Diseño lógico de bases de datos. Transformación a relacional a partir del modelo ER

PID_00267039

Ignasi Lorente Puchades
Manel Díaz Llobet
Jaume Sistac Planas
Elena Rodríguez González

Tiempo mínimo de dedicación recomendado: 5 horas

Universitat Oberta de Catalunya

Ignasi Lorente Puchades

Ingeniero superior en Informática por la Universitat Oberta de Catalunya (UOC). Ejerce de jefe de proyectos de aplicaciones web en el ámbito público y privado. Profesor colaborador del Grado Multimedia en la Universitat Oberta de Catalunya (UOC) y profesor asociado en la Universitat Pompeu Fabra (UPF).

Manel Díaz Llobet

Ingeniero superior en Informática por la UPC. Trabaja como profesor de desarrollo de aplicaciones, de sitios web, sistemas operativos, redes y bases de datos desde el año 2000. Ha realizado distintos proyectos de gestión para la empresa privada y ha intervenido en proyectos de inteligencia artificial en el ámbito de la gestión editorial para medios de comunicación.

Jaume Sistac Planas

Elena Rodríguez González

El encargo y la creación de este recurso de aprendizaje UOC han sido coordinados por el profesor: Àngels Rius Gavidia (2019)

Segunda edición: septiembre 2019

Av. Tibidabo, 39-43, 08035 Barcelona
Realización editorial: FUOC

Índice

Introducción..... 5

Objetivos..... 6

1. El modelo relacional..... 7
1.1. Claves de una relación 9
1.2. Claves foráneas de una relación 11

2. Transformación del modelo entidad-relación en el modelo relacional..... 13
2.1. Interrelaciones binarias 14
2.1.1. Interrelaciones binarias con conectividad 1:1 14
2.1.2. Interrelaciones binarias con conectividad 1:N 15
2.1.3. Interrelaciones binarias con conectividad N:M 15
2.2. Interrelaciones ternarias y n-arias 16
2.2.1. Interrelaciones ternarias de conectividad N:M:P 16
2.2.2. Interrelaciones ternarias de conectividad N:M:1 17
2.2.3. Interrelaciones ternarias de conectividad N:1:1 18
2.2.4. Interrelaciones ternarias de conectividad 1:1:1 19
2.2.5. Interrelaciones n-arias 21
2.3. Transformación de entidades débiles 22
2.4. Transformación de entidades asociativas 22
2.5. Resumen del proceso de transformación 23
2.6. Ejemplo de transformación de modelo conceptual (ER) en modelo relacional 24

3. Teoría de la normalización..... 27
3.1. Anomalías de diseño 28
3.1.1. Anomalías de modificación 28
3.1.2. Anomalías de borrado 29
3.1.3. Anomalías de inserción 30
3.2. El proceso de normalización 31
3.2.1. Dependencias funcionales y dependencias funcionales completas (o plenas) 31
3.2.2. Primera forma normal 35
3.2.3. Segunda forma normal 36
3.2.4. Tercera forma normal 37
3.2.5. Forma normal de Boyce-Codd 38
3.2.6. Conclusiones sobre dependencias funcionales en las formas normales 41

3.3. Aplicación de la teoría de la normalización en el diseño de bases de datos relacionales 42
3.4. Ejemplo de normalización 44

Resumen..... 50

Actividades..... 51

Glosario..... 58

Introducción

En el módulo «Diseño conceptual de bases de datos» hemos aprendido a crear un modelo conceptual de datos a partir de la descripción de un problema planteado. Hemos utilizado el modelo entidad-relación (ER) para describir los datos que son relevantes y cómo se relacionan entre ellos. En este módulo trataremos la segunda etapa del proceso de diseño de una base de datos, es decir, la del diseño lógico, y, a partir del modelo conceptual, veremos cómo obtener el lógico.

El modelo lógico permitirá describir formalmente la estructura de datos que se implementará en la etapa siguiente, la del diseño físico. Además, esta solución será independiente de la tecnología y del sistema de bases de datos que podamos utilitzar para gestionar la base de datos. Sin embargo, en esta fase habrá que elegir el modelo de datos. Nosotros elegiremos el modelo relacional porque resulta muy sencillo hacer la traducción a relacional desde el diagrama ER y permite obtener diseños de calidad al aplicar la teoría de la normalización.

En este módulo, estudiaremos los elementos que constituyen el modelo de datos relacional, en especial aquellos que son relevantes para la transformación del modelo ER a relacional, como las claves y la estructura de las relaciones. Además, veremos cómo sistematizar esta traducción y obtener así un primer modelo lógico, que podrá ser mejorado posteriormente con la teoría de la normalización con la aplicación de una serie de reglas para obtener un modelo lógico sin redundancias de datos ni anomalías de diseño.

Objetivos

En los materiales didácticos de este módulo se hallan las herramientas indispensables para alcanzar los objetivos siguientes:

1. Conocer en qué consiste el modelo relacional y cuáles son los elementos que ofrece para la estructuración y la organización de datos.

2. Conocer los mecanismos de transformación del diagrama entidad-relación (ER) a relacional utilizando los elementos propios del modelo relacional.

3. Identificar los distintos hechos semánticos de un modelo lógico y, con la aplicación de las reglas de la teoría de la normalización, obtener un modelo lógico sin anomalías de diseño.

1. El modelo relacional

En este módulo estudiaremos el modelo relacional, que es el modelo de bases de datos según el cual se organizan las bases de datos relacionales. Este modelo nos será útil para representar el modelo lógico correspondiente al modelo conceptual obtenido en la etapa anterior.

Un modelo de datos está formado por tres elementos:

1) una estructura para almacenar los datos,

2) un conjunto de operaciones para manipularlos,

3) un conjunto de reglas o restricciones inherentes al modelo.

Explicaremos en qué consiste el modelo relacional sobre la base de estos elementos.

La estructura de almacenamiento que utiliza el modelo relacional es la relación, conocida también como tabla. Cada tabla permite almacenar información sobre un concepto y cada ocurrencia o instancia concreta de este concepto (dato) se representa mediante una tupla, también llamada registro o hilera. Una tupla de una relación tiene una estructura y se compone de un conjunto de atributos o columnas definidos sobre un conjunto de datos. Así pues, la agrupación de un conjunto de tuplas con una misma estructura da lugar a una relación.

Empezaremos a estudiar, de forma más detallada y formal, la estructura de datos que soporta el modelo relacional desde el elemento más pequeño que la compone, el tipo de datos de las columnas de la tabla o relación.

Como hemos visto en el módulo «Diseño conceptual», los datos para representar son definidos sobre un tipo de datos, que determina el conjunto de valores posibles que este puede tomar.

Un **dominio** *D* es un conjunto de valores atómicos, indivisibles, que no se puede representar como combinaciones de otros valores dentro de un SGDB relacional.

Dominio definido por el usuario

Por ejemplo, se puede definir un dominio para los distintos colores que puede tener una caja de lápices de colores, los palos de una baraja de cartas, etc.

En el modelo relacional encontraremos dos tipos de dominios:

1) Dominios predefinidos, correspondientes a los tipos de datos básicos como los enteros, las cadenas de caracteres, los booleanos, las fechas, etc.

2) Dominios definidos por el usuario, correspondientes a los tipos de datos específicos de cada problema del mundo real que se quiere representar.

Es importante darse cuenta de que no se pueden confundir las relaciones (conjuntos de datos agrupados) del modelo relacional con las interrelaciones (asociaciones entre entidades) que hemos visto en el estudio del modelo conceptual.

¡Atención!

El dominio de datos de cada atributo también forma parte de la descripción de la estructura de la relación o tabla pero, para hacerlo más sencillo, la omitiremos al representar las relaciones ya sea en forma textual o tabular.

Una **relación** es la agrupación de un conjunto de datos dentro del modelo relacional, que está formada por el **esquema** o **intensión** y la **extensión**.

El esquema de una relación lo podemos entender como el patrón de los datos que queremos almacenar, mientras que la intensión hay que entenderla como los datos concretos que siguen este patrón.

Por ejemplo, si consideramos el caso de las personas, de las que queremos conocer su DNI, nombre, apellidos y la dirección de correo electrónico, las ocurrencias de persona en el modelo relacional se podrían representar así:

PERSONA			
DNI	**nombre**	**apellidos**	**correoElectronico**
00000000T	Ramón	Cosina Riera	ramon@exemple.com
11111111V	Verónica	Martí Roura	veronica@exemple.com

Intensión

Extensión

Así pues, formalmente podremos definir el **esquema de la relación** o **intensión** como el nombre de la relación R y un conjunto de atributos A_1, A_2, ... A_n y se representará como $R\{A_1, A_2, ... A_n\}$.

En el ejemplo anterior, el nombre de la relación será PERSONA y el conjunto de atributos será DNI, nombre, apellidos, correo electrónico y se representará formalmente como:

PERSONA (DNI, nombre, apellidos, correoElectronico)

La **extensión de la relación** del esquema es un conjunto de tupas t_i en el que los valores que se contienen son propios del dominio o bien toman un valor NULL y se representa como:

t_i=<v_{i1}, v_{i2}, ..., v_{in}>.

El valor NULL

Consideraremos el valor NULL como una marca que permite indicar que un valor, que es desconocido o no, toma un valor dentro del dominio.

Siguiendo el ejemplo de la relación PERSONA, una posible tupla sería:

t_1= <34901276T, Ramon, Cosina Riera, ramon@exemple.com>

Podría suceder que una persona no tenga correo electrónico, por lo que el valor que tendría asignado en la tupla sería el valor nulo:

t_1= <34901276T, Ramon, Cosina Riera, NULL>

El **grado** de una relación es el número de atributos de su esquema.

En el caso de la relación PERSONA, su grado será 4.

La **cardinalidad** de la relación es el número de tuplas que pertenecen a su extensión.

En el caso de la relación PERSONA, su cardinalidad será 2.

El hecho de que el modelo relacional esté basado en la teoría de conjuntos nos permitirá realizar una serie de operaciones sobre las relaciones que facilitará la recuperación de datos independientemente de su ubicación.

Ved también

Para saber más, se puede consultar el apartado 4 de este módulo.

Estas operaciones a menudo requerirán asociar y recuperar datos que se encuentran en distintas relaciones. Para poderlas referenciar de forma adecuada necesitaremos identificar de forma unívoca las tuplas de cada relación y, por este motivo, será necesario definir el concepto de *clave de la relación*.

1.1. Claves de una relación

Es importante poder identificar de forma unívoca cada tupla de la base de datos. La identificación de la información no solo nos servirá para recuperar únicamente un dato en particular, sino que nos permitirá relacionar los datos con independencia de su ubicación física. Este es uno de los fundamentos del modelo relacional.

Ved también

Para saber más sobre estas claves, se puede consultar el módulo «Diseño conceptual».

Debido a la importancia de las claves en el modelo relacional, será necesario establecer un conjunto de claves candidatas que nos permitan alcanzar este objetivo. En el modelo relacional introduciremos el concepto de *superclave*.

Una **superclave** de una relación efinida por el esquema $R\{A_1, A_2, ... A_n\}$ es un conjunto de atributos para los que no hay dos tuplas distintas dentro de la relación que tengan los mismos valores por los atributos de este conjunto.

Superclau

Toda relación tiene, como mínimo, una superclave que está formada por todos los atributos de su esquema.

Además, el resto de atributos del esquema de relación son funcionalmente dependientes de este conjunto de atributos.

La **dependencia funcional** entre dos atributos *a* i *b* (***a→b***) implica que, si se conoce el valor del atributo *a*, se puede inferir cual será el valor de *b* dentro de la extensión de la relación.

En una relación, las superclaves nos servirán para identificar las tuplas de una relación y, por este motivo, los otros atributos tendrán una dependencia funcional para con ellas. Además, una superclave puede tener más atributos de los estrictamente necesarios para identificar una tupla. Nuestro objetivo será el de encontrar el mínimo conjunto posible de atributos que nos permita identificar una tupla.

Una **clave candidata** de una relación es una superclave *c* de la relación en la que ninguno de sus subconjuntos no es superclave de la relación.

Por ejemplo, en la relación PERSONA que estamos considerando, una superclave podría ser la formada por el total de atributos de la relación:

PERSONA (DNI, nombre, apellidos, correoElectronico)

Sin embargo, plantearemos como claves candidatas las superclaves formadas por los atributos {DNI} i {correoElectronico}.

La **clave primaria**, CP (en inglés *PK, primary key*), de una relación es la clave candidata elegida por el diseñador durante el proceso de diseño como identificador de las tuplas de la relación.

Una *clave alternativa* es una clave candidata que no ha sido elegida como clave primaria.

Por ejemplo, en la relación PERSONA, escogeremos como clave primaria la clave candidata formada por el atributo {DNI}, por lo que la clave candidata formada por el atributo {correoElectronico} lo consideraron como una clave alternativa.

Como veremos más adelante, una técnica habitual en el diseño del modelo relacional es añadir atributos numéricos que, a pesar de no tener relación conceptual con la entidad que se quiere representar, nos permitirán identificar y manipular las tuplas fácilmente.

Por lo tanto, siempre encontraremos una superclave de la relación y, en consecuencia, siempre existirá una clave candidata. La clave primaria será una de las claves candidatas, en el caso de que haya más de una, y siempre la podremos designar.

1.2. Claves foráneas de una relación

Las entidades se asocian entre ellas mediante interrelaciones. Estas interrelaciones pueden ser de distinto grado y tener distintos tipos de conectividad.

Ved también

Para saber más sobre claves foráneas, consulte el módulo «Diseño conceptual».

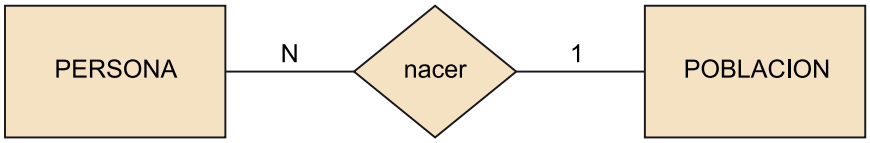
En el modelo relacional estableceremos estas asociaciones a partir del uso de claves foráneas, que nos permitirán hacer referencia a las otras relaciones a partir de sus claves primarias.

Una *clave foránea*, CF (FK en inglés), es un conjunto de atributos de una relación que permite identificar unívocamente una tupla de otra relación con la que se asocia, o de ella misma, a partir de su clave primaria.

¡Atención!

Las relaciones se podrán relacionar con otras o con ellas mismas, como sucede en el modelo conceptual, en el que hemos visto las interrelaciones recursivas en las que una entidad se asocia con ella misma.

Tomamos por ejemplo el modelado de las poblaciones donde han nacido las personas, por lo que la asociación en el modelo conceptual deriva en una relación que tendrá una conectividad 1:N.



En este caso tenemos inicialmente las relaciones POBLACION(*codigo, nombre*) y PERSONA(*DNI, nombre, apellidos, correoElectronico*), en las que se definen las claves primarias {*codigo*} y {*DNI*} para cada relación.

Si queremos establecer una asociación entre las dos relaciones, añadiremos un nuevo atributo {poblacionNacimiento} a la relación PERSONA, que haga referencia a la clave primaria de la relación POBLACION y, por lo tanto, se considerará una clave foránea.

