código como el de los números enteros de 0 hasta 999, entonces no podremos insertar, por ejemplo, un valor en el atributo que forma la clave primaria que no pertenezca a su dominio, tales como INF, o LAB.

Tampoco podremos aplicar determinados operadores para comparar valores de la clave primaria con valores que no pertenezcan a su dominio. Así, no podremos consultar las características de un aula con Código = INF ', ya que' INF 'es una cadena de caracteres.

1.3 Traducción del modelo Entidad-Relación al modelo relacional

Una vez conocidas las características de un modelo de bases de datos relacional, habrá que partir del modelo conceptual general (del modelo Entidad-Relación) y hacer un estudio del diseño lógico de bases de datos en este ámbito, el relacional.

En todos los ejemplos, presupone que previamente ha tenido lugar una fase de diseño conceptual de la que ha resultado un modelo Entidad-Relación recogido en los diagramas Chen de que se trate en cada caso.

Antes de implementar propiamente la BD dentro del entorno ofrecido por el SGBD utilizado, hay que transformar estos diagramas en estructuras de datos relacionales.

El modelo ER se basa en las entidades y en las interrelaciones existentes entre ellas.

Podemos avanzar unos cuantos aspectos generales sobre cómo se deben traducir estos elementos al modelo relacional:

• Las entidades siempre dan lugar a relaciones, sean del tipo que sean (a excepción de las entidades auxiliares de tipo FECHA).

• Las interrelaciones binarias de conectividad 1-1 o 1-N originan claves foráneas en relaciones ya existentes.

• Las interrelaciones binarias de conectividad M-N y todas las n-arias de orden superior a 2 siempre se transforman en nuevas relaciones.

Es conveniente seguir un cierto orden a la hora de diseñar lógicamente una base de datos. Una buena práctica puede consistir proceder de la siguiente manera:

1. En primer lugar, hay que transformar las entidades del diagrama con el que trabajamos en relaciones.

2. Luego se debe continuar transformando en relaciones las entidades que presentan algún tipo de especificidad (es decir, las débiles, las asociativas, o las derivadas de un proceso de generalización o especialización).

3. A continuación, se deben añadir a las anteriores relaciones los atributos necesarios para formar las claves foráneas derivadas de las interrelaciones binarias con conectividad 1-1 y 1-N presentes en el diagrama ER.

4. Y, finalmente, ya puede comenzar la transformación de las interrelaciones binarias con conectividad M-N y de las interrelaciones n-arias.

De esta manera evitaremos que haya claves foráneas que hagan referencia a relaciones que aún no se han descrito. Esto hace más lector el modelo relacional obtenido, ciertamente, pero también ahorra el trabajo de tener que ordenar las relaciones a la hora de escribir (típicamente en lenguaje SQL) y las instrucciones pertinentes para que el SGBD utilizado cree las tablas de la base de datos.

Las técnicas necesarias para realizar correctamente el diseño lógico de bases de datos, según el tipo de conceptualización de que se trate en cada caso.

1.3.1 Entidades

Cada entidad del modelo ER se transforma en una relación del modelo relacional:

• Los atributos de la entidad originaria serán los atributos de la relación resultante.

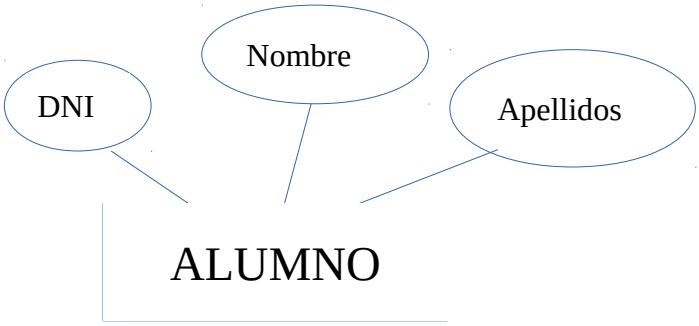
• La clave primaria de la entidad originaria será la clave primaria de la relación resultante.

• Cuando una entidad interviene en alguna interrelación binaria 1-1 o 1-N, puede ser necesario añadir ulteriormente nuevos atributos, para que actúen como claves foráneas de la relación.

Ejemplo de transformación de entidad

El diagrama ER de la figura 4 se traduce al modelo relacional de la siguiente forma:

Figura 4.



ALUMNO (<u>DNI</u>, Nombre, Apellidos)

1.3.2 Interrelaciones

Una vez transformadas todas las entidades en relaciones, hay que traducir las interrelaciones en que aquellas participan.

1. Binarias. Para traducir las interrelaciones binarias hay que tener en cuenta su conectividad, así como las dependencias de existencia.
a) Conectividad 1-1 y dependencias de existencia. Hay que añadir a cualquiera de las dos relaciones una clave foránea que haga referencia a la otra relación.

Pero si una de las dos entidades es opcional en la relación, entonces es ella quien debe acoger la clave foránea, para evitar, en caso contrario, el almacenamiento de valores nulos en esta, y ahorrarse así espacio de almacenamiento.

Los atributos de la interrelación (si los hay) acompañan la clave foránea.

Ejemplo de transformación de interrelación binaria con conectividad 1-1.

El diagrama ER de la figura 5 representa una interrelación binaria con conectividad 1-1. Por tanto, en principio habría dos posibilidades de transformación, según si se coloca la clave foránea en la entidad PROFESOR o en la entidad DEPARTAMENTO:

DEPARTAMENTO (Código, Descripcion)

PROFESOR (DNI, nombre, apellidos, CodigoDepartamento) DONDE {CodigoDepartamento} REFERENCIA DEPARTAMENTO y CodigoDepartamento ADMITE VALORES NULOS

O bien:

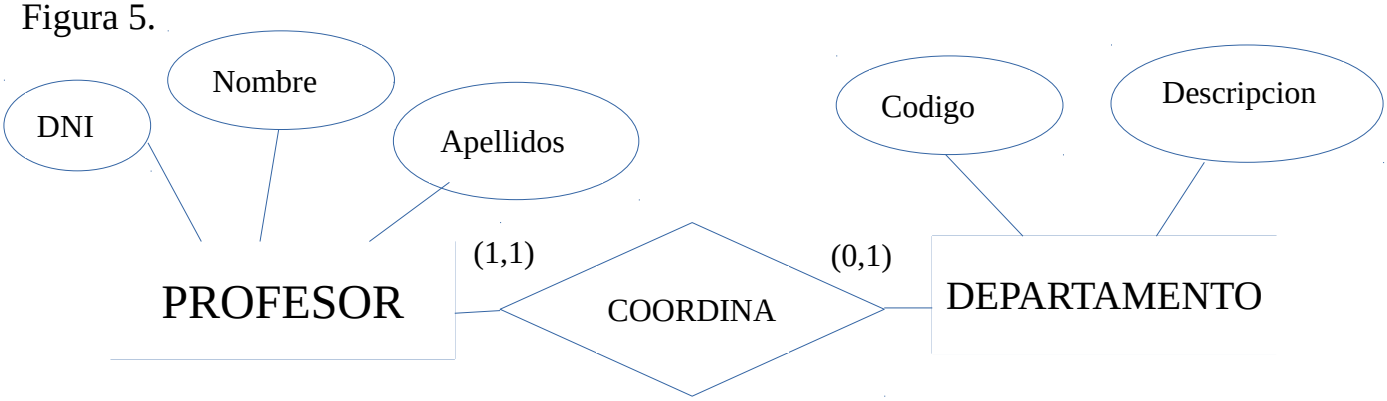
PROFESOR (DNI, Nombre, Apellidos)

DEPARTAMENTO (Código, Descripcion, DNIProfesor) DONDE {DNIProfesor} REFERENCIA PROFESOR

Ahora bien, la entidad DEPARTAMENTO es opcional en la interrelación Coordina. Esto significa que puede haber profesores que no coordinen ningún departamento.

Por lo tanto, la opción más correcta consiste en añadir la clave foránea a la relación DEPARTAMENTO, ya que si se añadiera a la relación PROFESOR debería tomar el valor nulo en muchos casos, y ocuparía un espacio de almacenamiento innecesario.

Figura 5.



b) Conectividad 1-N. En estos casos hay que añadir una clave foránea en la relación que resulta de traducir la entidad ubicada en el lado N de la interrelación, que haga referencia en la otra relación.

Si se colocara la clave foránea en la otra relación, el atributo que la forma debería ser multivalente para poder representar todas las conexiones posibles, y esto no está permitido dentro del modelo relacional.

Los atributos de la interrelación (si los hay) acompañan la clave foránea.

Ejemplo de transformación de interrelación binaria con conectividad 1-N

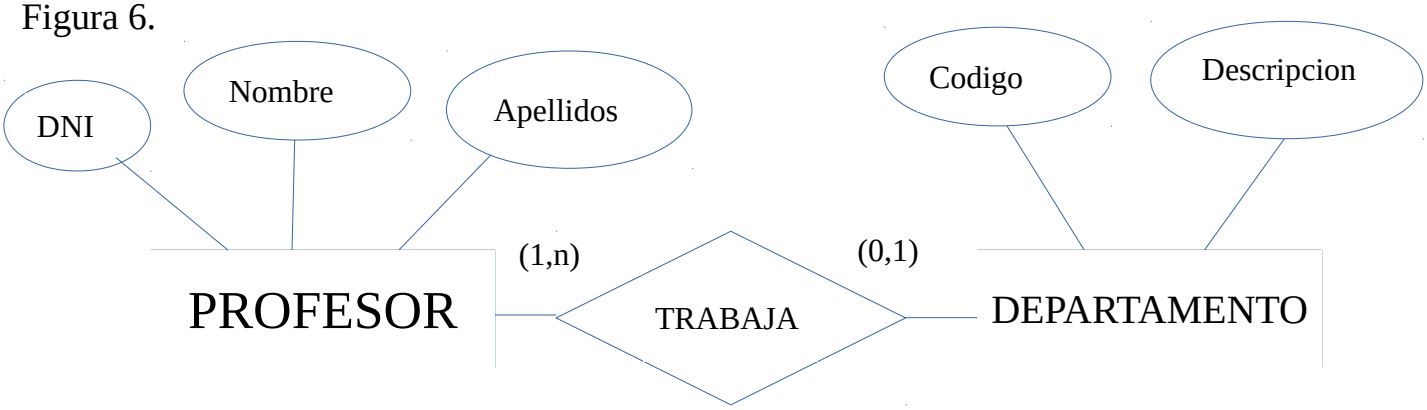
El diagrama ER de la figura 6 representa una interrelación binaria con conectividad 1-N. Por lo tanto, la clave foránea deberá añadir necesariamente a la entidad derivada de la entidad del lado N, y resulta el siguiente modelo:

DEPARTAMENTO (Código, Descripcion)

PROFESOR (DNI, nombre, apellidos, CodigoDepartamento) DONDE {CodigoDepartament} REFERENCIA DEPARTAMENTO y CodigoDepartamento ADMITE VALORES NULOS

La entidad del lado 1 (DEPARTAMENTO) es opcional en la interrelación Trabaja. esto implica que la entidad PROFESOR admitirá valores nulos en su llave foránea que hace referencia a DEPARTAMENTO, ya que podrá haber profesores no asignados a ningún departamento. Pero, al contrario de lo que ocurría con las interrelaciones 1-1, aquí no se podrán evitar estos valores nulos, ya que la clave foránea debe ir necesariamente a la entidad resultante de traducir al modelo relacional la entidad ubicada en el lado N de la interrelación.

Figura 6.



c) Conectividad M-N. Cada interrelación M-N se transforma en una nueva relación con las siguientes características:

• Su clave primaria estará formada por los atributos de las claves primarias de las dos entidades interrelacionadas.

• Los atributos de la interrelación (si los hay) se convertirán en atributos de la nueva relación.

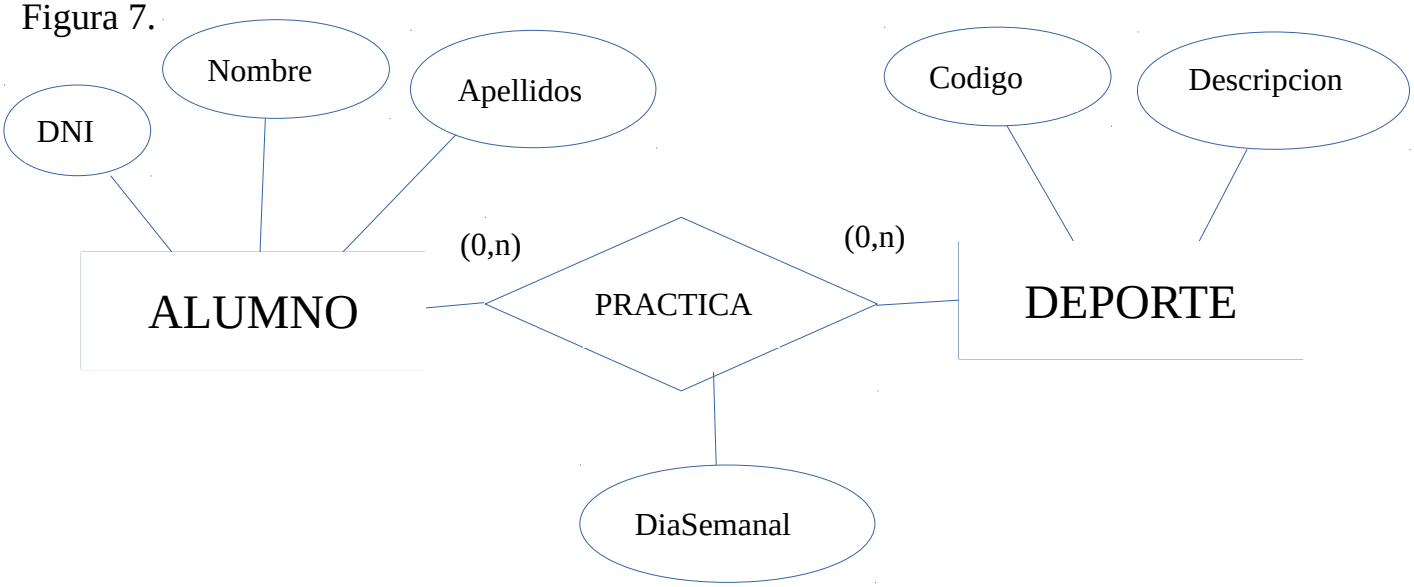
Ejemplo de transformación de interrelación binaria con conectividad M-N
El diagrama ER de la figura 7 se traduce al modelo relacional de la siguiente forma:

ALUMNO (DNI, Nombre, Apellidos)

DEPORTE (Codigo, Descripcion)

PRACTICA (DNIAlumno, CodigoDeporte, DiaSemanal) DONDE {DNIAlumno} REFERENCIA ALUMNO y {CodigoDeporte} REFERENCIA DEPORTE

Figura 7.



2. Ternarias. Toda interrelación ternaria se transforma en una nueva relación, que tendrá por atributos los de las claves primarias de las tres entidades interrelacionadas, más los atributos propios de la interrelación, si tiene.

La composición de clave primaria de la nueva relación depende de la conectividad de la interrelación ternaria originaria.

a) Conectividad M-N-P. En este caso, la clave primaria está formada por todos los atributos que forman las claves primarias de las tres entidades interrelacionadas (Si no fuera así, la clave primaria debería repetir algunas combinaciones de sus valores para modelizar todas las posibilidades, pero esta posibilidad no está permitida dentro del modelo relacional).