De este modo, definiremos las relaciones de la siguiente manera:

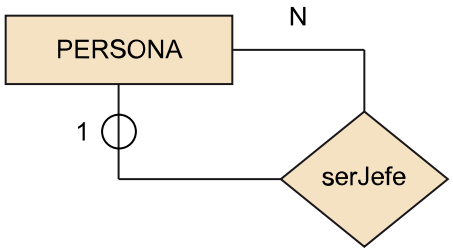
POBLACION(codigo, nombre)
PERSONA(DNI, nombre, apellidos, correoElectronico, poblacionNacimiento)
donde {poblacionNacimiento} referencia POBLACION (codigo)

En el caso de asociaciones recursivas podemos establecer, del mismo modo, un nuevo atributo a la relación que proporcione la funcionalidad de una clave foránea para con la clave primaria de la relación.

¡Atención!

Al definir una clave foránea, indicaremos explícitamente el nombre de los atributos a los que se hace referencia cuando no haya coincidencia de nombres entre los atributos de la relación a la que referencia.

Por ejemplo, el caso de una relación reflexiva entre personas de una empresa que modela la relación serCap, que permite saber de cuántas personas es hacia una persona (1 o muchas) y cuántas cabezas tiene una persona (0 si es el cargo más alto de la empresa, o 1 en todos los demás casos) lo representaríamos así:



Nota

El círculo en uno de los extremos de la interrelación indica opcionalidad.

Así, definiremos la relación PERSONA de la siguiente forma:

PERSONA(DNI, nombre, apellidos, correoElectronico, DNICap)
donde {DNICap} referencia PERSONA

Sin embargo, puede que una tupla de una relación no tenga correspondencia con una tupla de la relación a la que se refiere su clave foránea. En este caso, la clave foránea se tomará como un valor nulo.

Por ejemplo, la clave foránea podrá tomar valor nulo cuando un trabajador no tenga jefe o sea el máximo responsable, ya que no tendrá que hacer referencia a ninguna otra persona.

Es importante tener en cuenta que tanto el número de atributos de una clave foránea como el dominio de valores de cada uno de sus atributos debe coincidir exactamente con los de los atributos que forman la clave primaria de la relación a la que hace referencia. Así pues, habrá que verificar los dominios sobre los que están definidos los atributos referenciados por las claves foráneas.

Sin embargo, puede darse el caso de que algunos atributos de una clave foránea de una relación, o bien su totalidad, formen parte de su clave primaria. Este hecho se puede dar habitualmente en relaciones con entidades débiles.

Consideramos una relación que hace referencia a los viajes que ha realizado una persona, en la que una persona puede viajar a más de una población y, además, una población puede ser visitada por más de una persona.

Para representar este hecho mediante una relación (VIAJAR), necesitaremos una tupla distinta para cada pareja de ocurrencias de PERSONA-POBLACIÓN, por lo que ambos atributos se podrían considerar clave primaria. Además, los valores de estos atributos nos permitirán identificar las tuplas asociadas en las relaciones PERSONA y POBLACIÓN por separado.

Por lo tanto, podemos considerar el esquema de la relación VIAJAR como:

VIAJAR(codigoPoblacion, DNI, fecha)
donde {codigoPoblacion} referencia POBLACION(codigo)
donde {DNI} referencia PERSONA

2. Transformación del modelo entidad-relación en el modelo relacional

En los distintos ejemplos que hemos visto a lo largo de este módulo, hemos hecho referencia a la representación de los datos para resolver un problema dada su representación conceptual en términos del modelo entidad-relación. Por lo tanto, han servido para ilustrar la transformación del modelo conceptual a modelo lógico.

En este apartado veremos toda la casuística posible y daremos pautas para hacer esta transformación.

Empezamos por los elementos básicos del modelo ER: las entidades, las interrelaciones y los atributos.

Una *entidad* del modelo ER se transformará en una relación en el modelo relacional.

Una *interrelación* del modelo ER se transformará en una clave foránea o bien en una nueva relación.

En el caso de las interrelaciones habrá que tener en cuenta su grado y su conectividad.

Dado un diagrama ER, el procedimiento que utilizaremos para obtener el correspondiente modelo lógico será el siguiente:

1) Se transformarán todas las entidades en relaciones del modelo relacional.

2) Los atributos de la entidad se considerarán atributos de la relación y la clave primaria elegida será la clave primaria de la relación.

3) Las interrelaciones binarias 1:1 y 1:N darán lugar a claves foráneas, por lo que se deberán añadir nuevos atributos a la relación.

4) Las interrelaciones binarias N:M y todas las n-arias darán lugar a nuevas relaciones.

A continuación, veremos con más detalle cómo se transformarán las interrelaciones del modelo entidad-relación en el modelo relacional.

2.1. Interrelaciones binarias

En este subapartado, trataremos las relaciones binarias, que nos servirán de base a la hora de transformar el resto de interrelaciones con más entidades (n-arias).

2.1.1. Interrelaciones binarias con conectividad 1:1

Las interrelaciones binarias con conectividad 1:1 se transformarán en dos relaciones, una por cada entidad, en las que cada una de ellas tendrá una clave foránea que referencie la otra relación.

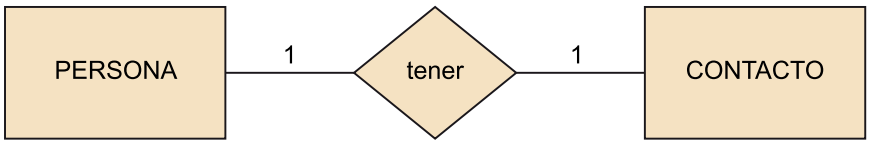
En este caso, durante el diseño, se deberá decidir cuál de las entidades es la que tendrá la clave foránea.

En este tipo de interrelaciones, hay que tener en cuenta los nuevos atributos que se añaden para indicar las claves foráneas, también serán claves candidatas de la relación.

En el caso de que una de las entidades sea opcional, es decir, que pueda participar o no en la interrelación, la clave foránea podrá tomar valores nulos.

Pensemos en una agenda en la que por cada persona guardaremos su DNI, nombre y apellidos y, además, tendrá asociada una ficha de contacto, en la que se guardará su teléfono, correo electrónico, dirección postal, etc. No hay personas ni contactos repetidos

En el modelo conceptual se representará de la siguiente forma:



De este modo, tendremos dos posibles opciones de transformación. En primer lugar, la opción cuya clave foránea forma parte de la relación CONTACTO.

PERSONA(DNI, nombre, apellidos)
CONTACTO(codigo, telefono, correoElectronico, direccion, DNI)
donde {DNI} referencia PERSONA, es único y no puede ser Null.

En la segunda opción la clave foránea forma parte de la relación PERSONA.

PERSONA(DNI, nombre, apellidos, codigo)
donde {codigo} referencia CONTACTO y es único
CONTACTO(codigo, telefono, correoElectronico, direccion)

¡Atención!

Si el atributo DNI de la relación CONTACTO no es único, podría ocurrir que una persona (un DNI) tuviese más de un contacto (código) y se pudiese dar una situación como esta:

CONTACTO (codigo, …… , DNI)

C01, ……., 52111111A

C02, ……., 52111111A

Aclaración

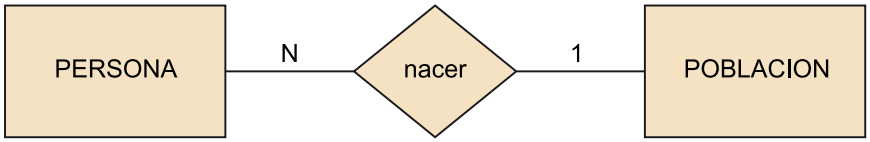
Como en el caso anterior, el atributo *codigo* de PERSONA debería ser único para garantizar que no existen dos contactos (código) de una misma PERSONA (DNI) y no null. Si pudiera ser Null, entonces la interrelación 'tener' tendría un cero, indicando opcionalidad, en uno de los extremos 1.

2.1.2. Interrelaciones binarias con conectividad 1:N

De manera parecida a las interrelaciones binarias con conectividad 1:1, las interrelaciones binarias con conectividad 1:N se transformarán en dos relaciones, una por cada entidad, en la que la relación del lado N tendrá una clave foránea que referencie la otra relación del lado 1.

Supongamos que queremos saber en qué poblaciones ha nacido cada una de las personas que constan en la base de datos.

Puesto que cada persona ha nacido en un solo lugar, en el modelo conceptual lo representaremos como una asociación entre dos entidades con conectividad 1:N, como se indica a continuación:



La transformación a relacional correcta será:

POBLACION(codigo, nombre)
PERSONA(DNI, nombre, apellidos, correoElectronico, poblacionNacimiento)
donde {poblacionNacimiento} referencia POBLACION(codigo)
donde {DNI} es único

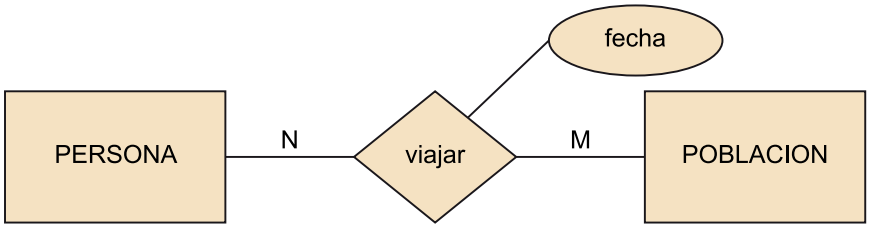
Si declaráramos la clave foránea en la relación POBLACION, nos encontraríamos que cada persona podría tener más de una ciudad de nacimiento y en una ciudad solo podría haber nacido una persona, por lo que esta solución no es correcta.

2.1.3. Interrelaciones binarias con conectividad N:M

Las interrelaciones binarias con conectividad N:M se transformarán en tres relaciones, una por cada entidad y una nueva relación formada por los atributos de la clave primaria de cada una de las dos entidades que participan en la interrelación.

En el caso de que las interrelaciones tengan atributos, estos formarán parte de la nueva relación creada.

Si consideramos los viajes hechos por una persona y entendemos que una persona puede viajar a más de un lugar (M junto a POBLACION) y que una misma población puede ser visitada por varias personas (N junto a PERSONA) su representación binaria N-M sería esta:



La transformación en modelo relacional correcta será:

POBLACION(codigo, nombre)
PERSONA(DNI, nombre, apellidos, correoElectronico)
VIAJAR(codigo, DNI, fecha)
donde {codigo} referencia POBLACION

donde {DNI} referencia PERSONA

En este caso las claves foráneas de la nueva relación para con las relaciones referidas constituirán la clave primaria de la nueva relación viajar. Será posible, también, añadir un nuevo atributo que identifique de forma unívoca las tuplas de la nueva relación, de modo que el conjunto de las claves foráneas pasarán a ser una clave alternativa.

2.2. Interrelaciones ternarias y n-arias

De forma parecida a las interrelaciones binarias de conectividad N:M, la transformación de una interrelación ternaria o n-aria dará siempre lugar a una nueva relación, que tendrá como atributos las claves primarias de las entidades interrelacionadas y todos los atributos que tenga la interrelación.

La clave primaria de la nueva relación, si no se crea un nuevo atributo identificador, será el conjunto o subconjunto de las claves foráneas de las entidades interrelacionadas según sea la conectividad de la interrelación.

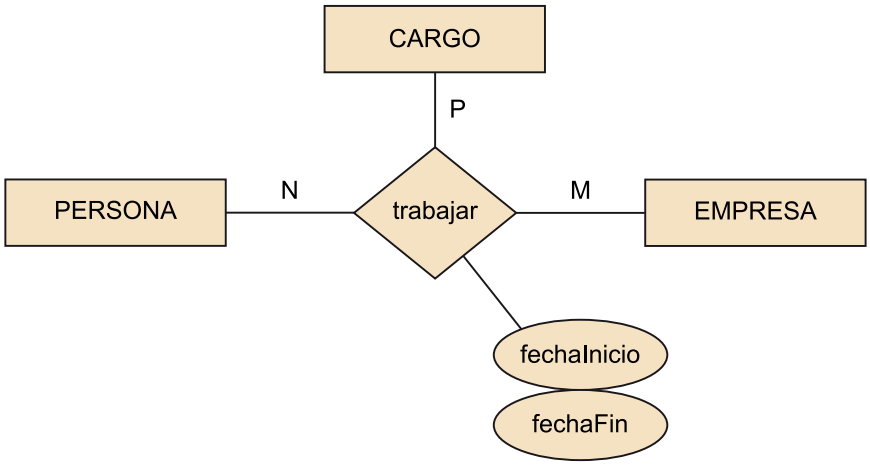
2.2.1. Interrelaciones ternarias de conectividad N:M:P

Las interrelaciones ternarias de conectividad N:M:P se transformarán en cuatro relaciones, una por cada entidad y una nueva relación formada por los atributos de la clave primaria de cada una de las tres entidades que participan en la interrelación.

En el caso de que las interrelaciones tengan atributos, estos formarán parte de la nueva relación creada.

La nueva relación tendrá como clave primaria todos los atributos que forman las claves primarias de las entidades asociadas por la interrelación.

Por ejemplo, si se quiere disponer de la información relativa a cargos ocupados por personas en empresas a lo largo del tiempo, podríamos representarlo conceptualmente mediante una asociación ternaria en la que se asocian las entidades: PERSONA, CARGO y EMPRESA, de modo que los atributos de la interrelación sean la fecha de inicio y la del final del cargo.



La transformación en modelo relacional correcta será:

PERSONA(DNI, nombre, apellidos, correoElectronico)
EMPRESA(codigo, nombre, direccion)
CARGO(codi, nombre)
TRABAJAR(DNI, codigoEmpresa, codigoCargo, fechaInicio, fechaFinal)
donde {codigoEmpresa} referencia EMPRESA(codigo)
donde {codigoCargo} referencia CARGO(codigo)
donde {DNI} referencia PERSONA

2.2.2. Interrelaciones ternarias de conectividad N:M:1

Las interrelaciones ternarias de conectividad N:M:1 se transformarán en cuatro relaciones, una por cada entidad y una nueva relación formada por los atributos de la clave primaria de cada una de las tres entidades que participan en la interrelación.

En elcaso de que las interrelaciones tengan atributos, estos formarán parte de la nueva relación creada.

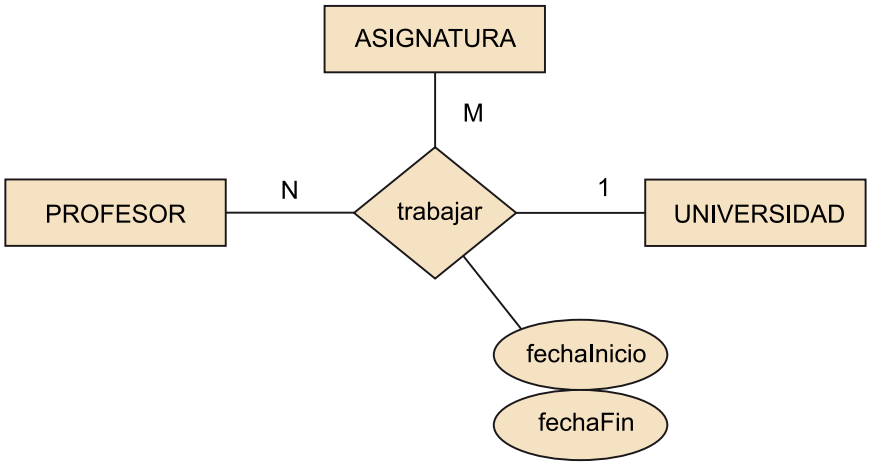
La nueva relación tendrá como clave primaria todos los atributos que forman las claves primarias de las dos entidades asociadas por la interrelación y etiquetadas con M y con *N*. Se puede ver cómo no se tienen en cuenta los atributos de la clave primaria del lado 1.