Ejemplo de transformación de interrelación ternaria con conectividad M-N-P

El diagrama ER de la figura 8 se traduce al modelo relacional de la siguiente forma:

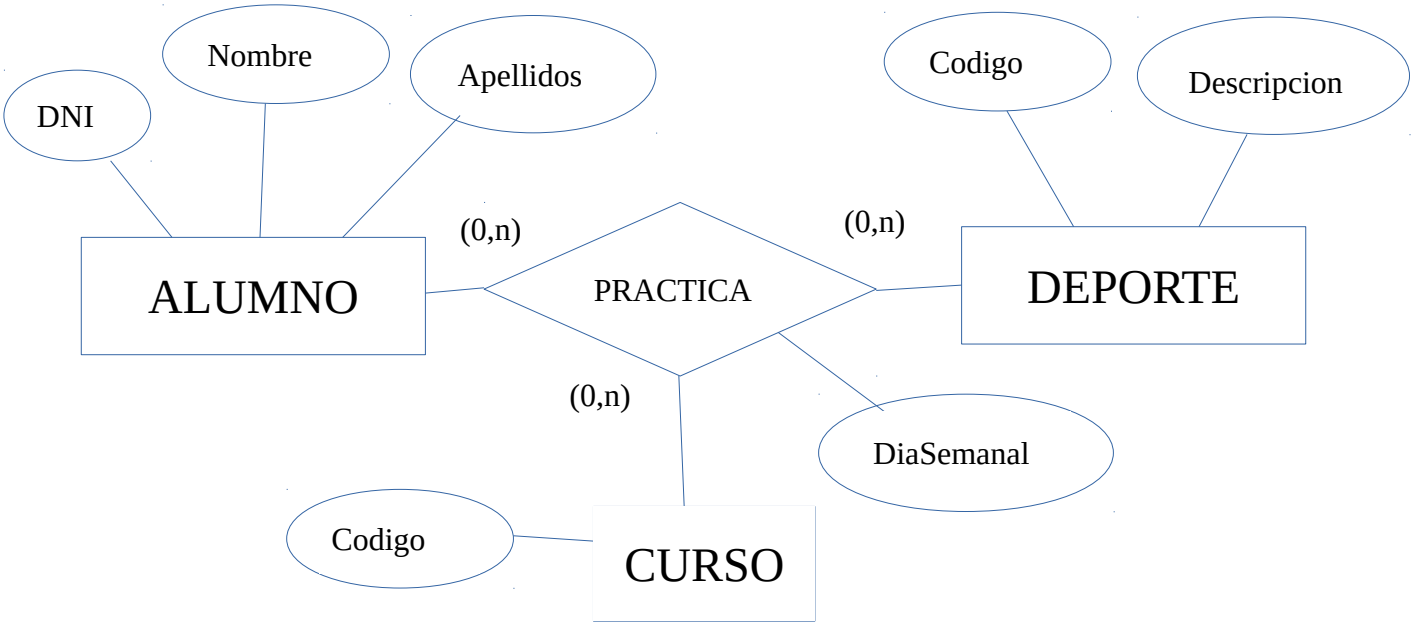
ALUMNO (DNI, Nombre, Apellidos)

DEPORTE (Código, Descripcion)

CURSO (Código)

PRACTICA (DNIAlumno, CodigoDeporte, CodigoCurso, DiaSemanal) DONDE {DNIAlumno} REFERENCIA ALUMNO {CodigoDeporte} REFERENCIA DEPORTE y {CodigoCurso} REFERENCIA CURSO

Figura 8



b) Conectividad 1-M-N. La clave primaria está compuesta por todos los atributos que forman las claves primarias de las dos entidades que están en ambos lados de la interrelación etiquetados con una N (o con lo que es equivalente, una flecha de punta doble).

Ejemplo de transformación de interrelación ternaria con conectividad 1-M-N

El diagrama ER de la figura 9 se traduce al modelo relacional de la siguiente forma:

ALUMNO (DNI, Nombre, Apellidos)

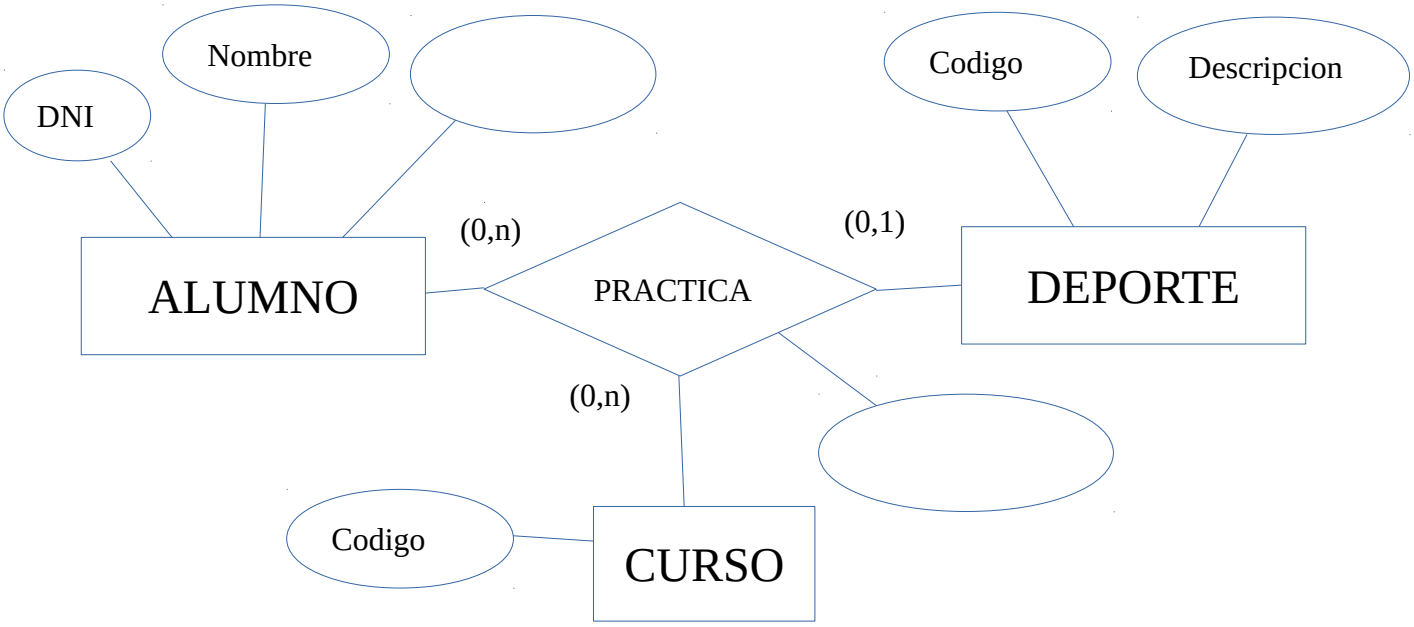
DEPORTE (Código, Descripción)

CURSO (Código)

PRACTICA (DNIAlumno, CodigoDeporte, CodigoCurso, DiaSemanal) DONDE {DNIAlumno} REFERENCIA ALUMNO {CodigoDeporte} REFERENCIA DEPORTE, Y {CodigoCurso} REFERENCIA CURSO

Fijémonos en que, en este caso, un alumno sólo puede practicar un deporte en cada curso académico y, por tanto, no es necesario incorporar la clave de la entidad DEPORTE en la clave de la relación PRACTICA.

Figura 9



c) Conectividad 1-1-N. En estos casos, la clave primaria está compuesta por los atributos que forman la clave primaria de la entidad del lado N de la interrelación, más los atributos que forman la clave primaria de cualquiera de las otras dos entidades conectadas con cardinalidad 1.

Así pues, toda nueva relación derivada de una interrelación ternaria con conectividad 1-1-N dispondrá de dos claves candidatas. La elección de una de estas como clave primaria de la nueva relación quedará al criterio del diseñador lógico de BD.

Ejemplo de transformación de interrelación ternaria con conectividad 1-1-N

El diagrama ER de la figura 10 se puede traducir al modelo relacional de dos maneras:

ALUMNO (DNI, Nombre, Apellidos)

DEPORTE (Código, Descripción)

CURSO (Código)

COORDINACION (CodigoCurso, DNIAlumno, CodigoDeporte, DiaSemanal) DONDE {DNIAlumno} REFERENCIA ALUMNO {CodigoDeporte} REFERENCIA DEPORTE y {CodigoCurso} REFERENCIA CURSO

O bien:

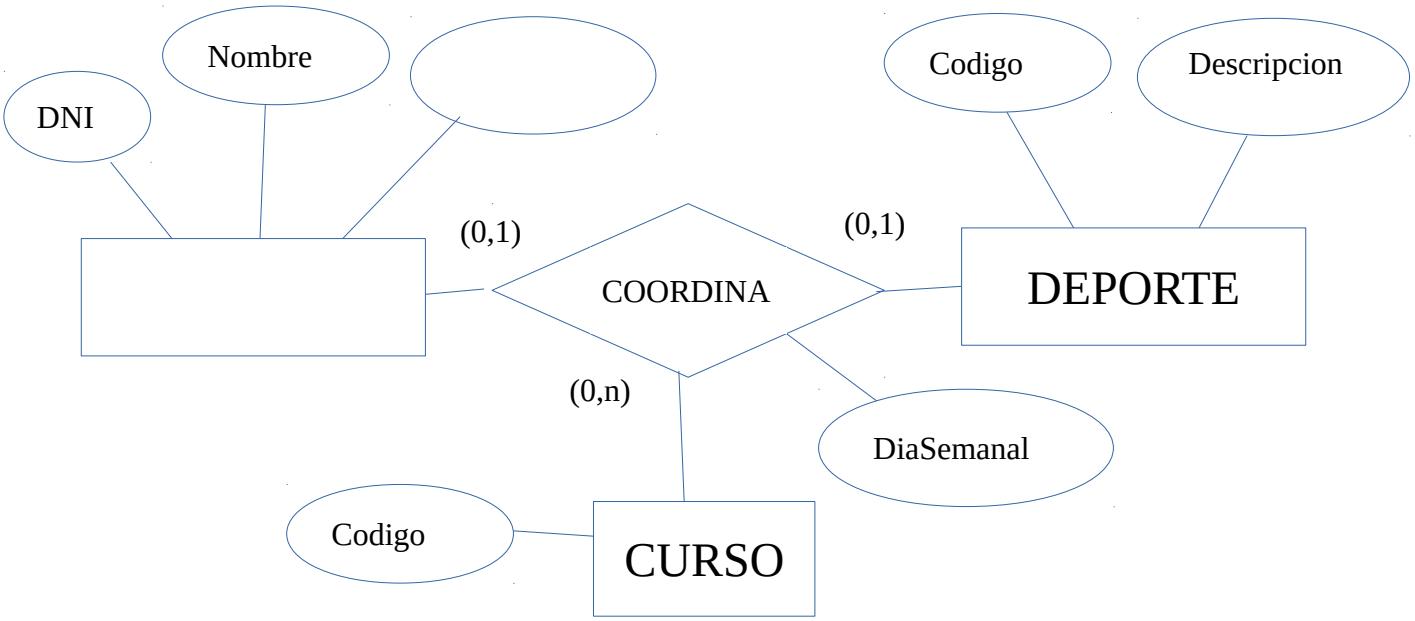
ALUMNO (DNI, Nombre, Apellidos)

DEPORTE (Código, Descripción)
CURSO (Código)

COORDINACION (CodigoCurso, CodigoDeporte, DNIAlumno, DiaSemanal) DONDE {DNIAlumno} REFERENCIA ALUMNO {CodigoDeporte} REFERENCIA DEPORTE y {CodigoCurso} REFERENCIA CURSO

Se puede ver que hemos modificado el nombre de la relación derivada de la interrelación con el fin de convertir el verbo originario en un sustantivo, que normalmente es más adecuado para designar relaciones.

Figura 10



d) Conectividad 1-1-1. En estos casos, la clave primaria está compuesta por los atributos que forman la clave primaria de dos entidades cualesquiera, ya que las tres están conectadas con cardinalidad 1.

Así pues, toda nueva relación derivada de una interrelación ternaria con conectividad 1-1-1 dispondrá de tres claves candidatas. La elección de una de estas como clave primaria de la nueva relación quedará al criterio del diseñador lógico de BD.

Ejemplo de transformación de interrelación ternaria con conectividad 1-1-1

El diagrama ER de la figura 11 se puede traducir al modelo relacional de tres maneras:

ALUMNO (DNI, Nombre, Apellidos)

DEPORTE (Código, Descripción)

CURSO (Código)

COORDINACION (CodigoCurso, DNIAlumno, CodigoDeporte, DiaSemanal) DONDE {DNIAlumno} REFERENCIA ALUMNO {CodigoDeporte} REFERENCIA DEPORTE y {CodigoCurso} REFERENCIA CURSO

O bien:

ALUMNO (DNI, Nombre, Apellidos)

DEPORTE (Código, Descripción)

CURSO (Código)

COORDINACION (CodigoCurso, CodigoDeporte, DNIAlumno, DiaSemanal) DONDE {DNIAlumno} REFERENCIA ALUMNO {CodigoDeporte} REFERENCIA DEPORTE y {CodigoCurso} REFERENCIA CURSO

O bien:

ALUMNO (DNI, Nombre, Apellidos)

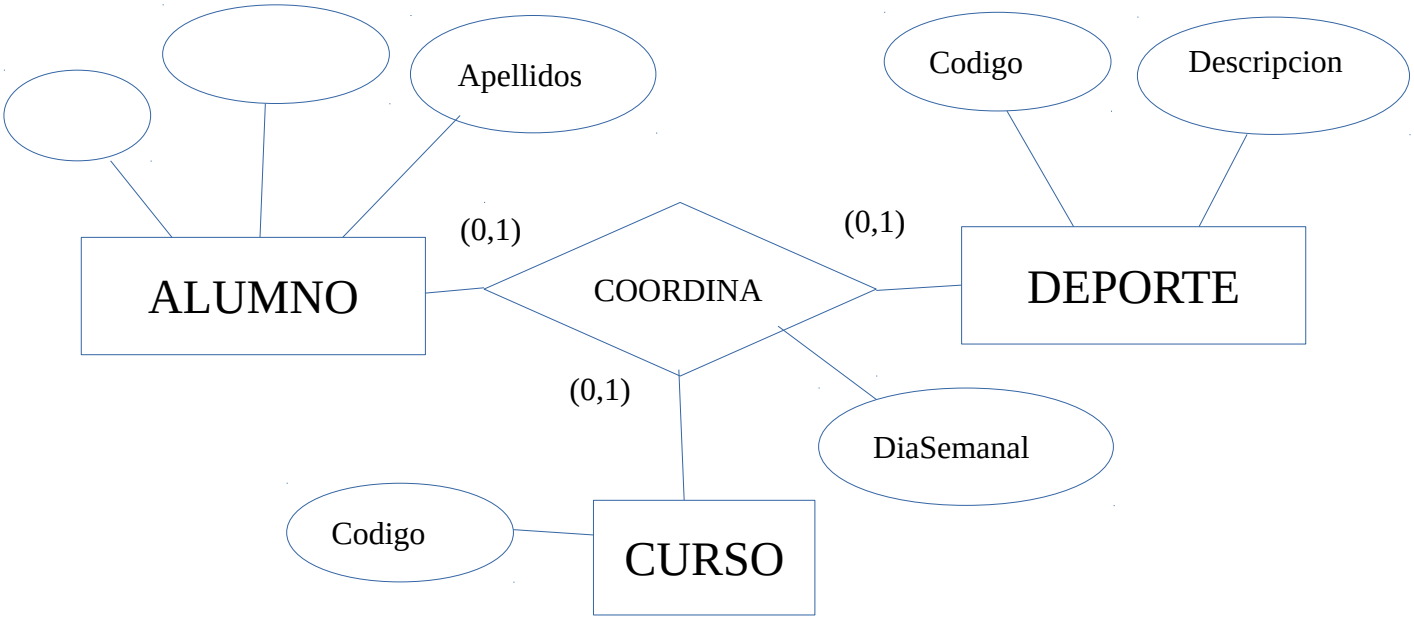
DEPORTE (Código, Descripción)

CURSO (Código)

COORDINACION (CodigoDeporte, DNIAlumno, CodigoCurso, DiaSemanal) DONDE {DNIAlumno} REFERENCIA ALUMNO {CodigoDeporte} REFERENCIA DEPORTE y {CodigoCurso} REFERENCIA CURSO

Fijémonos que hemos cambiado el significado del diagrama respecto a lo que hemos representado en la figura 10, ahora un alumno sólo puede coordinar la práctica de un deporte durante un solo curso académico, a lo largo de sus estudios, para favorecer la rotación en los cargos de coordinación del centro.

Figura 11



3. n-arias. Cada interrelación n-aria se transforma en una nueva relación, que tiene como atributos las claves primarias de todas las entidades relacionadas, más los atributos propios de la interrelación originaria, si tiene.

La composición de la clave primaria de la nueva relación depende de la conectividad de la interrelación n-aria.

a) Conectividad de todas las entidades con cardinalidad N. La clave primaria está formada por todos los atributos que forman las claves primarias de todas las entidades interrelacionadas (n).

Hay que seguir el mismo mecanismo que con las interrelaciones ternarias con conectividad M-N-P.

b) Conectividad de una o más entidades con cardinalidad 1. La clave primaria está formada por todos los atributos que forman las claves primarias de todas las entidades interrelacionadas excepto una (n-1). La entidad que no incorpora su clave primaria a la de la nueva relación debe estar forzosamente conectada con un 1.

Hay que seguir el mismo mecanismo que con las interrelaciones ternarias con conectividad 1-M-N, 1-1-N y 1-1-1.

4. recursiva. Las interrelaciones recursivas traducidas se comportan de la misma por lo que la del resto de interrelaciones:

• Las binarias con conectividad 1-1 y 1-N dan lugar a una clave foránea.

• Las binarias con conectividad M-N y n-arias originan una nueva relación.

a) Binarias con conectividad 1-1 o 1-N. En estas situaciones, hay que añadir a la relación surgida de la entidad originaria que se relaciona con ella misma una llave foránea que haga referencia a la propia clave primaria.

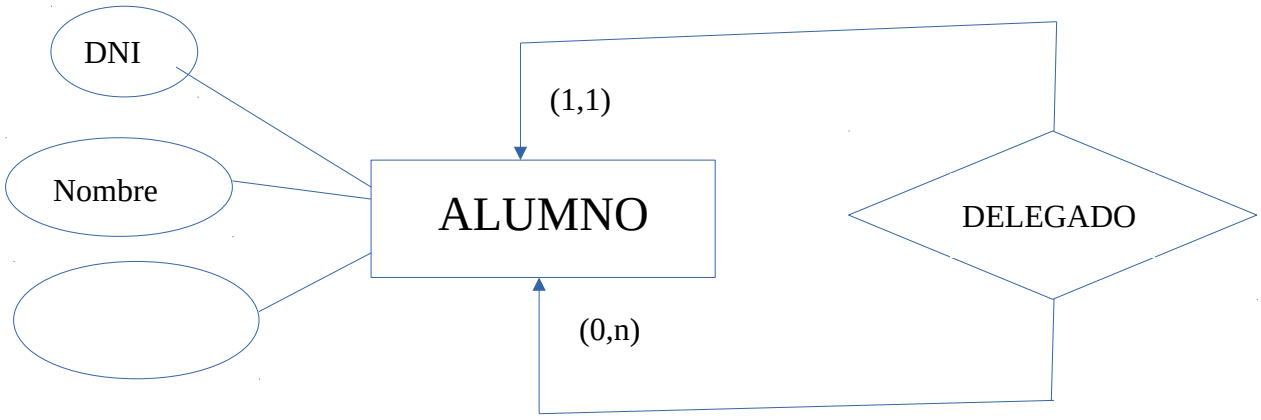
Evidentemente, los atributos de la clave foránea no pueden tener los mismos nombres que los de la clave primaria a los que hacen referencia, ya que ambos se encuentran en la misma relación, y eso atentaría contra los principios del modelo relacional.

Ejemplo de transformación de interrelación recursiva binaria con conectividad 1-N

El diagrama ER de la figura 12 se traduce al modelo relacional de la siguiente forma:

ALUMNO (DNI, nombre, apellidos, DNIDelegado) DONDE {DNIDelegado} REFERENCIA ALUMNO

Figura 12.



b) Binarias con conectividad M-N. Cuando la interrelación recursiva binaria tiene conectividad M-N se origina una nueva relación, la cual tiene como clave primaria los atributos que forman la clave primaria de la entidad originaria, pero dos veces, ya que hay que modelizar el hecho de que la única entidad que interviene en la conceptualización prevista interrelaciona con ella misma (y no con otra distinta).

Hay que modificar convenientemente los nombres de estos atributos que están presentes dos golpes en la nueva relación que no coincidan, y respetar así las directrices del modelo relacional.

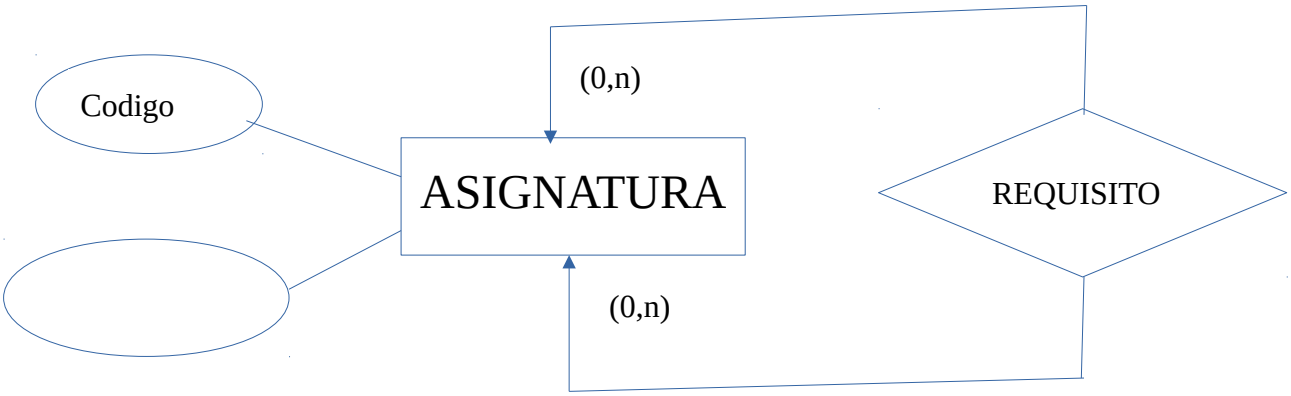
Ejemplo de transformación de interrelación recursiva binaria con conectividad M-N

El diagrama ER de la figura 13 se traduce al modelo relacional de la siguiente forma:

ASIGNATURA (Código, Descripcion)

REQUISITO (CodigoAsignatura, CodigoRequisito) DONDE {CodigoAsignatura} REFERENCIA ASIGNATURA y {CodigoRequisito} REFERENCIA ASIGNATURA

Figura 13.



c) n-arias. Se origina una nueva relación, la clave primaria de la que se construye de manera diferente en función de la conectividad:

Cuando la conexión de todas las entidades se produce con cardinalidad N, la clave primaria de la nueva relación se compone de todos los atributos que forman parte de las claves primarias de todas las entidades interrelacionadas (n).

Cuando la conexión de una o más de las entidades se produce con cardinalidad 1, la clave primaria de la nueva relación se compone de todos los atributos que forman las claves primarias de todas las entidades interrelacionadas excepto una (n-1). La entidad que no incorpora su clave primaria a la de la nueva relación debe estar forzosamente conectada con un 1.

Ejemplo de transformación de interrelación recursiva n-aria

El diagrama ER de la figura 14 se traduce al modelo relacional de la siguiente forma:

ALUMNO (DNI, Nombre, Apellidos)

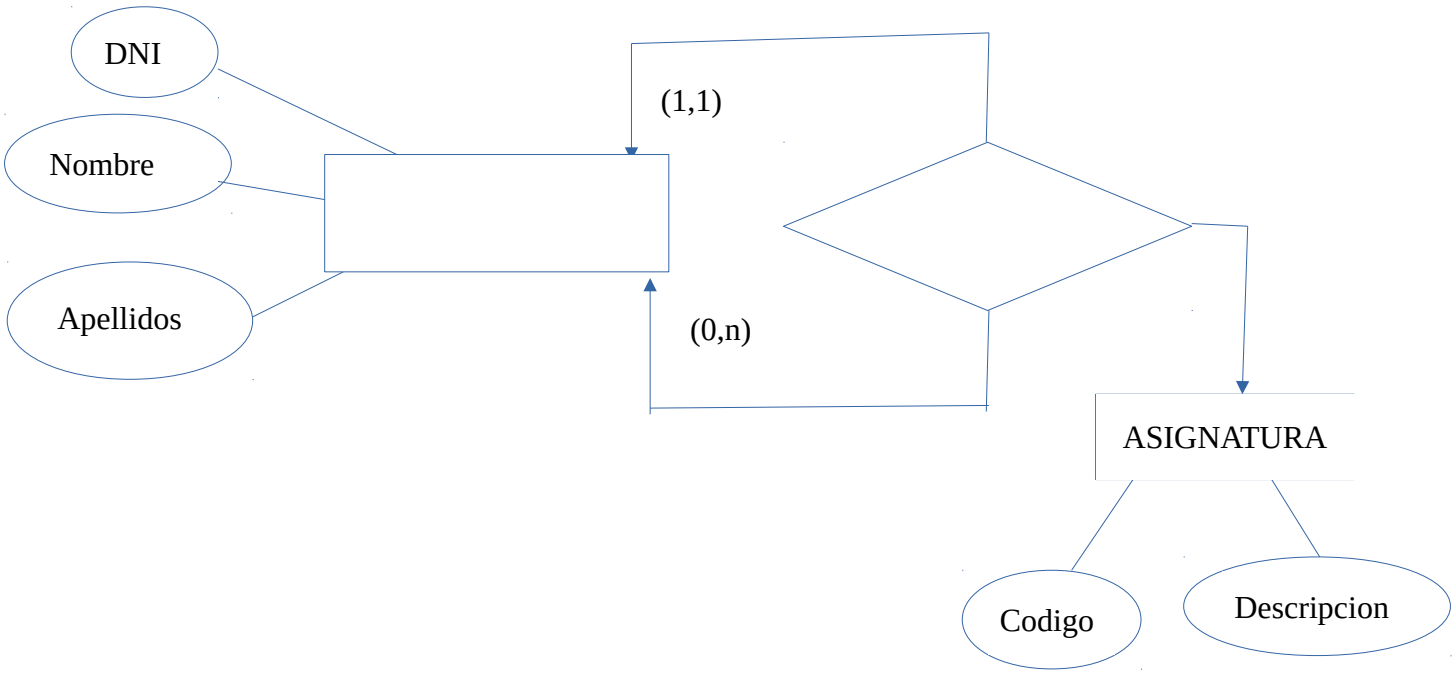
ASIGNATURA (Código, Descripcion)