Imaginemos que queremos saber qué asignaturas imparte cada profesor en una universidad, así como el periodo de tiempo durante el cual las ha impartido o imparte.

Si partimos del hecho que un profesor solo puede impartir cada asignatura en una universidad, que en una universidad imparten asignaturas muchos profesores y que una asignatura, en un mismo centro, la pueden impartir varios profesores, en distintos momentos, podríamos representarlo así:

!Alerta!

¡Hay que fijarse que la clave primaria de la esquina 1 no forma parte de la clave primaria de la nueva relación!



La transformación en modelo relacional será:

PROFESOR(DNI, nombre, apellidos, correoElectronico)
UNIVERSIDAD(codigo, nombre, direccion)
ASIGNATURA(codigo, nombre)
TRABAJAR(DNI, codigoAsignatura, , codigoUniversidad, fechaInicio, fechaFinal)
donde {codigoUniversidad} referencia UNIVERSIDAD(codigo)
donde {codigoAsignatura} referencia ASIGNATURA(codigo)
donde {DNI} referencia PROFESOR

A diferencia del caso anterior, el atributo codigoUniversidad no será parte de la clave primaria de la nueva relación.

2.2.3. Interrelaciones ternarias de conectividad N:1:1

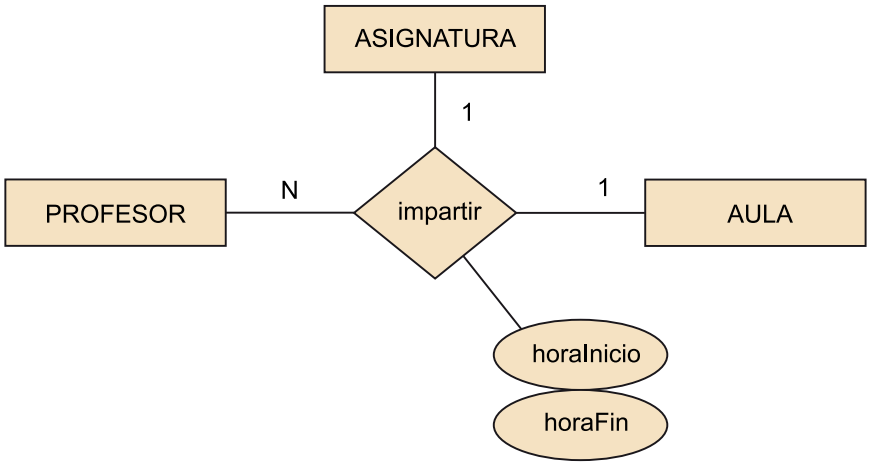
Las interrelaciones ternarias de conectividad N:1:1 se transformarán en cuatro relaciones, una por cada entidad y una nueva relación formada por los atributos de la clave primaria de cada una de las tres entidades que participan en la interrelación.

En el caso de que las interrelaciones tengan atributos, estos formarán parte de la nueva relación creada.

La nueva relación tendrá como clave primaria todos los atributos que forman la clave primaria de la entidad etiquetada *N* y los atributos que forman la clave primaria de una de las otras dos relaciones.

Así obtendremos dos claves candidatas y, por lo tanto, habrá más de una opción de transformación.

Consideramos a continuación el modelado de la interrelación que se establece cuando una asignatura puede ser impartida por más de un profesor, pero solo una vez en una misma aula. Se representaría mediante el modelo conceptual de la siguiente forma:



En este caso, encontraremos dos posibles soluciones. Si bien las relaciones que derivan de las entidades serán las mismas:

PROFESOR(DNI, nombre, apellidos, correoElectronico)
AULA(codigo, aforo)
ASIGNATURA(codigo, nombre)

Encontraremos dos posibles soluciones a la representación de la interrelación:

IMPARTIR(DNI, codigoAsignatura, codigoAula, horaInicio, horaFinal)
donde {codigoAula} referencia AULA(codigo)
donde {codigoAsignatura} referencia ASIGNATURA(codigo)
donde {DNI} referencia PROFESOR
y {DNI, codigoAula} es único

o bé

IMPARTIR(DNI, codigoAsignatura, codigoAula, horaInicio, horaFinal)
donde {codigoAula} referencia AULA(codigo)
donde {codigoAsignatura} referencia ASIGNATURA(codigo)
donde {DNI} referencia PROFESOR
y {DNI, codigoAsignatura} es único

De este modo, la clave primaria estará siempre formada por la clave foránea para con la entidad del lado N y una de las otras dos claves foráneas. Se puede observar que {DNI, codigoAsignatura} y {DNI, codigoAula}, por el hecho de ser clave primaria de IMPARTIR, identifican unívocamente las tuplas de la relación IMPARTIR.

2.2.4. Interrelaciones ternarias de conectividad 1:1:1

Las interrelaciones ternarias de conectividad 1:1:1 se transformarán en cuatro relaciones, una por cada entidad y una nueva relación formada por los atributos de la clave primaria de cada una de las tres entidades que participan en la interrelación.

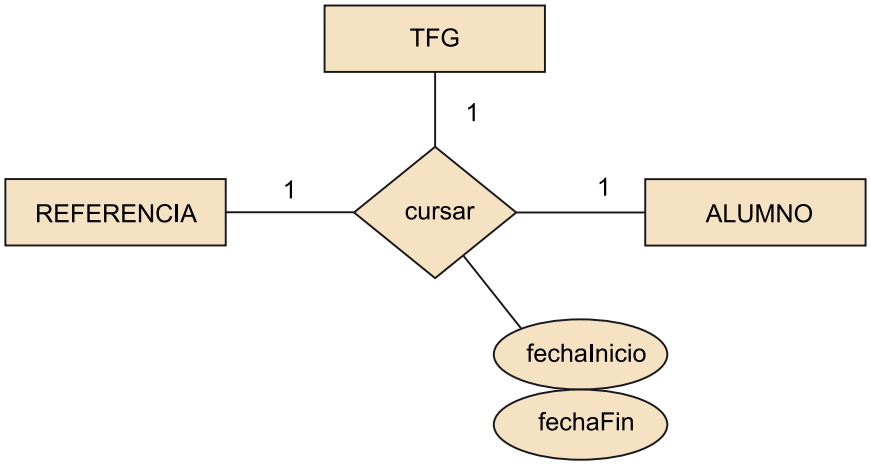
En el caso de que las interrelaciones tengan atributos, estos formarán parte de la nueva relación creada.

La nueva relación tendrá como clave primaria un único atributo de los tres que forman, respectivamente, la clave primaria de las relaciones que intervienen en ella.

De este modo obtendremos tres claves candidatas y, por lo tanto, habrá más de una opción.

Consideramos finalmente el modelado de la interrelación que hay entre el trabajo de final de grado (TFG) que realiza un alumno y la entrada bibliográfica que tendrá el trabajo en la biblioteca de la universidad, suponiendo que cada estudiante solo podrá realizar un TFG a lo largo de su vida.

En el modelo conceptual, se representará de la siguiente forma:



En este caso, encontraremos tres posibles soluciones. Si bien las relaciones que derivan de las entidades serán las mismas:

ALUMNO(DNI, nombre, apellidos)
TFG(codigo, nombre)
REFERENCIA(codigoEntrada, tematica, palabrasClave)

Encontraremos tres posibles soluciones a la representación de la interrelación:

CURSAR(codigoEntrada, codigoTFG, DNIAlumno, fechaInicio, fechaFinal)
donde {DNIAlumno} referencia ALUMNO(DNI) y es único
donde {codigoTFG} referencia TFG(codigo) y es único
donde {codigoEntrada} referencia REFERENCIA y es único

>o bien

CURSAR(codigoEntrada, codigoTFG, DNIAlumno, fechaInicio, fechaFinal)
donde {DNIAlumno} referencia ALUMNO(DNI) y es único
donde{DNIAlumno} referencia ALUMNO(DNI) y es único
donde {codigoEntrada} referencia REFERENCIA y es único

o bien

CURSAR(codigoEntrada, codigoTFG, DNIAlumno, fechaInicio, fechaFinal)
donde {DNIAlumno} referencia ALUMNO(DNI) y es único
donde {codigoTFG} referencia TFG(codigo) y es único
donde {codigoEntrada} referencia REFERENCIA y es único

De este modo, la clave primaria estará siempre formada por la combinación de dos de las claves foráneas para con dos entidades de las que forman la interrelación.

En el caso de que quisiéramos que uno de los otros atributos de la interrelación formase parte de la clave primaria tendríamos soluciones análogas combinando este atributo con alguno de los otros.

2.2.5. Interrelaciones n-arias

Las interrelaciones n-arias son una generalización de las interrelaciones ternarias y, por lo tanto, se transformarán en *n*+1 relaciones, una por cada entidad y una nueva relación formada por los atributos de la clave primaria de cada una de las tres entidades que participan en la interrelación.

En el caso de que todas las entidades estén conectadas con «muchos», la nueva relación tendrá como clave primaria todos los atributos que forman las claves primarias de las entidades asociadas por la interrelación.

En el caso de que una o más entidades estén conectadas con «uno», la nueva relación tendrá como clave primaria todos los atributos que forman las claves primarias de las entidades asociadas por la interrelación de los lados «muchos» y todas menos una de las entidades de los lados «uno».

¡Atención!

Así que, en las interrelaciones N:M:1 hemos visto que la clave primaria está formada por 2+1 atributos y en las relaciones 1:1:1 la clave primaria está formada por 0+1 atributos.

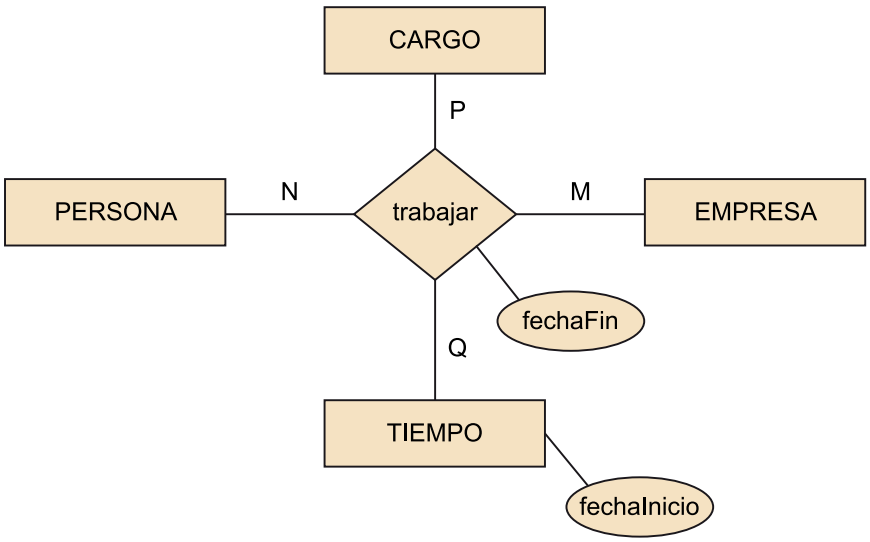
Por otra parte, como hemos visto en el módulo «Diseño conceptual», las interrelaciones de grado superior a 3 a menudo pueden facilitar el mantenimiento de un histórico de valores que tome, a lo largo del tiempo, las instancias de las entidades propias de un mundo que queremos representar.

Esta será una necesidad que aparecerá habitualmente en el diseño de las bases de datos y por ello es importante saber representarla en el modelo relacional.

Por ejemplo, para poder mantener el histórico de cargos ocupados por personas que trabajan en una empresa, deberíamos conocer el intervalo de tiempo en el que han ocupado cada cargo.

Por este motivo, deberemos tener la fecha de inicio del cargo, que siempre estará informada porque, si no, no existiría la interrelación entre las entidades, y la fecha de fin, que podrá ser desconocida o tener valor nulo, si la persona actualmente aún ocupa el cargo en la empresa.

En el modelo conceptual, se representará de la siguiente forma:



En este caso todas las entidades estarán conectadas con «muchos», de modo que la clave primaria de la relación «trabajar» estará formada por todos los atributos que formen todas las claves primarias de las entidades que asocia la interrelación «trabajar».

Las relaciones derivadas de la transformación de las entidades serán:

PERSONA(DNI, nombre, apellidos)
CARGO(codigo, nombre)
EMPRESA(codigo, nombre, CIF)
TIEMPO(codigo, fechaInicio)

Y la relación derivada de la transformación de la interrelación será:

TRABAJAR(DNI, codigoCargo, codigoEmpresa, codigoTiempo, fechaFinal)
donde {DNI} referencia PERSONA
donde {codigoCargo} referencia CARGO(codigo)
donde {codigoEmpresa} referencia EMPRESA(codigo)
donde {codigoTiempo} referencia TIEMPO(codigo)
donde {fechaFinal} puede ser nulo

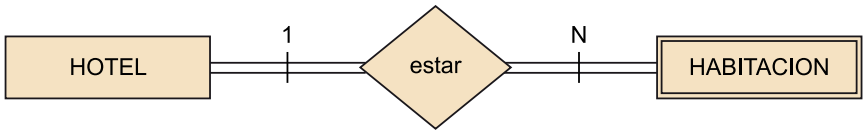
2.3. Transformación de entidades débiles

Como sabemos, las entidades débiles tienen la particularidad de que no tienen un atributo que las identifique unívocamente, sino que dependen de otra entidad.

Este tipo de entidades se transformarán en el modelo relacional, como si se tratara de una entidad cualquiera, pero su clave primaria tendrá como uno de sus atributos, la clave foránea para con su relación principal.

Consideremos, por ejemplo, las habitaciones de un hotel que pertenece a una cadena hotelera. Como que el número de las habitaciones se podrá repetir en hoteles diferentes, este número no servirá para identificarlas de forma unívoca, siendo necesario saber en qué hotel estará situada la habitación.

En el modelo conceptual, se representará de la siguiente forma:



La transformación en modelo relacional será:

HOTEL(codigo, nombre)
HABITACION(numero, codigoHotel)
donde {codigoHotel} referencia HOTEL(codigo)

Como vemos, la clave primaria de la relación HABITACION está formada por los atributos *número* y *codigoHotel* pero, sin embargo, su clave foránea para con la relación HOTEL es el atributo *codigoHotel*.

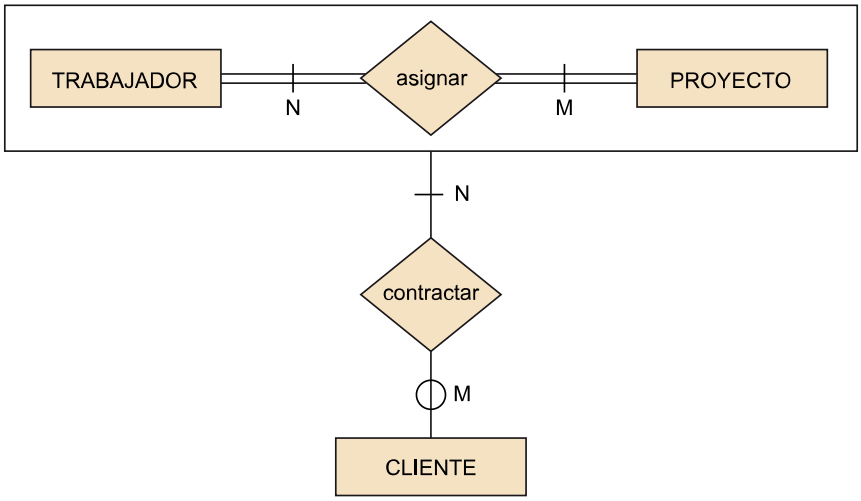
2.4. Transformación de entidades asociativas

Las entidades asociativas vendrán dadas al considerar una interrelación entre entidades como si fuese una entidad.

La transformación de este modelo conceptual se realizará en dos pasos, modelando en primer lugar la entidad asociativa como si fuese una interrelación binaria y, en segundo lugar, el resto de entidades e interrelaciones que intervienen en ella.

Pensemos en el caso de una empresa en la que los trabajadores están asignados a distintos proyectos y estos proyectos pueden ser de uso interno o contratados por distintos clientes.

En el modelo conceptual, se representará de la siguiente forma:



La transformación en modelo relacional será:

TRABAJADOR(DNI, nombre, ...)
PROYECTO(codigo, nombre, ...)
ASIGNAR(DNI, codigoProyecto)
donde {DNI} referencia TRABAJADOR
donde {codigoProyecto} referencia PROYECTO(codigo)
CLIENTE(codigo, nombre, ...)
CONTRACTAR(codigoCliente, DNI, codigoProyecto, ...)
donde {codigoCliente} referencia CLIENTE(codigo)
donde {DNI, codigoProyecto} referencia ASIGNAR

2.5. Resumen del proceso de transformación

A continuació es mostra una taula de resum de les accions que s'han de realitzar a l'hora de transformar un model ER en un model relacional.

Elemento del modelo ER	Transformación al modelo relacional
Entidad	Relación
Interrelación 1:1	Clave foránea
Interrelación 1:N	Clave foránea
Interrelación M:N	Nueva relación
Interrelación n-aria	Nueva relación
Interrelación recursiva	Como en las interrelaciones no recursivas: • Clave foránea para binarias 1:1 y 1:N • Nueva relación para binarias M:N y n-arias

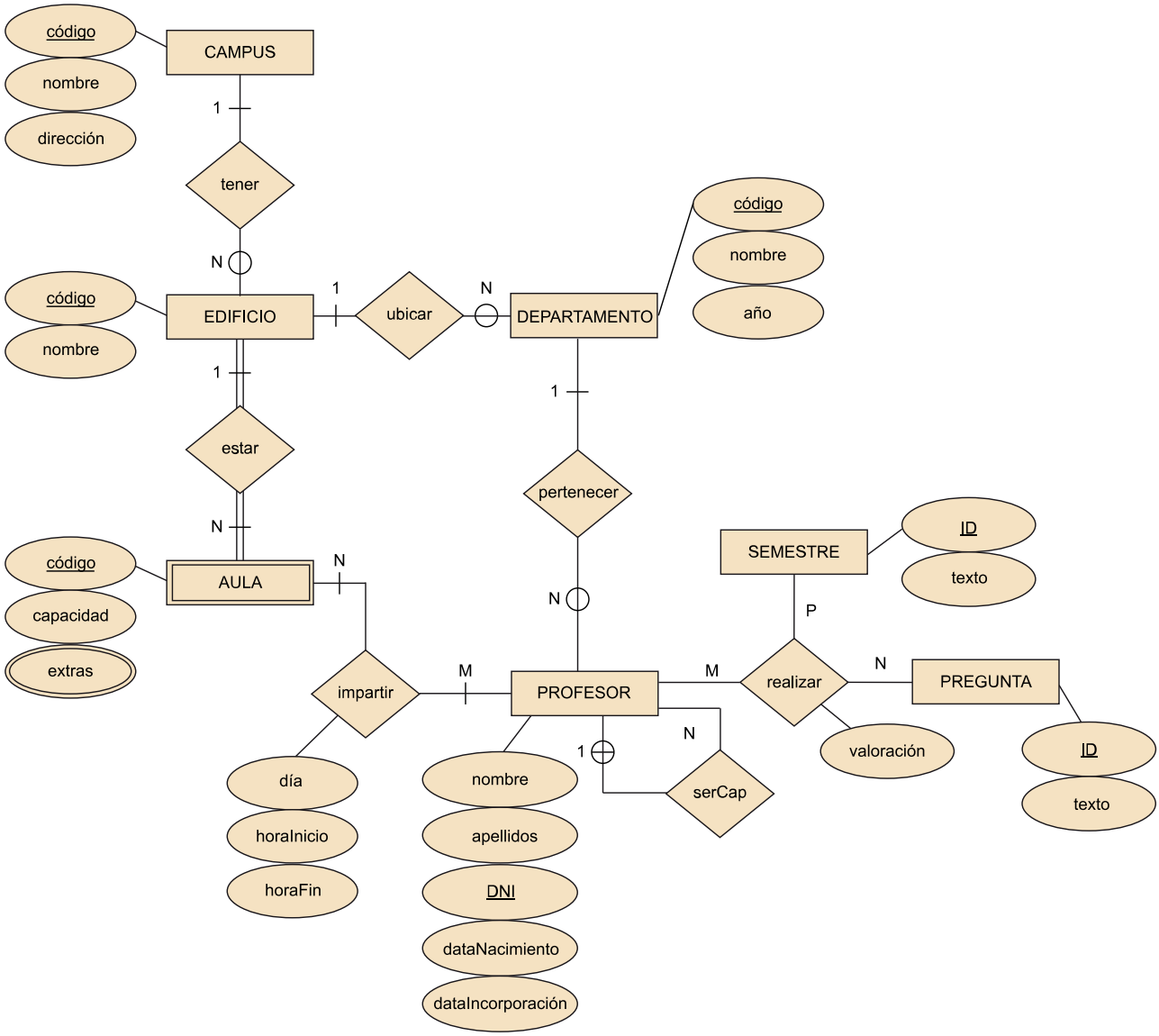
Elemento del modelo ER	Transformación al modelo relacional
Entidad débil	La clave foránea de la interrelación identificadora forma parte de la clave primaria
Generalización/especialización	• Nueva relación para la entidad superclase • Nueva relación para cada una de las entidades subclase
Entidad asociativa	La transformación de la interrelación que la origina es a la vez su transformación

Es importante hacer notar que el modelo lógico relacional no tiene tanta riqueza semántica como el modelo conceptual y que, por lo tanto, a veces las restricciones modeladas en el modelo conceptual que no son representables en el lógico relacional deben ser tratadas en el diseño físico a nivel de programación.

2.6. Ejemplo de transformación de modelo conceptual (ER) en modelo relacional

Una universidad quiere implementar una aplicación que les permita gestionar las actividades que realizan sus profesores, de modo que se pueda tener un control de su docencia en los distintos campus en los que se divide. Por este motivo, será necesario definir una base de datos que permita mantener y tratar esta información.

Después de revisar sus requerimientos, se ha diseñado el modelo conceptual representado en el siguiente diagrama ER:



En primer lugar, identificaremos las relaciones provenientes de las entidades fuertes, identificando sus atributos y las claves primarias:

CAMPUS(codigo, nombre, dirección)
EDIFICIO(codigo, nombre)
DEPARTAMENTO(codigo, nombre, año)
PROFESOR(nombre, apellidos, DNI, fechaNacimiento, fechaIncorporacion)
SEMESTRE(id, texto)
PREGUNTA(id, texto)

En segundo lugar, transformaremos la entidad débil AULA que dependerá de la entidad EDIFICIO:

AULA(codigo, codigoEdificio, capacidad, extras)
donde {codigoEdificio} referencia EDIFICIO(codigo)

A continuación, transformaremos las interrelaciones binarias de conectividad 1:N, que modificarán la definición de las relaciones ya existentes:

EDIFICIO(codigo, nombre, dirección, codigoCampus)
donde {codigoCampus} referencia CAMPUS(codigo)
DEPARTAMENTO(codigo, nombre, año, codigoEdificio)
donde {codigoEdificio} referencia EDIFICIO(codigo)
PROFESOR(nombre, apellidos, DNI, fechaNacimiento, fechaIncorporacion, codigo-Departamento, DNICap)
donde {codigoDepartamento} referencia DEPARTAMENTO(codigo) y no puede ser null
donde {DNICap} referencia PROFESOR(DNI)

Acto seguido, transformaremos las interrelaciones binarias de conectividad N:M y las ternarias. En este caso, la interrelación binaria pertenece a una entidad de asociación, de modo que hará referencia a la relación correspondiente a la entidad débil.

IMPARTIR(codigoAula, codigoEdificio, profesorDNI, dia, horaInicio, horaFinal)
donde {codigoAula, codigoEdificio} referencia AULA(codigo, codigoEdificio)
donde {profesorDNI} referencia PROFESOR(DNI)
REALIZAR(profesorDNI, preguntaID, semestreID, valoracion)
donde {profesorDNI} referencia PROFESOR(DNI)
donde {preguntaID} referencia PREGUNTA(id)
donde {semestreID} referencia SEMESTRE(id)

3. Teoría de la normalización

El objetivo de la teoría de la normalización es formalizar un conjunto de ideas simples que guían un buen diseño de bases de datos relacionales clásicos.

La teoría de la normalización se fundamenta en el principio básico: toda relación debe describir un concepto semántico único.

La teoría de la normalización nos permite reconocer los casos en los que este principio no se cumple. El mecanismo que la teoría de la normalización utiliza para ello son las formas normales (FN). Una relación está en una forma normal determinada, si satisface un conjunto de restricciones determinadas que son propias de esta forma normal. La violación de estas restricciones origina que la relación tenga las anomalías y las redundancias que hemos mencionado antes.

Nomenclatura

Abreviamos la forma normal con las siglas FN.

Hay varias formas normales, de las cuales nosotros estudiaremos las cuatro más importantes. Cada forma normal indica unas restricciones específicas que debe cumplir una relación.

Cuando una relación no cumple una forma normal, es porque viola la restricción asociada con aquella forma normal. Para conseguir que se verifique la forma normal, habrá que evitar la condición que hace que se viole la restricción en cuestión. El procedimiento que aplicaremos para conseguir que no se viole la restricción asociada a la forma normal recibe el nombre de **normalización**.

Grados de las formas normales

Si una relación está en tercera forma normal (3FN), también estará en segunda forma normal (2FN) y en primera forma normal (1FN).
Para indicar que una relación no está en 1FN, se suele utilizar el acrónimo FN2.
(NF NF, *Non First Normal Form*).

Nosotros analizaremos una a una cada forma normal y resolveremos las condiciones que hacen que se viole una forma normal determinada.

La primera forma normal (1FN) se basa en el concepto de *valor atómico*, mientras que el resto de formas normales se basan en el concepto de *dependencia*. La segunda forma normal (2FN), la tercera forma normal (3FN) y la forma normal de Boyce-Codd (FNBC) se basan en el concepto de ***dependencia funcional***. Sin embargo, hay otras formas normales que se basan en otros conceptos y que no estudiaremos.

Un modelo lógico que no está normalizado presenta una serie de anomalías perjudiciales para la buena gestión de los datos que almacena. A continuación, veremos con un ejemplo cuáles son estas anomalías y los problemas que generan.

3.1. Anomalías de diseño

Como consecuencia de un mal diseño, podemos tener relaciones que presentan un alto grado de redundancia, es decir, presentan repeticiones que son evitables. Este hecho complica su mantenimiento, dado que se producen anomalías. A continuación, analizamos estas anomalías sobre la siguiente relación de *SUMINISTROS*.

Tabla 1. Relación de *SUMINISTRO*

SUMINISTRO			
codigoProveedor	codigoArticulo	cantidad	ciudadProveedor
P1	A1	100	Reus
P1	A2	150	Reus
P2	A1	200	Vic
P2	A2	250	Vic
P3	A2	100	Vic

Fuente: elaboración propia.

La clave primaria de la relación del esquema *SUMINISTRO* (*codigoProveedor, codigoArticulo, cantidad, ciudadProveedor*) está formada por los atributos *codigoProveedor* y *codigoArticulo*. Esta relación representa los artículos que suministran los distintos proveedores y en qué cantidad. Además, también registra la ciudad de cada proveedor.

3.1.1. Anomalías de modificación

Si un proveedor cambia de ciudad, habrá que poner la nueva ciudad del proveedor a todas las tuplas que hagan referencia al proveedor en cuestión, si no queremos que la base de datos sea inconsistente. Por ejemplo, si el proveedor con *codiProv = P1* cambia de ciudad y se va a Salou, tendremos que modificar dos tuplas; si este proveedor suministrara mil artículos distintos, deberíamos modificar mil tuplas, etc. La situación ideal sería que solo tuviéramos que hacer una modificación.

Tabla 2. Modificaciones

SUMINISTRO			
codigoProveedor	codigoArticulo	cantidad	ciudadProveedor
P1	A1	100	Salou
P1	A2	150	Salou
P2	A1	200	Vic

Fuente: elaboración propia.

SUMINISTRO			
P2	A2	250	Vic
P3	A2	100	Vic

Fuente: elaboración propia.

Las anomalías de modificación obligan a modificar todas las tuplas que guardan un hecho determinado, si este cambia.

3.1.2. Anomalías de borrado

Si un proveedor que solo suministra un producto (por ejemplo, el proveedor con *codiProv = P3*) deja de suministrarlo, será necesario borrar la tupla de la relación *SUMINISTRO*. Como consecuencia, habremos perdido los datos personales del proveedor en cuestión, en este caso, el código de proveedor y la ciudad. Este proveedor nos ha suministrado 100 unidades del artículo *A2* que tenemos en el almacén. Si se elimina el proveedor, también se elimina esta información del *stock*, que puede provocar descuadres en el almacén.

Si tenemos la tabla SUMINISTRO:

SUMINISTRO			
codigoProveedor	codigoArticulo	cantidad	ciudadProveedor
P1	A1	100	Salou
P1	A2	150	Salou
P2	A1	200	Vic
P2	A2	250	Vic
~~P3~~	~~A2~~	~~100~~	~~Vic~~

Fuente: elaboración propia.

Al dejar de comprar el articulo A2, borraríamos la fila correspondiente a este artículo y en consecuencia, se perderían los datos del proveedor P3. Quedando la tabla SUMINISTRO como sigue:

Tabla 3. Anomalías de borrado

SUMINISTRO			
codigoProveedor	codigoArticulo	cantidad	ciudadProveedor
P1	A1	100	Reus

Fuente: elaboración propia.

SUMINISTRO			
P1	A2	150	Reus
P2	A1	200	Vic
P2	A2	250	Vic

Fuente: elaboración propia.

Sin querer, debido a las anomalías de borrado, se pueden perder hechos elementales.

3.1.3. Anomalías de inserción

No podemos almacenar datos personales de un proveedor con, por ejemplo, *codiProv = P4* y de la ciudad de Mollet, si no sabemos los artículos que suministra, dado que habría que añadir una tupla a la relación de *SUMINISTRO* con valor *null* para el atributo *codigoArticulo*, y se violaría la regla de integridad de entidad de la clave primaria:

Integridad de entidad de la clave primaria

Hay que recordar que la regla de integridad de entidad de la clave primaria dispone que los atributos de la clave primaria de una relación no pueden tener valor *null*.

Tabla 4. Anomalías de inserción

SUMINISTRO			
codigoProveedor	codigoArticulo	cantidad	ciudadProveedor
P1	A1	100	Reus
P1	A2	150	Reus
P2	A1	200	Vic
P2	A2	250	Vic
P3	A2	100	Vic
P4	NULL	NULL	Mollet

Fuente: elaboración propia.