DELEGADO (DNIAlumno, CodigoAsignatura, DNIDelegado) DONDE {DNIAlumno} REFERENCIA ALUMNO, {CodigoAsignatura} REFERENCIA ASIGNATURA y {DNIDelegado} REFERENCIA ALUMNO

Fijémonos que hemos incorporado a la clave primaria de la nueva relación los atributos que forman las claves primarias de las dos entidades conectadas con cardinalidad N, es decir, ASIGNATURA y ALUMNO, pero desde la posición de los alumnos que no son delegados.

De este modo, se modeliza el hecho de que cada alumno tiene un delegado para cada asignatura, y que el delegado de cada asignatura representa una pluralidad de alumnos.

Figura 14.



1.3.3 Entidades débiles

Como las entidades débiles siempre están situadas en el lado N de una interrelación 1-N que les sirve para completar la identificación inequívoca de sus instancias, la relación derivada de la entidad débil debe incorporar a su clave primaria los atributos que forman la clave primaria de la entidad de la que son tributarias. Los atributos mencionados constituyen, simultáneamente, una clave foránea que hace referencia a la entidad de la que dependen.

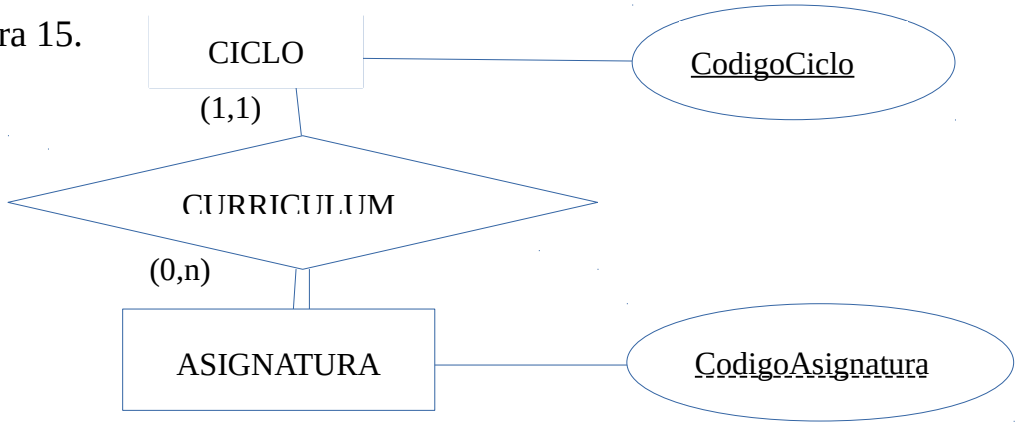
Ejemplo de transformación de entidad débil

El diagrama ER de la figura 15 se traduce al modelo relacional de la siguiente forma:

CICLO (CodigoCiclo)

ASIGNATURA (CodigoCiclo, CodigoAsignatura) DONDE {CodigoCiclo} REFERENCIA CICLO

Figura 15.



1.3.4 Generalización y especialización

En estos casos, tanto la entidad superclase como las entidades de tipo subclase se transforman en nuevas relaciones.

La relación derivada de la superclase hereda de esta la clave primaria. Además, se encarga de almacenar los atributos comunes a toda la especialización o generalización.

Las relaciones derivadas de las entidades de tipo subclase también tienen, como clave primaria, la clave de la entidad superclase, que al mismo tiempo actúa como clave foránea, al referenciar la entidad derivada de la superclase.

Ejemplo de transformación de generalización o especialización

La figura 16 muestra un encadenamiento de generalizaciones o especializaciones. Si traducimos a un modelo relacional obtenemos el siguiente resultado:

PERSONA (DNI, Nombre, Apellidos)

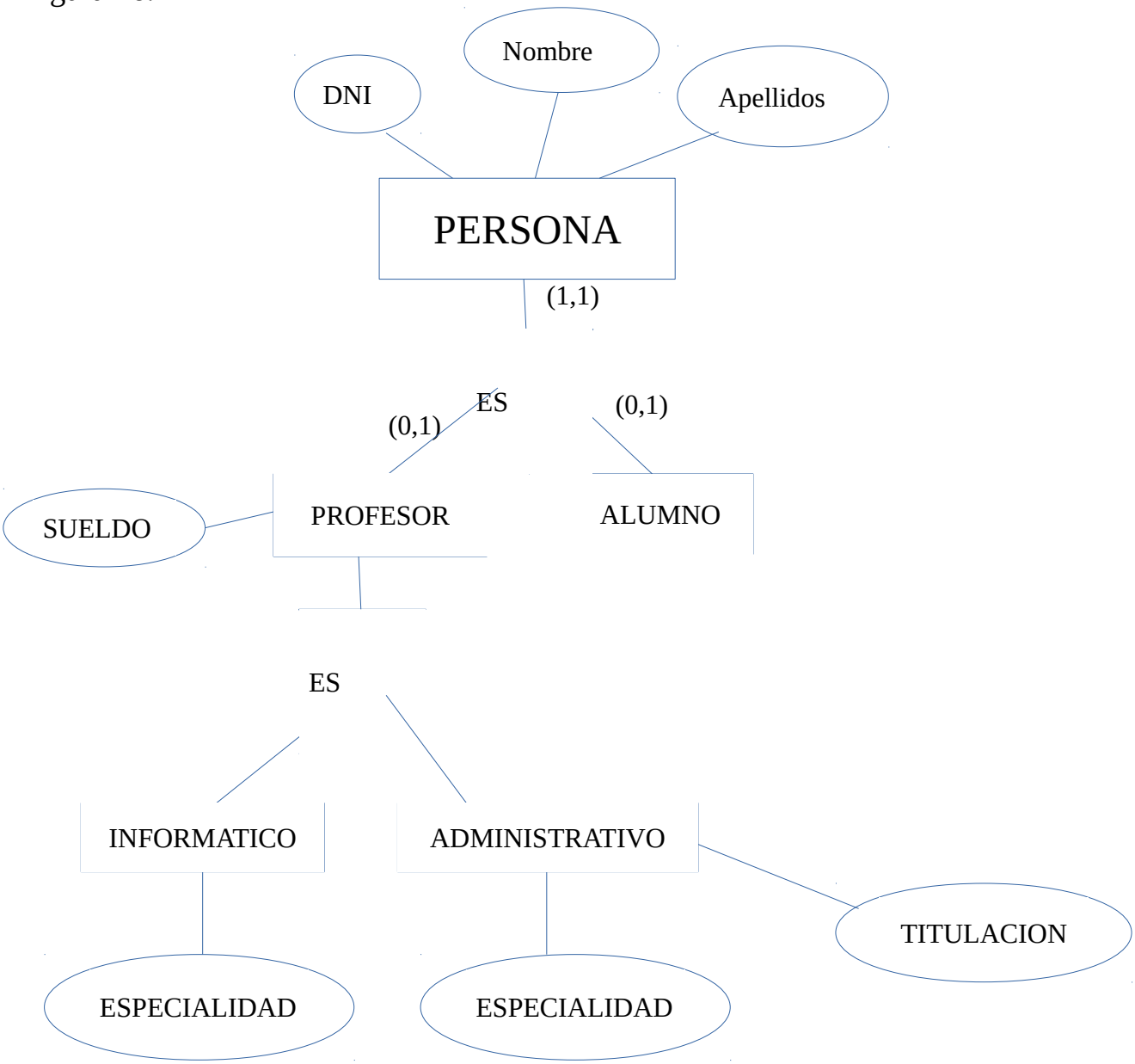
PROFESOR (DNI, Sueldo) DONDE {DNI} REFERENCIA PERSONA

ALUMNO (DNI) DONDE {DNI} REFERENCIA PERSONA

INFORMATICO (DNI, Especialidad) DONDE {DNI} REFERENCIA PROFESOR

ADMINISTRATIVO (DNI, Titulación, Especialidad) DONDE {DNI} REFERENCIA PROFESOR

Figura 16.