Las anomalías de inserción consisten en no poder insertar hechos elementales de forma independiente.

El origen de todas estas anomalías es que la relación *SUMINISTRO* describe dos hechos elementales, del mundo real, distintos: por un lado, los artículos que suministra cada proveedor; y por el otro, el proveedor en sí mismo. Además, estos hechos son completamente independientes entre sí, dado que los artículos concretos que suministra cada proveedor no mantienen ninguna relación directa con el hecho de que el proveedor sea, por ejemplo, de una ciudad o

de otra, y viceversa. En todo caso, entre estos dos hechos hay una relación indirecta porque afectan a un mismo individuo del mundo real, es decir, el mismo proveedor.

En conclusión, toda relación que no representa un concepto (o hecho elemental) único del mundo real está sujeta a presentar redundancias, anomalías de mantenimiento e inconsistencias potenciales.

Este es el caso de nuestra relación *SUMINISTRO*.

Este será el objetivo del proceso de normalización; separar los distintos conceptos o hechos elementales del mundo real que están en una misma relación, de modo que cada concepto o hecho elemental acabe estando en una relación distinta, aplicando las reglas de las distintas formas normales.

3.2. El proceso de normalización

3.2.1. Dependencias funcionales y dependencias funcionales completas (o plenas)

Como ya sabemos, sobre una base de datos podemos imponer restricciones o reglas de integridad. Estas restricciones regulan los posibles estados válidos, es decir, los estados íntegros de una base de datos. A continuación, presentamos un nuevo tipo de restricción, las dependencias funcionales.

Intentaremos explicar el significado de la definición de dependencia funcional mediante el ejemplo de relación que se muestra a continuación:

Tabla 5. *SUMINISTRO* (*codigoProveedor*, *codigoArticulo*, *cantidad, ciudadProveedor*)

SUMINISTRO			
codigoProveedor	codigoArticulo	cantidad	ciudadProveedor
P1	A1	100	Reus
P1	A2	150	Reus
P2	A1	200	Vic
P2	A2	250	Vic
P3	A2	100	Vic

Fuente: elaboración propia.

Algunos ejemplos de dependencias funcionales que existen en esta relación serían:

{*codigoProveedor, codigoArticulo*} → *cantidad*
{*codigoProveedor, codigoArticulo*} → *ciudadProveedor*

Dado que {*codigoProveedor, codigoArticulo*} es la clave primaria de la relación *SUMINISTRO*, estamos seguros de que cada valor de la pareja {*codigoProveedor, codigoArticulo*} determina de forma unívoca los valores de los atributos *cantidad* y *ciudadProveedor*.

También sería dependencia funcional:

codigoProveedor → *ciudadProveedor*

En este caso, también podemos ver que, cada vez que el atributo *codigoProveedor* adquiere el mismo valor, el valor del atributo *ciudadProveedor* se repite.

Más concretamente:

- Para las dos primeras hileras que corresponden al proveedor P1, el valor asociado al atributo *ciudadProv* es siempre el mismo e indica que la ciudad del proveedor es Reus.

- Para las hileras 3 y 4 que corresponden al proveedor P2, el valor asociado al atributo *ciudadProv* se repite e indica que la ciudad del proveedor es Vic.

- Para la quinta hilera que corresponde al proveedor P3, el valor asociado al atributo *ciudadProv* indica que la ciudad del proveedor es Vic.

Dicho de otro modo, para conocer el valor del atributo *ciudadProveedor* solo hay que fijarse en el valor del atributo *codigoProveedor*.

El significado que tiene este hecho en el mundo real es que un proveedor está localizado siempre en la misma ciudad y que este hecho NO depende del código del artículo que provea.

Por otra parte, fijémonos en la primera dependencia funcional:

{*codigoProveedor, codigoArticulo*} → *cantidad*

En este caso, para conocer la cantidad provista de un artículo concreto, es necesario conocer tanto el código del proveedor que nos lo provee, como el código del artículo que se provee.

Si solo se tiene en cuenta el *codigoProveedor*, se puede ver que, para el proveedor con código P1, hay dos cantidades distintas, 100 y 150 (o sea, que el *codigoProveedor* solo no determina el valor del atributo *cantidad*).

Por otro lado, si nos fijamos solo en el *codigoArticulo*, se puede ver que, para el artículo con código A1, también hay dos cantidades distintas, 100 y 200 (o sea, que el *codigoArticulo* solo no determina el valor del atributo *s*).

Para determinar el valor del atributo *cantidad* de forma unívoca, es necesario conocer tanto el *codigoProveedor* como el *codigoArticulo*. Lo que es lo mismo, es necesario conocer el valor de toda la clave primaria.

Estos tipos de dependencias funcionales se denominan «dependencias funcionales completas» o «dependencias funcionales plenas».

En cambio, consideramos lo siguiente:

codigoArticulo → *cantidad*

Esto no sería dependencia funcional, dado que no siempre que el atributo *codigoArticulo* toma el mismo valor se repite el valor asociado al atributo *cantidad*. Por ejemplo, la hilera 1 y 3 tienen el mismo valor asociado (*A1*) para el atributo *codigoArticulo*, pero en cada caso el valor que toma el atributo *cantidad* es distinto (100 en la primera hilera y 200 en la tercera hilera, respectivamente).

> Así, un atributo (o conjunto de atributos) {Y} depende funcionalmente de otro atributo (o conjunto de atributos) {X}, expresado de otro modo {X} → {Y}; si dado un valor v_1 para {X} solo se puede tener un solo valor v_2 para {Y}, y, además, siempre que se dé v_1 a {X} deberá darse v_2 a {Y}.

En el ejemplo:

{X} = *codigoProveedor*
{Y} = *ciudadProveedor*
v_1 = «P1»
v_2 = «Reus»

Siempre que el atributo *codigoProveedor* tenga el valor «P1», el valor del atributo *ciudadProveedor* deberá ser obligatoriamente Reus.

Las conclusiones principales que podemos extraer de la definición y el ejemplo de las dependencias funcionales son estas:

a) Para identificar las dependencias funcionales de una relación, hay que analizar detenidamente el significado de los atributos que intervienen en ella.

b) La clave primaria de una relación siempre determina funcionalmente el resto de atributos de la relación. Esta conclusión se puede extender a todas las claves alternativas que la relación pueda tener.

La primera conclusión es especialmente importante. En el ejemplo, nosotros hemos deducido que el código de un proveedor determina funcionalmente su ciudad. Además, aparentemente, hemos llegado a esta conclusión por el examen de la extensión de la relación *SUMINISTRO*. Pero esto no es del todo exacto o, dicho de otro modo, no es suficiente. Examinar la extensión de una relación nos ayuda a descartar dependencias funcionales (en el ejemplo, hemos descartado que el código de los artículos determine el número de unidades que un proveedor es capaz de suministrar de este artículo), pero no nos permite asegurar rotundamente que existe dependencia funcional. Realmente, tenemos que ir más allá y preguntarnos si el comportamiento que inferimos a partir del examen de la extensión se da, o no, en la realidad.

En resumen, para deducir dependencias funcionales no necesitamos en absoluto examinar la extensión de la relación, dado que es una propiedad de intensión. En todo caso, examinar la extensión nos puede ayudar a corroborar las dependencias funcionales. Lo que necesitamos es conocer el significado de los atributos a la realidad para poder asegurar que las dependencias funcionales calculadas son las correctas.

Una vez identificadas las dependencias funcionales y las dependencias funcionales completas (o plenas), un modo de indicarlas es el siguiente:

SUMINISTROS (*codigoProveedor*, *codigoArticulo*, *cantidad*, *ciudadProveedor*)
{*codigoProveedor*, *codigoArticulo*} → *cantidad*
codigoProveedor → *ciudadProveedor*

Como {*codigoProveedor, codigoArticulo*} es la clave primaria de la relación y determinan el valor de cantidad, la dependencia funcional de *cantidad* está llena.

Como *codigoProveedor* determina el valor de *ciudadProveedor* y *codigoProveedor* no es la clave primaria de la relación, la dependencia funcional de *ciudadProveedor* NO es plena.

Cuando hacemos el diseño conceptual de una base de datos y luego encontramos el diseño lógico correspondiente, todas las dependencias funcionales que pueda haber en las distintas relaciones que forman parte del diseño lógico deben ser dependencias funcionales completas.

3.2.2. Primera forma normal

Una relación está en **primera forma normal (1FN)** si y solo si ningún atributo de la relación es en sí mismo una relación, es decir, si todo atributo de la relación es atómico, no se puede descomponer y no es un grupo repetitivo.

Información

La primera forma normal fue propuesta por Codd en el año 1970.

La relación *SUMINISTRO* que se muestra a continuación no está en 1FN, dado que los atributos *codigoArticulo* y *cantidad* no son atómicos porque son, en sí mismos, una relación, es decir que se pueden descomponer.

Tabla 6. Relación no normalizada en 1FN

SUMINISTRO			
codigoProveedor	codigoArticulo	cantidad	ciudadProveedor
P1	A1 A2	100 150	Reus
P2	A1 A2	200 250	Vic
P3	A1	200	Vic

Fuente: elaboración propia.

Para conseguir normalizar una relación en 1FN, habrá que aplicar una operación que se conoce con el nombre de *allanar*. Como consecuencia de esta operación, la relación *SUMINISTRO* quedaría de la siguiente forma:

Tabla 7. Relación normalizada en 1FN

SUMINISTRO			
codigoProveedor	codigoArticulo	cantidad	ciudadProveedor
P1	A1	100	Reus
P1	A2	150	Reus
P2	A1	200	Vic
P2	A2	250	Vic
P3	A2	100	Vic

Fuente: elaboración propia.

Es importante darse cuenta de que, cuando se allana la relación *SUMINISTRO* original, la clave primaria de la relación cambia y pasa a ser compuesta. También cabe destacar que el modelo relacional siempre garantiza que las relacio-

nes están en 1FN, dado que solo hay una estructura para representar los datos (la relación) y, además, cada dato se representa uniformemente mediante valores atómicos.

3.2.3. Segunda forma normal

Una relación está en **segunda forma normal (2FN)** si y solo si está en 1FN, y todo atributo no-clave depende funcionalmente y de forma completa de la clave primaria.

Información

La segunda forma normal fue propuesta por Codd en el año 1970.

Hay una excepción: un atributo puede depender funcionalmente de parte de la clave primaria, si este atributo es parte de una clave alternativa.

Por otra parte, también puede darse el caso de que un atributo {Z} que no forma parte de la clave primaria dependa funcionalmente de otro atributo {Y} que tampoco no forma parte de la clave primaria, pero este segundo atributo sí que tiene una dependencia funcional completa con la clave primaria {X}:

{Z} → {Y} → {X}, siendo {X} la clave primaria de la relación, y {Y} y {Z} atributos de la relación que no forman parte de la clave primaria.

En este caso, se aplica uno de los axiomas de Armstrong, llamado ***transitividad***, que dice:

Axiomas de Armstrong

Son un conjunto de reglas de inferencia que permiten obtener todas las dependencias funcionales de una relación.

Si {Z} → {Y} i {Y} → {X} entonces {Z} → {X}

En este caso se considera, por aplicación del axioma de transitividad, que el atributo {Z} también tiene una dependencia funcional plena con la clave primaria {X}.

Es importante destacar que la 2FN se basa en el concepto de «dependencia funcional completa» y, por lo tanto, esta forma normal solo se puede violar cuando la clave primaria de una relación es compuesta. En resumen, aquellas relaciones con clave primaria simple, es decir, con clave primaria formada por un único atributo siempre estarán en 2FN.

Por ejemplo, la relación de *SUMINISTRO* que se muestra a continuación no está en 2FN:

SUMINISTRO (*codigoProveedor*, *codigoArticulo*, *cantidad, ciudadProveedor*)
{*codigoProveedor, codigoArticulo*} → *cantidad*
codigoProveedor → *ciudadProveedor*

La idea que hay detrás de la 2FN es muy simple. Si un atributo no-clave (en nuestro ejemplo, *ciudadProveedor*) depende funcionalmente de una parte de la clave primaria (*codigoProveedor*, en el ejemplo) es porque representa un hecho relacionado con esta parte de la clave primaria y, por lo tanto, ambos (el atributo no-clave y los atributos de la clave del que depende) son otro concepto semántico (en nuestro caso, la ciudad de residencia del proveedor). La redundancia que hay en una relación que no está en 2FN es inmediata: los valores del atributo no-clave se repetirán para todos los valores distintos de la parte de la clave de la que no depende.

Para conseguir pasar una relación a 2FN, debemos evitar que haya dependencias funcionales no completas respecto a la clave. Por lo tanto, todos los atributos que participan en la dependencia funcional no completa deberán proyectarse en una nueva relación que corresponde al concepto semántico que representan, como se muestra a continuación:

SUMINISTRO (*codigoProveedor*, *codigoArticulo*, *cantidad*)
donde *{codigoProveedor}* } se refiere a *CIUDADES_PROVEEDOR (codigoProveedor)*
{*codigoProveedor*, *codigoArticulo*} → *cantidad*
CIUDADES_PROVEEDOR (*codiProveïdor*, *ciudad*)
codigoProveedor → *ciudadProveedor*

De este modo, los distintos conceptos semánticos se describen en diferentes relaciones, siguiendo el principio básico de la normalización.

3.2.4. Tercera forma normal

Una relación está en **tercera forma normal (3FN)** si y solo si está en 2FN y ningún atributo no-clave depende funcionalmente de ningún otro conjunto de atributos no-clave.

Información

La tercera forma normal fue propuesta por Codd en el año 1970.

La excepción aplicada a la 2FN se propaga también a la tercera forma normal. Consideremos la relación *CLIENTE* que mostramos a continuación:

CLIENTE (*codigoCliente*, *calle, número, ciudad, provincia*)
codigoCliente → *calle*
codigoCliente → *número*
codigoCliente → *ciudad* → *provincia*

Esta relación está en segunda forma normal, pero no en tercera forma normal, dado que hay una dependencia funcional *ciudad → provincia* entre dos atributos no-clave y, además, ninguno de los dos atributos es clave alternativa de la relación y, por lo tanto, la excepción no se aplica. De nuevo, el hecho de que el atributo *provincia* dependa funcionalmente del atributo *ciudad* es independiente del cliente en sí mismo. Las redundancias y las anomalías que presenta esta relación son inmediatas.

Redundancias asociadas a 3FN

Para cada cliente que esté localizado en la ciudad de Mataró, tendremos repetido el hecho de que esta ciudad pertenezca a la provincia de Barcelona.

Para normalizar una relación que viola la 3FN habrá que evitar la dependencia funcional entre atributos no-clave. Por lo tanto, los atributos que participan en la dependencia funcional deberán proyectarse en una nueva relación que corresponda al concepto semántico que representan, como se muestra a continuación:

CLIENTE (*codigoCliente, calle, número, ciudad*)
donde {*ciudad*} se refiere a *CIUDAD (ciudad)*
codigoCliente → calle
codigoCliente → número
codigoCliente → ciudad
CIUDAD (*ciudad, provincia*)
ciudad → provincia

3.2.5. Forma normal de Boyce-Codd

Ahora analizamos la siguiente relación *NOTA*:

NOTA (*dniAlumno, codigoAsignatura*, *codigoMatricula, nota*)
dniAlumno → codigoMatricula
codigoMatricula → dniAlumno
{*dniAlumno*, *codigoAsignatura*} → *nota*

La relación *NOTA* verifica la 1FN (recordemos que el modelo relacional garantiza siempre, por defecto, esta forma normal). Esta relación también está en 2FN, pese a que hay una dependencia funcional no completa de parte de la clave hacia un atributo no-clave ({*dniAlumno*} → {*codigoMatricula*}). Dado que esta dependencia involucra un atributo no-clave {*codigoMatricula*} que forma parte de una clave alternativa de la relación {*codigoMatricula, codigoAsignatura*}, se aplica la excepción. Finalmente, la relación también está en 3FN aunque hay una dependencia funcional entre atributos no-clave {*codigoMatricula*} → {*dniAlumno*}. De nuevo, dado que involucra atributos que forman parte de claves alternativas, se aplica la excepción.

Sin embargo, tal como podemos ver a continuación, la relación *NOTA* presenta redundancias y anomalías:

Tabla 8. Relación *NOTA*

NOTA			
dniAlumno	codigoAsignatura	codigoMatricula	nota
52111111A	05.001	123	A
52111111A	04.002	123	B
52222222B	05.002	215	B
52222222B	05.001	215	B
52222222B	04.002	215	C

Fuente: elaboración propia.

a) Redundancias: por cada asignatura distinta en la que esté matriculado un alumno, se repetirá su código de matrícula (por ejemplo, el alumno con DNI 52222222B tiene repetido tres veces su código de matrícula).

b) Anomalías de modificación: si queremos modificar, por ejemplo, el código de matrícula del estudiante con DNI 52222222B para que sea 312 en vez de 215, tendremos que modificar tres tuplas. Como ya sabemos, lo ideal sería modificar solo una tupla.

El origen de estas anomalías es histórico y se debe a un error de omisión. Cuando Codd enunció la 2FN y la 3FN en el año 1970, no consideró la posibilidad de que en una relación pudiera haber varias claves candidatas que fueran compuestas, y tampoco no consideró la posibilidad de que entre estas pudiera haber solapamientos. Por eso, en el año 1974, Boyce y Codd propusieron la forma normal de Boyce-Codd que soluciona las limitaciones de la 2FN y la 3FN. De hecho, a menudo se normaliza una relación directamente a la forma normal de Boyce-Codd sin pasar previamente por la 2FN y la 3FN.

Una relación está en forma normal de Boyce-Codd (FNBC)* si y solo si está en 1FN, y si todos los determinantes son una clave candidata de la relación.

*** Observación**

BCNF en inglés.

En la definición aparece un concepto nuevo: el concepto ***determinante***.

Dada una dependencia funcional {X} → {Y}, un ***determinante*** es el conjunto {X}.

Por lo tanto, si lo expresamos de una manera informal, los determinantes en una dependencia funcional son los orígenes de flecha.

Si volvemos a examinar la relación *NOTA* y buscamos todas las dependencias funcionales que hay en la relación, llegamos a la siguiente situación:

NOTA (*dniAlumno*, *codigoAsignatura*, *codigoMatricula*, *nota*)
dniAlumno → *codigoMatricula*
codigoMatricula → *dniAlumno*
{*dniAlumno*, *codigoAsignatura*} → *nota*
{*codigoMatricula*, *codigoAsignatura*} → *nota*

La tabla siguiente muestra todos los determinantes (primera columna) y todas las claves candidatas de la relación *NOTA* (segunda columna):

Tabla 9. Determinantes y claves candidatas de la relación *NOTA*

NOTA	
Determinantes	Claves candidatas
dniAlumno	dniAlumno, codigoAsignatura
codigoMatricula	codigoMatricula, codigoAsignatura
dniAlumno, codigoAsignatura	
codigoMatricula, codigoAsignatura	

Fuente: elaboración propia.

Dado que no todos los determinantes son claves candidatas de la relación *NOTA*, llegamos a la conclusión de que *NOTA* no está en FNBC. De nuevo, para normalizar la relación habrá que evitar la condición que causa la violación de la restricción. Por lo tanto, tendremos que evitar las dependencias funcionales {*dniAlumno*} → {*codigoMatricula*} y {*codigoMatricula*} → {*dniAlumno*} de la relación *NOTA*.

Esto se consigue si se proyectan los atributos involucrados en una nueva relación que corresponde al concepto semántico que representan. Ahora bien, en este caso y a diferencia de los casos anteriores, tenemos varias alternativas:

a)

NOTA (*dniAlumno*, *codigoAsignatura*, *nota*)
donde *{dniAlumno}* se refiere a *ALUMNOS {dniAlumno}*
{dniAlumno, codigoAsignatura} → *nota*

ALUMNO (*dniAlumno*, *codigoMatricula*)
dniAlumno → *codigoMatricula*
codigoMatricula → *dniAlumno*

b)

NOTA (*codigoMatricula*, *codigoAsignatura*, *nota*)
donde *{codigoMatricula}* se refiere a *ALUMNOS (codigoMatricula)*
{*codigoMatricula*, *codigoAsignatura*} → *nota*

ALUMNO (*codigoMatricula*, *dniAlumno*)
codigoMatricula → *dniAlumno*
dniAlumno → *codiMatrícula*

c)

NOTA (*dniAlumno*, *codigoAsignatura*, *nota*)
{*dniAlumno*, *codigoAsignatura*} → *nota*

ALUMNO (*codigoMatricula* , *dniAlumno*)
codigoMatricula → *dniAlumno*
dniAlumno → *codigoMatricula*

d)

NOTA (*codigoMatricula*, *codigoAsignatura*, *nota*)
{*codigoMatricula*, *codigoAsignatura*} → *nota*

ALUMNO (*dniAlumno*, *codigoMatricula*)
dniAlumno → *codigoMatricula*
codigoMatricula → *dniAlumno*

Las distintas posibilidades de proyección de atributos entre la relación *NOTA* y la nueva relación *ALUMNO* se obtienen como resultado de combinar las claves candidatas de la relación original *NOTA*. Como diseñadores, somos libres de escoger cualquiera de las opciones presentadas. Ahora bien, las más naturales son las opciones a) o b), dado que las opciones c) y d) son mixtas. Entre las opciones a) y b), desde el punto de vista de la secretaría del centro, posiblemente la opción b) sería más coherente.*

*** Observación**

Hay que pensar que hay alumnos de otros países que no tienen DNI.

Observamos que, independientemente de la opción elegida, las relaciones *NOTA* y *ALUMNO* están en FNBC, dado que todos los determinantes son claves candidatas de la relación. Esto es especialmente importante para la nueva relación *ALUMNO*.

3.2.6. Conclusiones sobre dependencias funcionales en las formas normales

Las conclusiones principales que se pueden extraer del proceso de normalización son las que se presentan a continuación:

a) Siempre es posible normalizar hasta la forma normal de Boyce-Codd.

b) El proceso de normalización no es único, aunque hay soluciones mejores, en función de la utilización posterior que se tenga que hacer de la base de datos normalizada.

c) Como consecuencia de la normalización, el modelo lógico final es mejor que el modelo lógico inicial por los siguientes motivos:

- Elimina redundancias y anomalías.
- Separa hechos semánticamente diferentes.

d) Dado un modelo lógico inicial de una base de datos, si le aplicamos un proceso de normalización, el modelo lógico final obtenido equivale siempre al modelo lógico inicial. Es decir, el proceso de normalización preserva la semántica (o significado) del modelo lógico inicial. Hay que tener en cuenta que las dependencias funcionales y las dependencias funcionales completas representan los distintos conceptos semánticos relacionados que se almacenan en las distintas relaciones de la base de datos. Es por este motivo que, en el proceso de normalización, nunca no se puede romper una dependencia funcional (completa o no). Si un atributo {Y} depende de otro atributo {X}, y el atributo {X} se debe reubicar en otra relación, forzosamente el atributo {Y} también se debe reubicar en la misma relación que {X}, para no perder el concepto semántico que indica su dependencia funcional (se verá este caso en el ejemplo expuesto más adelante).

3.3. Aplicación de la teoría de la normalización en el diseño de bases de datos relacionales

El diseño de bases de datos relacionales empieza siempre con la captura y la abstracción de los requisitos de datos de los usuarios de la organización que encargan el diseño de la base de datos. Esto implica que el diseñador debe representar, con la máxima exactitud posible, los objetos (o las entidades) del mundo real que intervienen, las propiedades (o los atributos), las interrelaciones entre entidades y, también, las reglas de integridad que deben cumplirse. Sin embargo, es importante distinguir entre un diseño de una base de datos partiendo de cero y un diseño de una base de datos que parta de diseños parciales ya existentes.

En el segundo caso, los diseños parciales se integran en un diseño único y nuevo para la organización. La utilización de estos diseños es frecuente en organizaciones que quieren centralizar distintos sistemas de información que hasta ahora funcionaban de forma independiente, y también en el diseño de grandes bases de datos; entonces, en el diseño de la base de datos intervienen varios equipos que hacen diseños parciales con el objetivo de obtener un diseño global. En ambos casos, es necesaria una etapa de integración de vistas para obtener un esquema conceptual global.

En cambio, en el primer caso, el diseño de la base de datos se realiza a partir de cero y se puede tratar teóricamente desde dos puntos de vista extremos:

1) Se puede partir de una única relación, también conocida como *relación universal*, que contiene todos los atributos de interés para una organización concreta, y también todas las dependencias funcionales entre estos atributos. A partir de estos elementos, será necesario aplicar de manera reiterada un proceso de normalización con el objetivo de obtener un esquema relacional (o conjunto de esquemas de relaciones) que caracterizará la base de datos de la organización.

2) Se parte de un conjunto de atributos de interés para una determinada organización y de las respectivas dependencias funcionales, y luego se construye un esquema relacional que caracteriza la base de datos de la organización.

El primer enfoque sobre el diseño teórico de la base de datos da lugar a los métodos de análisis o descomposición, también conocidos como **métodos descendientes**,* mientras que el segundo da lugar a los métodos de síntesis o composición, también conocidos como **métodos ascendentes**.*

*** Observación**

Métodos descendientes, en inglés top-down.

Métodos ascendientes, en inglés bottom-up.

Ambos enfoques utilizan métodos que podríamos denominar *puros*. Sin embargo, en la práctica, se utilizan métodos mixtos que combinan análisis y síntesis.

En los **métodos mixtos**, el diseñador deberá elaborar el diseño conceptual de la base de datos mediante un modelo semántico de datos.

Después de la elaboración del modelo conceptual mediante los métodos mixtos, habrá que traducir el modelo obtenido a un diseño equivalente, pero ahora expresado en el modelo relacional. Como resultado de este proceso, obtenemos lo que se conoce como *diseño lógico de la base de datos*. Finalmente, habrá que implementar el diseño lógico obtenido sobre el SGBD relacional concreto que emplea la organización y, así, dar lugar al diseño físico de la base de datos.

¿Por qué es más natural la utilización de los métodos mixtos? La respuesta es sencilla. Cuando un diseñador detecta un atributo, también detecta a qué entidades pertenece el atributo en cuestión. O al revés: cuando se detecta una entidad, también se pueden detectar fácilmente gran parte de sus atributos. Esto significa que implícitamente se identifican algunas de las relaciones que forman parte del diseño lógico de la base de datos, precisamente aquellas que corresponden a entidades en el diseño conceptual.

Explicitar las interrelaciones entre las entidades es una tarea que está contemplada en cualquier modelo semántico de datos. En el diseño lógico posterior, estas interrelaciones darán lugar a relaciones nuevas o atributos nuevos que deben cumplir unas reglas de integridad determinadas.

Así como la normalización siempre es necesaria en los diseños elaborados con los métodos de análisis 1 (basados en la relación universal), también es conveniente aplicarla al final del proceso de los métodos mixtos para el diseño de bases de datos relacionales clásicas. De este modo eliminaremos redundancias y evitaremos las anomalías de diseño que hemos estudiado al inicio de la teoría de la normalización. Sin embargo, conviene tener presente que el seguimiento correcto de cualquier metodología de diseño conduce a un diseño lógico compuesto por relaciones que verifican, como mínimo, la tercera forma normal.

En definitiva, la normalización resulta útil como comprobación de aspectos semánticos difíciles, pero pocas veces es realmente necesaria, dado que un diseñador mínimamente experimentado es capaz de distinguir hechos semánticamente independientes y, por lo tanto, ya tiene en cuenta el principio básico en el que se fundamenta la teoría de la normalización.

En casos muy particulares, puede interesar que una relación concreta no verifique una forma normal determinada. Esta técnica se conoce en el diseño físico de bases de datos como ***desnormalización***. Pensamos que la normalización minimiza la existencia de redundancias y prevé las anomalías de diseño de una relación para mejorar la integridad de los datos; con ello, sin embargo, se paga el precio de una degradación en el rendimiento porque los datos que antes estaban en una única relación, después de la normalización, están dispersos en varias relaciones.

La normalización, pues, tiende a penalizar la recuperación o la consulta de los datos porque exige operaciones de combinación para recuperar datos que originalmente estaban en una única relación. Cuando los atributos de una relación que representan otro concepto semántico no se actualizan con frecuencia, puede ser conveniente utilizar la *desnormalización*.

3.4. Ejemplo de normalización

Tenemos la siguiente intención de relación que almacena información sobre visitas de pacientes a médicos:

VISITA (numColegiado, numTS, datosPaciente, datosHospital, datosVisita)

Para dar contexto a la información que contiene la relación, diremos que: un paciente que es identificado por su número de la tarjeta sanitaria (*numTS*) se visita con un médico, identificado por su número de colegiado (*numColegiado*) en una fecha concreta (*fechaVisita*) y en un hospital concreto (*datosHospital*) de una ciudad (*datosHospital*). Solo nos interesa conocer la última fecha de visita de cada paciente, no el historial de visitas. Por eso, para cada par (*numColegiado, numTS*), hay una sola fecha de visita y un solo hospital, y este par de atributos determina la clave primaria. Además, sabemos que un médico puede visitar varios pacientes en diferentes hospitales.

De cada paciente se quiere saber su nombre completo (*datosPaciente*) –entendido como nombre y apellidos– y su DNI (*datosPaciente*). La fecha de la visita tendrá el formato dd/mm/aaaa.

La extensión de la relación *VISITA* en este momento es:

Tabla 10. Extensión de la relación VISITA

numColegiado	**numTS**	**datosPaciente**	**datosHospital**	**fechaVisita**
C001	TS001	52111111A, Joan Pi Dot	del Mar, Barcelona	03/03/2019
C001	TS002	52222222B, Blai Morell Pi	Arnau de V., Lleida	05/03/2019
C002	TS001	Joan Pi Dot, 52111111A	del Mar, Barcelona	08/03/2019

Fuente: elaboración propia.

¿Está en 1FN?

Como se puede ver con los datos informados, los atributos *datosPaciente* y *datosHospital* no son atómicos, ya que estas columnas almacenan varias informaciones distintas a las que, además, interesa acceder de forma individual, según el contexto.

Normalización a 1FN

Para normalizar a 1FN hay que atomizar los atributos, esto quiere decir que solo pueden tener un valor. Esto hace que se generen nuevos atributos para que cada atributo almacene una única información.