2. Normalización

El diseño de una base de datos puede ser una tarea extremadamente compleja. Hay diferentes metodologías que permiten abordar el problema de encontrar el esquema relacional que represente mejor la realidad que se quiere modelizar.

Conocemos el modelo Entidad-Relación para establecer modelos para cualquier realidad, se obtiene, como resultado, el diagrama Entidad-Relación. También conocemos el proceso de traducción de un diagrama Entidad-Relación a un esquema relacional.

Por lo tanto, si para llegar al esquema relacional que debe modelizar la realidad hemos seguido el camino que consiste en, primero, efectuar el diagrama ER para

luego efectuar la traducción al modelo relacional, y el diagrama ER era correcto, habremos obtenido un esquema relacional del todo correcto.

Este sería el camino aconsejable. Pero no siempre es así y nos encontramos diseños efectuados directamente en el esquema relacional. Existen diferentes causas que lo provocan:

• De entrada, el modelo Entidad-Relación es posterior al modelo relacional y, por tanto, hay bases de datos que fueron formuladas directamente en la terminología relacional. No había ninguna otra opción.

• Hay diseñadores que "no quieren perder el tiempo" en un modelo ER y diseñan directamente en el modelo relacional. Qué error más grande.

• A veces, se debe modificar la base de datos debido a nuevas necesidades, y el diseño se efectúa directamente sobre ésta en lugar de analizarse y realizarse sobre el modelo Entidad-Relación para luego transferir los cambios al esquema relacional. Qué error más grande!

Fíjense que damos un apoyo absoluto al hecho de utilizar el modelo Entidad-Relación para obtener posteriormente el modelo relacional. Un buen diseño en el modelo ER suele proporcionar una base de datos relacional bien diseñada, algo que no ocurrirá si el diseño Entidad-Relación incorpora errores. Por otra parte, si no ha habido el diseño Entidad-Relación previo, hay más posibilidades de tener una base de datos relacional mal diseñada.

La teoría de la normalización es un método que permite asegurar si un diseño relacional (tanto si proviene de la traducción de un diagrama Entidad-Relación como si se ha efectuado directamente en el modelo relacional) es más o menos correcto.

En general, los malos diseños pueden originar las siguientes situaciones:

• Repetición de la información

• Imposibilidad de representar cierta información:
    - Anomalías en las inserciones
    - Anomalías en las modificaciones
    - Anomalías en los borrados

Un buen diseño debe lograr lo siguiente:

• Almacenar toda la información necesaria con el mínimo de información redundante.

• Mantener el mínimo de vínculos entre las relaciones de la base de datos a fin

de facilitar su utilización.

• Mejorar la consultabilitat de los datos almacenados.

• Minimizar los problemas de actualización (altas, bajas y modificaciones) que pueden surgir al tener que actualizar simultáneamente datos de diferentes relaciones.

El método que propone la teoría de la normalización para determinar si un diseño relacional es correcto consiste en evaluar el diseño de todas las relaciones (tablas) para ver en qué grado de normalidad se encuentra cada una y, así, poder decidir si el diseño ya es correcto o si hay que refinarlo.

La teoría de la normalización define las formas normales como indicadores para evaluar el grado de normalidad de las relaciones, y se dice que una relación está en una forma normal determinada cuando satisface un conjunto determinado de condiciones.

Hay diferentes grados de normalidad y, por tanto, de formas normales, las cuales cumplen la relación de inclusión, que debe interpretarse en el sentido que a medida que aumenta el nivel de la forma normal, la relación debe cumplir un conjunto de condiciones más restrictivo y, por tanto, continúa verificando las condiciones de las formas normales de nivel inferior.

Así pues, el objetivo debería ser conseguir un esquema relacional en el que todas las relaciones tuvieran el grado máximo de normalidad, es decir, en que todas se encontraran en la quinta forma normal (5FN).

El proceso de normalización para conseguir que una relación que se encuentra en una forma normal X pase a estar en una forma normal Y superior a X consiste siempre en la descomposición o subdivisión de la relación original (Forma normal X) en dos o más relaciones que verifiquen el nivel de forma normal Y.

Por tanto, el proceso de normalización aumenta el número de relaciones presentes en la base de datos. Con ello, seguro que se consigue una disminución de redundancias y una disminución de las anomalías en los problemas de actualización de la información, pero, en cambio, se penalizan las consultas, ya que su ejecución tendrá que ir a buscar la información en muchas tablas relacionadas entre ellas.

Así, pues, hay que encontrar un equilibrio, ya veces puede ser conveniente renunciar al nivel máximo de normalización (5FN) y, por tanto, permitir una cierta redundancia en los esquemas con el fin de aliviar los costes de las consultas. En estas situaciones, se habla de un proceso de desnormalización.

Nuestro objetivo final es conocer las condiciones que deben cumplir las relaciones para alcanzar cada uno de los niveles de forma normal, y el proceso para dividir las

relaciones en nuevas relaciones que verifiquen las condiciones deseadas. Para conseguirlo, debemos conocer los conceptos de relación universal y dependencia funcional.

2.1 La relación universal

Al efectuar directamente el diseño relacional de una base de datos, el diseñador se encuentra con un conjunto de conceptos que traduce en atributos, los cuales, por su significado, agrupará en una o más relaciones.

Llamamos relación universal la relación consistente en el agrupamiento de los atributos correspondientes a todos los conceptos que constituyen una base de datos relacional.

Evidentemente, el diseño relacional de una base de datos basado en la relación universal suele ser del todo incorrecto, y hace necesario aplicar un proceso de normalización con el fin de ir dividiendo la relación en otras relaciones de forma que alcancen grados de normalidad mejores, es decir, cumplan las restricciones correspondientes a las formas normales más elevadas.

Muy pocas veces se parte de la relación universal. La experiencia de los diseñadores provoca que, de entrada, ya se piense en relaciones que alcanzan un cierto grado de normalidad.

2.2 Dependencias funcionales

Las definiciones de las diferentes formas normales, es decir, el conjunto de condiciones que las definen, se basan en el concepto de dependencia funcional.

Dados dos atributos (o conjuntos de atributos) A y B de una relación R, diremos que B depende funcionalmente de A si para cada valor de A existe uno, y sólo uno, valor de B asociado con él. También diremos que A implica B. Lo simbolizaremos por A B.

El concepto de dependencia funcional establece vínculos entre atributos o conjunto de atributos de una misma relación.

Dados dos atributos (o conjuntos de atributos) A y B de una relación R, diremos que B tiene una dependencia funcional completa o total de A si B depende funcionalmente de A pero no depende funcionalmente de ningún subconjunto de A.

Es muy conveniente representar las dependencias funcionales de una relación mediante un esquema de dependencias funcionales.

Dado un atributo o conjunto de atributos A de una relación, diremos que A es un determinante de la relación si hay algún otro atributo o conjunto de atributos B que tiene dependencia funcional total de A.

Dados A, B y C atributos o conjuntos de atributos de una relación, diremos que C depende transitivamente de A a través de B si B depende funcionalmente de A, C depende funcionalmente de B, y A no depende funcionalmente de B.

2.3 Primera forma normal

Al aplicar el proceso de normalización en una relación, se empieza por comprobar si la relación está en primera forma normal y, si no es el caso, se efectúan las modificaciones oportunas para conseguirlo.

Una relación está en primera forma normal (1FN) si ningún atributo puede contener valores no atómicos (indivisible).

El proceso a seguir para lograr una 1FN es añadir tantas filas como sea necesario para cada uno de los diferentes valores del campo o campos que tengan valores no atómicos.

De hecho, la restricción que persigue la 1FN forma parte de la definición del modelo relacional y, por tanto, toda relación, por definición, debe estar en 1FN. Es decir, esta forma normal es redundante con la definición del modelo relacional y no habría que considerarla. Se mantiene, sin embargo, para asegurar que las relaciones diseñadas tienen un punto de partida correcto.