VISITA (numColegiado, numTS, nombrePaciente, nombreHospital, fechaVisita, dniPaciente, ciudadHospital)

La nueva extensión de la relación *VISITA* en este momento es:

Tabla 11. Nueva extensión de la relación

numColegiado	numTS	nombrePaciente	nombreHospital	fechaVisita	dniPaciente	ciudadHospital
C001	TS001	Joan Pi Dot	del Mar	03/03/2019	52111111A	Barcelona
C001	TS002	Blai Morell Pi	Arnau de V.	05/03/2019	52222222B	Lleida
C002	TS001	Joan Pi Dot	del Mar	08/03/2019	52111111A	Barcelona

Fuente: elaboración propia.

¿Está en 2FN?

Para saber si la relación está en segunda forma normal, hay que comprobar que todos los atributos que no forman parte de la clave primaria tienen una dependencia funcional plena (DFP) con toda la clave primaria, excepto las claves alternativas, que se consideran normalizadas en segunda forma normal.

Veamos cuáles son las dependencias:

VISITA (numColegiado, numTS, nombrePaciente, nombreHospital, fechaVisita, dniPaciente, ciudadHospital)

- *nombrePaciente* tiene dependencia funcional (DF) con numTS.
- *nombreHospital* tiene DFP con toda la clave primaria, porque un médico puede visitar distintos pacientes en diferentes hospitales.
- *fechaVisita* tiene DFP con toda la clave primaria.
- *dniPaciente* es clave alternativa (se podría sustituir a la clave primaria el *numTS* por *dniPaciente*, y la clave primaria de la relación *VISITA* también identificaría correctamente todas las ocurrencias).
- *ciudadHospital* tiene DF con *nombreHospital*, pero como *nombreHospital* tiene DFP con la clave primaria, por la regla de transitividad, se considera que *ciudadHospital* también tiene una DFP con la clave primaria.

Por lo tanto, tenemos estas dependencias:

- *numTS* → *nombrePaciente*
- {*numColegiado, numTS*} → *nombreHospital*
- {*numColegiado, numTS*} → *fechaVisita*
- *dniPaciente* → *numTS*
- *numTS* → *dniPaciente*
- {*numColegiado, numTS*} → *nombreHospital* → *ciudadHospital*

Así, rápidamente podemos ver que esta relación NO está en 2FN porque hay atributos que no son clave alternativa, que no dependen de toda la clave, tales como *nombrePaciente*.

Normalización a 2FN

Para normalizar a 2FN, separaremos relaciones de modo que tengamos dependencias funcionales plenas en cada tabla. Así pues, por un lado, separaremos los datos relativos a visitas que dependen de toda la clave y crearemos nuevas tablas para el resto. En este caso, tendremos:

VISITA (numColegiado, numTS, nombreHospital, fechaVisita, dniPaciente, ciudadHospital)
donde {*numTS*} se refiere a *PACIENTE (numTS)*
PACIENTE (numTS, nombrePaciente)

¿Está en 3FN?

Para saber si la relación está en tercera forma normal, hay que comprobar que ningún atributo no-clave dependa de ningún otro atributo no-clave o de ningún conjunto de atributos no-clave. Se aplica también la excepción de las claves alternativas, como la segunda forma normal.

VISITA (numColegiado, numTS, nombreHospital, fechaVisita, dniPaciente, ciudadHospital)

- *nombreHospital* tiene DFP con toda la clave primaria, porque un médico puede visitar varios pacientes en diferentes hospitales.
- *fechaVisita* tiene DFP con toda la clave primaria.
- *dniPaciente* es clave alternativa (se podría sustituir a la clave primaria el *numTS* por el *dniPaciente*, y la clave primaria de la relación *VISITA* también identificaría correctamente todas las ocurrencias).
- *ciudadHospital* tiene una DF de *nombreHospital*, ya que, dado un nombre de hospital, podemos saber con certeza en qué ciudad está.

Así pues, tenemos las siguientes dependencias:

- {*numColegiado, numTS*} → *nombreHospital*
- {*numColegiado, numTS*} → *fechaVisita*
- *dniPaciente* → *numTS*
- *numTS* → *dniPaciente*
- *nombreHospital* → *ciudadHospital*

Vemos, pues, que esta relación NO está en tercera forma normal, dado que *ciudadHospital* tiene una DF con un atributo que no forma parte de la clave primaria, *nombreHospital*.

Normalización a 3FN

Se crea una relación para guardar la información relativa a los hospitales.

VISITA (numColegiado, numTS, nombreHospital, fechaVisita, dniPaciente)
donde {*numTS*} se refiere a *PACIENTE (numTS)*
donde {*nombreHospital*} se refiere a *HOSPITAL (nombreHospital)*
PACIENT (numTS, nombrePaciente)
HOSPITAL (nombreHospital, ciudadHospital)

¿Está en BCNF?

Puesto que la relación está en 1FN, habrá que comprobar que todos los determinantes también son claves candidatas de la relación.

Como dice la teoría, un determinante es el atributo origen de una dependencia funcional. Hacemos dos listas, una de determinantes y otra de claves candidatas. Si las listas coinciden, la relación está en FNBC. Si no, habrá que normalizar.

VISITA (numColegiado, numTS, nombreHospital, fechaVisita, DNIPaciente)

- *dniPaciente* determina el valor de *numTS*.
- *numTS* determina el valor de *dniPaciente*.
- {*numColegiado*, *numTS*} determina los valores de *nombreHospital* y *fechaVisita*.
- {*numColegiado*, *dniPaciente*} determina los valores de *nombreHospital* y *fechaVisita*.

Tenemos estas dependencias:

- *dniPaciente* → *numTS*
- *numTS* → *dniPaciente*
- {*numColegiado, numTS*} → *nombreHospital*
- {*numColegiado, numTS*} → *fechaVisita*
- {*numColegiado, dniPaciente*} → *nombreHospital*
- {*numColegiado, dniPaciente*} → *fechaVisita*

Y, por lo tanto, los determinantes son los conjuntos de atributos que hay a la izquierda de la dependencia y las claves candidatas son: {*numColegiado, numTS*} y {*numColegiado, dniPaciente*}.

Haciendo dos listas, tenemos:

Tabla 12. Determinantes y claves candidatas en la relación *VISITA*

Determinantes	Claves candidatas
numTS	*{numColegiado, numTS}*
dniPaciente	*{numColegiado, dniPaciente}*
{numColegiado, numTS}	

Fuente: elaboración propia.

Determinantes	Claves candidatas
{numColegiado, dniPaciente}	

Fuente: elaboración propia.

Así, podemos observar que la lista de determinantes NO es igual a la lista de claves candidatas, lo que permite deducir que la relación *VISITA* NO está en FNBC.

Normalización a FNBC

Para normalizar hasta FNBC, hay que eliminar de la relación *VISITA* el atributo *dniPaciente*, que hace que la relación no esté en FNBC.

Puesto que ya se dispone de una relación para almacenar los datos de los pacientes, solo hay que mover de lugar el atributo *dniPaciente*, de *VISITA* hacia *PACIENTE*.

VISITA (numColegiado, numTS, nombreHospital, fechaVisita)
donde {*numTS*} se refiere a *PACIENTE (numTS)*
y {*nombreHospital*} se refiere a *HOSPITAL (nombreHospital)*
PACIENTE (numTS, nombrePaciente, DNIPaciente)
HOSPITAL (nombreHospital, ciudadHospital)

Resumen

En este módulo se ha introducido el modelo lógico de las bases de datos considerando como modelo de datos el modelo relacional.

Como partíamos del modelo conceptual representado en notación ER, hemos visto las pautas que hay que seguir en la transformación del modelo conceptual, representado en un diagrama ER, en el modelo relacional utilizando los elementos propios de este modelo.

Hemos visto el concepto de clave primaria, candidata y foránea; como definir una relación, sus componentes y las reglas del modelo para asegurar la consistencia de los datos. El resultado es un modelo lógico que podrá ser mejorado con la teoría de la normalización e implementado en la etapa de diseño físico.

Finalmente, también hemos visto cómo aplicar las reglas de la teoría de la normalización para obtener un modelo lógico libre de anomalías de diseño, donde cada hecho semántico corresponde a una relación distinta.

Actividades

Ejercicio 1. Una discográfica quiere poder gestionar la información de las diferentes canciones de las que tiene adquiridos los derechos y de los músicos que la compusieron o interpretaron en los discos que ha editado.

De cada canción, se quiere guardar su nombre y el número de veces que ha sido editada en distintos discos. Además, se querrá guardar información sobre quién es el autor (teniendo en cuenta que una canción puede haber sido creada por más de un músico).

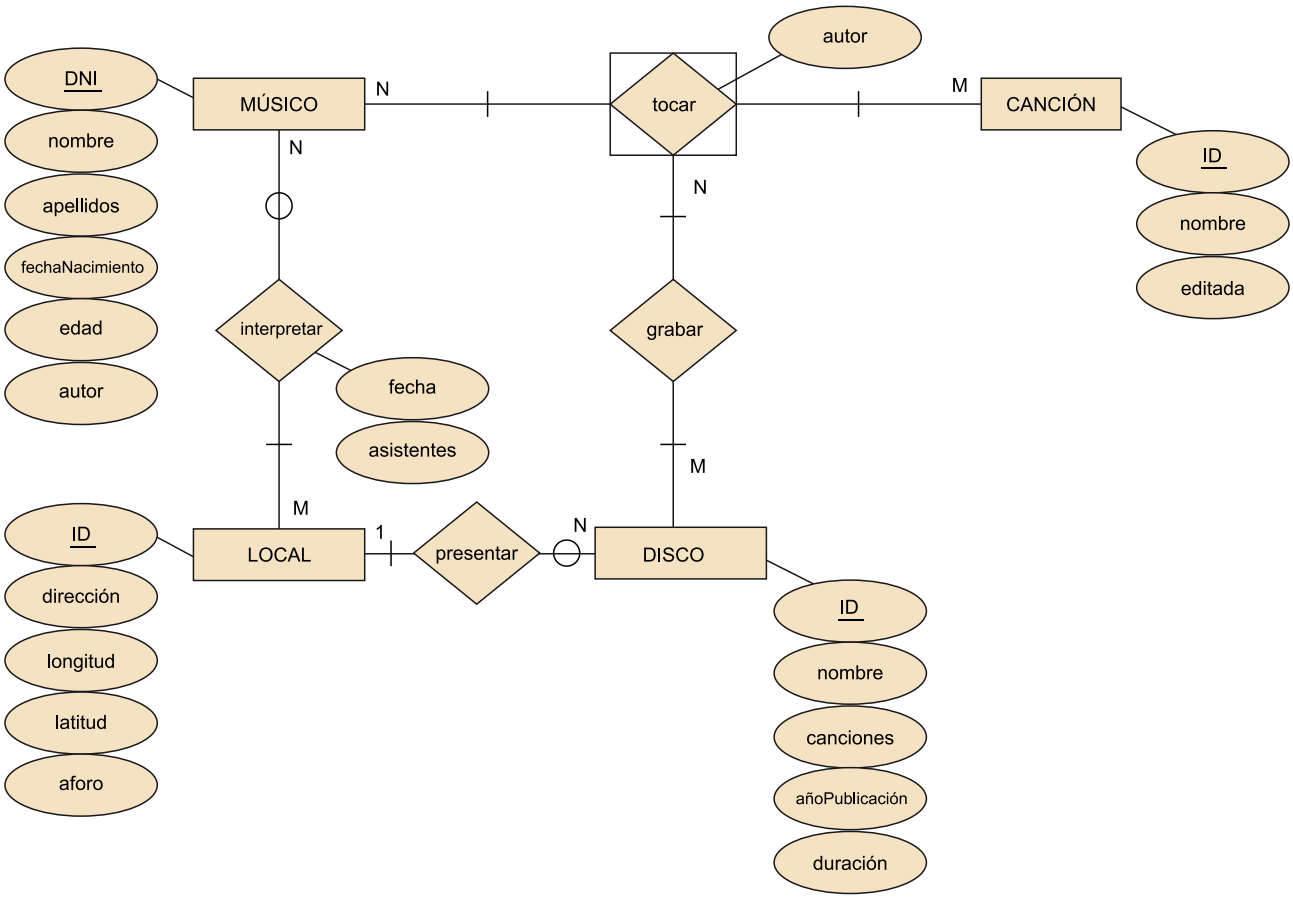
Por otra parte, se querrá tener un recuento de los músicos que la han grabado en un disco (ya sea porque es su autor o porque hace una versión distinta de ella).

Como en el caso anterior, de cada músico se querrá guardar el nombre y apellidos, el DNI (que se utilizará como identificador), la fecha de nacimiento, la edad y, también, el número de canciones de las que es autor y que la discográfica le ha editado.

De cada disco se querrá guardar, además de las canciones que contiene y los músicos que han participado en él, el nombre, el número de canciones que tiene, el año de publicación y su duración. Por otra parte, de cada disco se querrá saber en qué local se presentó.

Por cada local se querrá saber el nombre, la dirección postal, el aforo y la posición geográfica (longitud y latitud). Por otra parte, se querrá guardar una lista de los músicos que han interpretado en ese local, guardando la fecha de aquella ocasión y los asistentes al concierto.

El modelo ER resultante durante el diseño conceptual es:



Se pide transformar el modelo ER en modelo relacional siguiendo las pautas indicadas en el módulo.

Ejercicio 2. Se quiere crear una página web de noticias para difundir las noticias más relevantes de sucesos nacionales e internacionales. Las noticias se clasificarán por categorías (por ejemplo: «Educación», «Política», «Economía» o «Sociedad»). Dentro de cada categoría se pueden establecer subcategorías, así los lectores podrán disponer de información más catalogada (por ejemplo: dentro de la categoría «Educación», podríamos tener las subcategorías «Universidades», «Guarderías» e «Investigación») si lo desean.

Las noticias presentadas en el portal en línea podrán ser votadas por los lectores. Las más votadas podrán aparecer en la portada principal del sitio web. Hay dos tipos de votaciones, las positivas y las negativas. Para poder calcular el valor final de la noticia, las votaciones positivas suman un punto (+1) en la valoración y las negativas le restan un punto (-1). Interesa almacenar de cada votación quien la ha hecho y cuál ha sido el valor de la votación.