En general, las relaciones en 1FN pueden tener mucha información redundante. No debe preocuparnos, ya que la solución radica en las formas normales de nivel superior.

2.4 Preservación de información y dependencias en la normalización

En el proceso de normalización de una relación R se aplican procesos de descomposición para conseguir relaciones R1, R2, ..., Rn que verifiquen un nivel de normalización superior al de la relación R. La descomposición consiste en efectuar proyecciones de la relación R sobre atributos que verifican ciertas condiciones, lo que da lugar a la aparición de R1, R2, ..., Rn.

Hay que garantizar que la descomposición de R en R1, R2, ..., Rn preserve la información existente, es decir, que el natural-join R1 * R2 * ... * Rn proporcione exactamente la misma información que tenía la relación R original, tanto en intensión (cantidad de atributos) como en extensión (cantidad de filas).

Del mismo modo habría que garantizar la conservación de las dependencias, es decir, el conjunto de dependencias asociadas a la relación R original debe ser equivalente al conjunto de dependencias asociado a las relaciones R1, R2, ..., Rn.

Ya podemos adelantar que la conservación de las dependencias no se puede garantizar en todos los procesos de normalización.

2.5 Segunda forma normal

La segunda forma normal persigue la eliminación de los problemas motivados por la presencia de dependencias funcionales no totales de los atributos que no forman parte de la clave primaria respecto a la clave primaria.

Una relación está en segunda forma normal (2FN) si está en 1FN y todo atributo que no pertenece a la llave tiene dependencia funcional total de la clave.

El proceso a seguir para lograr una 2FN es dividir la relación (Conservando la información y las dependencias) en tantas relaciones como sea necesario de manera que cada relación verifique que sus atributos no-clave tienen dependencia funcional total de la clave.

El esquema de dependencias funcionales ayuda a ver las relaciones que han de aparecer.

2.6 Tercera forma normal

La tercera forma normal persigue la eliminación de los problemas motivados por la presencia de dependencias transitivas de los atributos que no forman parte de la clave primaria, respecto de la clave primaria.

Una relación está en tercera forma normal (3FN) si está en 2FN y ningún atributo que no pertenece a la clave depende transitivamente de la clave.

El proceso a seguir para lograr una 3FN es dividir la relación (Conservando la información y las dependencias) en nuevas relaciones más simples, de forma que cada relación verifique que ninguno de sus atributos no clave depende transitivamente de la clave.

2.7 Forma normal de Boyce-Codd

Una relación está en la forma normal de Boyce-Codd (FNBC) si está en 2FN y todos sus determinantes son claves candidatas.

Se verifica que toda relación en FNBC está en 3FN pero no al revés.

El proceso a seguir para lograr una FNBC es apartar de la relación los atributos que dependen de los determinantes que no son claves candidatas, y formar nuevas relaciones que recogen los atributos apartados y que preservan la información inicial.

En las roturas efectuados sobre una relación no normalizada para alcanzar relaciones 2FN y 3FN, hay que efectuar la división de manera que se preserven la información y las dependencias funcionales, hecho siempre posible en el paso a 2FN y 3FN. En el paso a FNBC, también siempre es posible efectuar la división manteniendo la información, pero no siempre es posible el mantenimiento de las dependencias funcionales.

Debido a la pérdida de dependencias funcionales, que no siempre se produce, a menudo no se normaliza a FNBC y se trabaja con relaciones en 3FN.

2.8 Cuarta forma normal

Dados A y B atributos o conjuntos de atributos de una relación, diremos que B tiene una dependencia multivalente de A si un valor de A puede determinar un conjunto de valores de B. Lo simbolizaremos con la notación A - >> B.

Las dependencias multivalentes no son dependencias funcionales. En cambio, sin embargo, una dependencia funcional se puede llegar a considerar una dependencia multivalente en que por cada valor del implicando hay un único valor del implicado.

Los problemas provocados por las dependencias funcionales han causado la definición de la 1FN, 2FN, 3FN y FNBC. La redundancia provocada por las dependencias multivalentes nos llevan a definir la 4FN.

Una relación se encuentra en cuarta forma normal (4FN) si está en 3FN y el implicado de toda dependencia multivalente es una clave candidata.

Cuando tiene lugar una dependencia multivalente A - >> B, también existe la dependencia multivalente A - >> X - (A U B), donde X indica el conjunto de todos los atributos de la relación. Es decir, las dependencias multivalentes se presentan por parejas.

Para alcanzar la 4FN a partir de una relación R (A, B, C) que tiene una dependencia multivalente A - >> B, hay que descomponer la relación R en dos relaciones R1(A,B) y R2(A,C).

A menudo, la descomposición causada por las dependencias multivalentes efectúa antes de las descomposiciones para alcanzar los niveles 2FN, 3FN y FNBC. En esta situación, en las relaciones obtenidas, se aplicarán las comprobaciones para conseguir que estén en 2FN, 3FN y FNBC.

2.9 Quinta forma normal

Hay lo que se llama dependencias mutuas entre los atributos de la relación. Las dependencias mutuas provocan que la descomposición de la relación en otras relaciones (proyecciones del original) no verifique que su natural-join coincide con la relación original.

Diremos que una relación R descompuesta en relaciones R1, R2, ..., Rn satisface una dependencia de reunión, también llamada dependencia de proyección-join, respecto a R1, R2, ..., Rn únicamente si R es igual al natural-join de R1, R2,..., Rn. La notaremos como DR * (R1, R2, ..., Rn).

Diremos que una relación está en quinta forma normal (5FN), también llamada forma normal proyección-join (FNPJ), si está en 4FN y toda dependencia de reunión es consecuencia de claves candidatas.

Una relación 4FN que no sea 5FN debido a una dependencia de reunión se puede descomponer, sin pérdida de información, en las relaciones sobre las que se define la dependencia de reunión, las cuales están en 5FN.

2.10 Desnormalización.

Puede parecer extraño plantearse la desnormalización de una base de datos, luego de argumentar sustancialmente la importancia que tiene disponer de bases de datos normalizadas, pero en algunos casos una desnormalización controlada puede ser muy útil y, incluso, deseable.

La desnormalización se puede definir como la introducción de redundancias de forma controlada en una bases de datos, a fin de hacer más eficientes algunos procesos que de otro modo harían que globalmente el rendimiento del sistema resultara poco óptimo.

Aunque para un sistema dado, se podrían prever ciertas modificaciones sobre la BD para mejorar el rendimiento una vez en funcionamiento, el más típico es detectar estas necesidades a posteriori. Así pues, este tipo de modificaciones sobre la base de datos se llevan a cabo, habitualmente, después de haber observado ciertas ineficiencias en algunas operaciones habituales sobre una base de datos.

A continuación se describen algunas situaciones, a modo de ejemplo, donde puede ser útil la desnormalización:

• En BD donde hay consultas intensivas sobre datos de una tabla que tienen referenciadas otras tablas donde se almacenan descripciones, en algún caso puede ser útil mantener la descripción en la misma mesa original, a fin de hacer más rápida la consulta mencionada. Pensamos, por ejemplo, en una tabla de direcciones de

empleados que tiene el código de la provincia y que hace referencia a otra tabla que tiene la descripción de la provincia. Cada vez que sea necesario consultar la provincia de los empleados deberá procesar la operación de *join* de las dos tablas. En cambio, si se dispone del nombre de la provincia en la tabla de direcciones, se evitará esta operación que resulta costosa.

• Añadir campos que son calculables, como el total de una factura, puede reducir, también, el tiempo de consulta de datos de esta tabla, por ejemplo.

La introducción de redundancias que desnormalizan la BD habitualmente va ligado a la implementación de ciertos mecanismos que intentan solucionar los posibles problemas que se podrían generar de estas acciones. La forma más típica de controlar los cambios para evitar inconsistencias en en estos casos es la programación de *triggers*. Así pues, se puede programar un *trigger*, por ejemplo, porque en insertar, modificar o eliminar una línea de factura, se recalcule y actualice el importe del total de la factura, para así tener este campo calculado siempre consistente.

Trigger. Es un procedimiento de BD que se ejecuta automáticamente cuando acontecen situaciones dadas. Por ejemplo, inserción de un nuevo registro en una tabla, modificación de un dato en un campo dado, de una tabla, etc.