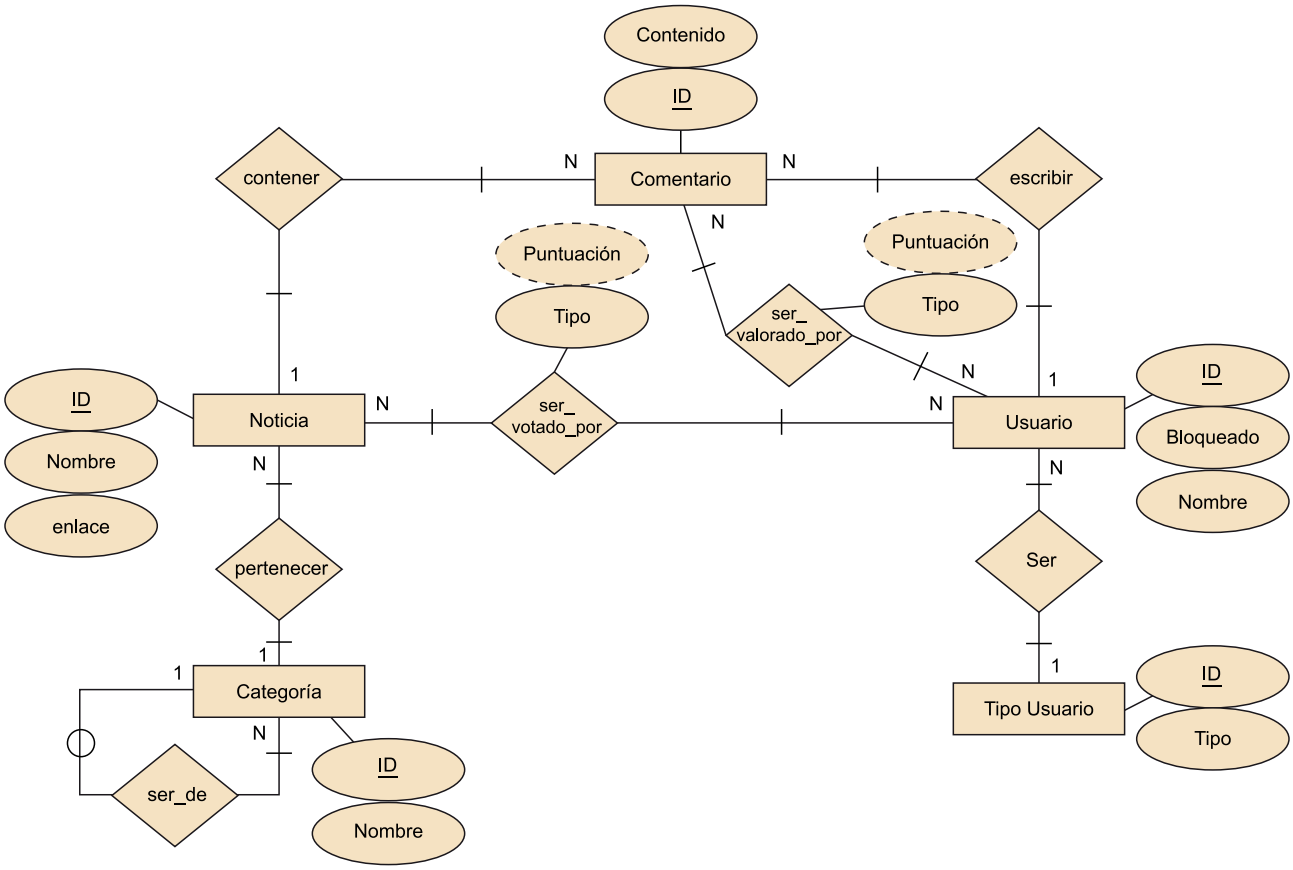
Los usuarios, en el caso de estar registrados, además de votar las noticias, pueden comentarlas. Esto permite crear un hilo de discusión sobre la noticia y, por lo tanto, aporta más información para los usuarios que quieran más. Los comentarios, así como las noticias, pueden tener votos positivos y votos negativos, pero solo de otros usuarios registrados.

Si se diera el caso de que un usuario obtuviera puntuaciones negativas a muchos de sus comentarios, se estudiaría el bloqueo de su cuenta. La acción de bloqueo de cuentas solo la podrán realizar los usuarios administradores. De los usuarios administradores no está previsto guardar información en la base de datos.

El portal, como se comentaba al principio, tiene como objetivo principal la difusión de noticias, y es por esto que no mantiene su contenido, sino que solo enlaza en la noticia original, ya sea de un blog, de un diario en línea o de una entidad pública como por ejemplo las secciones del BOC.

Se pide transformar el modelo ER en modelo relacional, siguiendo las pautas indicadas en el módulo.

El modelo ER resultante durante el diseño conceptual es:



Se pide transformar el modelo ER en modelo relacional, siguiendo las pautas indicadas en el módulo.

Ejercicio 3. Considérese la relación *PEDIDO* con el siguiente esquema:

PEDIDO (*codigoPedido*, *codigoProducto*, *cantidad, codigoCliente, fechaPedido*)

La intención de esta relación es representar los datos de los productos que los clientes piden en sus pedidos.

Respecto a esta relación, se pide:

a) ¿En qué forma normal está la relación? ¿Por qué?

b) Hay que normalizar la relación, lo máximo posible, hasta FNBC.

Ejercicio 4. Considerando la relación *HORARIO* con el siguiente esquema:

HORARIO (*codigoProf*, *codigoAsig*, *dia*, *hora*, *aula*, *nombreAsig*)

Su intención es representar datos de los horarios de las asignaturas que se imparten en un centro universitario presencial. Además, sabemos que una asignatura se puede identificar tanto por el código de asignatura como por el nombre de asignatura.

Respecto a esta relación *HORARIOS*, se pide:

a) ¿En qué forma normal está la relación? ¿Por qué?

b) Hay que normalizar la relación, lo máximo posible, hasta FNBC.

Ejercicio 5. Una empresa dedicada al transporte internacional de mercancías quiere registrar los datos de los viajes que realizan sus empleados. Por cada viaje que se hace se utiliza un único camión, donde viajan un conductor o más, según la distancia del viaje. Adicionalmente, cada viaje tiene como destino una única ciudad del continente. Se quieren registrar las dietas de los conductores en cada viaje. Para ello se ha creado una relación *DIETA* con el siguiente esquema:

DIETA (*codigoViaje*, *codigoConductor*, *matriculaCamion*, *importeDieta*, *ciudadDestino*, *paisDestino*)

Se pide:

a) ¿En qué forma normal está la relación? ¿Por qué?

b) Hay que normalizar la relación, lo máximo posible, hasta FNBC.

Solucionario

Ejercicio 1

MUSICO(DNI, nombre, apellidos, fechaNacimiento)
LOCAL(ID, nombre, direccion, longitud, latitud, aforo)
CANCION(ID, nombre, editada)
DISCO(ID, nombre, canciones, añoPublicacion, duración, localID)
donde {localID} referencia LOCAL(ID)
donde {localID} puede ser nulo
INTERPRETAR(musicoID, localID, fecha, asistentes)
donde {musicoID} referencia MUSICO(DNI)
donde localID referencia LOCAL(ID)
donde {musicoID} puede ser nulo
TOCAR(musicoID, cancionID, autor)
donde {musicoID} referencia MUSICO(DNI)
donde cancionID referencia CANCION(ID)
ENREGISTRAR(musicoID, cancionID , discID)
donde {musicoID, cancionID } referencia tocar
donde {discoID} referencia DISCO(ID)

Ejercicio 2

COMENTARIO(ID, contenido, usuarioID, noticiaID)
donde {usuarioID} referencia USUARIO(ID)
donde {noticiaID} referencia NOTICIA(ID)
USUARIO(ID, nombre, bloqueado, tipoUsuarioID)
donde {tipusUsuariID} referencia TIPO_USUARIO(ID)
NOTICIA(ID,nombre, enlace, categoriaID)
donde {categoriaID} referencia CATEGORIA (ID)
CATEGORIA(ID, nombre, categoriaID)
donde {categoriaID} referencia CATEGORIA (ID)
donde {categoriaID} puede ser nulo
TIPO_USUARIO(ID, tipo)
SER_VOTAT_PER(noticiaID, usuarioID, tipo)
donde {noticiaID} referencia NOTICIA (ID)
donde {usuarioID} referencia USUARIO (ID)

Ejercicio 3

Para solucionar este ejercicio, tenemos que buscar las dependencias funcionales de la relación *PEDIDO*, sabiendo que {*codigoPedido, codigoProducto*} es la clave primaria de la relación *PEDIDO*. De ello se derivan estas dependencias:

{*codigoPedido, codigoProducto*} → {*cantidad*}
{*codigoPedido, codigoProducto*} → {*codigoCliente*}
{*codigoPedido, codigoProducto*} → {*fechaPedido*}

Adicionalmente, dado que cada pedido es de un único cliente y se tramita en una fecha determinada, llegamos a la conclusión de que también están las dependencias funcionales siguientes:

{*codigoPedido*} → {*codigoCliente*}
{*codigoPedido*} → {*fechaPedido*}

a) Por lo tanto, en la relación *PEDIDO* hay atributos que no dependen totalmente de la clave primaria y, además, estos atributos no son clave alternativa de la relación ni forman parte de ninguna clave alternativa de la relación. En consecuencia, la relación *PEDIDO* está en 1FN.

Para normalizar la relación *PEDIDO* a 2FN habrá que evitar las dependencias funcionales no completas respecto de la clave, como se muestra a continuación:

PEDIDO (*codigoPedido*, *codigoCliente*, *fechaPedido*)
PRODUCTOPEDIDO (*codigoPedido*, *codigoProducto*, *cantidad*)
donde {*codigoPedido*} se refiere a *PEDIDO*

Si representamos el conjunto de dependencias funcionales de cada relación, obtenemos:

PEDIDO (*codigoPedido*, *codigoCliente*, *fechaPedido*)
{*codigoPedido*} → {*codigoCliente*}

{*codigoPedido*} → {*fechaPedido*}

PRODUCTOPEDIDO (*codigoPedido*, *codigoProducto*, *cantidad*)
donde {*codigoPedido*} se refiere a *PEDIDO*
{*codigoPedido*, *codigoProducto*} → {*cantidad*}

Vemos que, como consecuencia de la normalización, hemos separado los dos conceptos semánticos que estaban representados en la relación original *PEDIDO*:

- Los pedidos en sí mismos.
- Los productos que se ordenan en cada pedido.

Si examinamos las dos relaciones obtenidas (*PEDIDO* o *PRODUCTOPEDIDO*) llegamos rápidamente a la conclusión de que ambas están en FNBC.

Ejercicio 4

Para solucionar este ejercicio, tenemos que buscar las dependencias funcionales de la relación *HORARIO*. Como sabemos que la clave primaria es compuesta y está formada por los atributos (*codigoProf, codigoAsig, dia, hora*), tenemos las siguientes dependencias:

{*codigoProf*, *codigoAsig*, *dia*, *hora*} → {*aula*}
{*codigoProf*, *codigoAsig*, *dia*, *hora*} → {*nombreAsig*}

Además, también están las dependencias funcionales siguientes:

{*codigoAsig*} → {*nombreAsig*}
{*nombreAsig*} → {*codigoAsig*}

Observamos, pues, que en la relación hay un atributo que no depende totalmente de la clave primaria:

{*codigoAsig*} → {*nombreAsig*}

Este hecho podría violar la 2FN, pero este atributo forma parte de una clave alternativa de la relación *HORARIO*. Por lo tanto, la relación *HORARIO* tiene la siguiente clave alternativa:

{*codigoProf, nombreAsig, dia, hora*}

Se aplica la excepción asociada a la 2FN y por lo tanto podemos afirmar que como mínimo la relación está en 2FN.

Si volvemos a examinar el conjunto de dependencias funcionales, nos damos cuenta de que hay un atributo no-clave que determina funcionalmente otro atributo no-clave, lo que nos hace considerar la dependencia funcional:

{*nombreAsig*} → {*codigoAsig*}

Este hecho podría violar la 3FN y, dado que involucra un atributo que forma parte de una clave alternativa, se aplica la excepción. En consecuencia, la relación *HORARIO* también verifica la 3FN.

Para saber si la relación *HORARIO* está en FNBC, tenemos que refinar nuestro conjunto de dependencias funcionales. Puesto que (*codigoProf, nombreAsig, dia, hora*) es clave alternativa de la relación *HORARIO*, estamos seguros de que deben existir las dependencias funcionales siguientes:

{*codigoProf, nombreAsig, dia, hora*} → {*aula*}
{*codigoProf, nombreAsig, dia, hora*} → {*codigoAsig*}

Si ahora comparamos todos los determinantes obtenidos y las claves candidatas de la relación *HORARIO*, vemos que no todos los determinantes son claves candidatas, y, por lo tanto, la relación no está en FNBC. A continuación, mostramos los determinantes y las claves candidatas de la relación *HORARIO*:

Horario	
Determinantes	*Claves candidatas*
codigoProf, codigoAsig, dia, hora	codigoProf, codigoAsig, dia, hora

Horario	
codigoProf, nombreAsig, dia, hora	codigoProf, nombreAsig, dia, hora
codiAsig	
nombreAsig	

En conclusión, la relación *HORARIO* está en 3FN.

Para conseguir normalizar la relación *HORARIO* en FNBC, habrá que evitar las dependencias funcionales siguientes:

{*codigoAsig*} → {*nombreAsig*}
{*nombreAsig*} → {*codigoAsig*}

Esto se puede conseguir de distintas maneras. A continuación, mostramos todas las posibilidades:

a)

HORARIO (codigoProf, codigoAsig, dia, hora, aula)
{*codigoProf*, *codigoAsig*, *dia*, *hora*} → {*aula*}
ASIGNATURA (codigoAsig, nombreAsig)
{*codigoAsig*} → {*nombreAsig*}

b)

HORARIO (codigoProf, codigoAsig, dia, hora, aula)
{*codigoProf*, *codigoAsig*, *dia*, *hora*} → {*aula*}
ASIGNATURA (nombreAsig, codigoAsig)
{*nombreAsig*} → {*codigoAsig*}

c)

HORARIO (codigoProf, nombreAsig, dia, hora, aula)
{*codigoProf*, *nombreAsig*, *dia*, *hora*} → {*aula*}
ASIGNATURA (nombreAsig, codigoAsig)
{*nombreAsig*} → {*codigoAsig*}

d)

HORARIO (codigoProf, nombreAsig, dia, hora, aula)
{*codigoProf*, *nombreAsig*, *dia*, *hora*} → {*aula*}
ASIGNATURA (codigoAsig, nombreAsig)
{*codigoAsig*} → {*nombreAsig*}

Vemos que, como consecuencia de la normalización, hemos separado los dos conceptos semánticos que estaban representados en la relación original *HORARIO*:

- Los horarios en sí mismos
- Las asignaturas impartidas

Ejercicio 5

Para solucionar este ejercicio, tenemos que buscar las dependencias funcionales de la relación *DIETA*. Estas dependencias son:

{*codigoViaje, codigoConductor*} → {*matriculaCamion*}
{*codigoViaje, codigoConductor*} → {*importeDieta*}
{*codigoViaje, codigoConductor*} → {*ciudadDestino*}
{*codigoViaje, codigoConductor*} → {*paisDestino*}

Estas dependencias funcionales son consecuencia del hecho que (*codigoViaje, codigoConductor*) es la clave primaria de la relación *DIETA*.

Dado que en cada viaje solo se utiliza un camión y el destino es único, también hay las dependencias funcionales siguientes:

{*codigoViaje*} → {*matriculaCamion*}
{*codigoViaje*} → {*ciudadDestino*}
{*codigoViaje*} → {*paisDestino*}

Para terminar, la última dependencia funcional que encontramos es esta:

{*ciudadDestino*} → {paisDestino}

La veracidad de esta dependencia funcional permanece supeditada a que no se hacen viajes a ciudades que tienen el mismo nombre, pero están en países distintos. A partir de estas dependencias funcionales podemos decir lo siguiente:

a) Dado que en la relación *DIETA* hay atributos que no dependen totalmente de la clave primaria y, además, estos atributos no son clave alternativa ni forman parte de claves alternativas de la relación *DIETA*, podemos afirmar que la relación *DIETA* está en 1FN.

b) Hay varias alternativas para normalizar la relación. A continuación, mostramos dos de ellas:

1. Normalizamos la relación *DIETA* original a 2FN. Tenemos que descomponer la relación original en dos:

VIAJE (*codigoViaje*, *matriculaCamion*, *ciudadDestino*, *paisDestino*)
{*codigoViaje*} → {*matriculaCamion*}
{*codigoViaje*} → {*ciudadDestino*}
{*codigoViaje*} → {*paisDestino*}
{*ciudadDestino*} → {*paisDestino*}

DIETA (codigoViaje, *codigoConductor*, *importeDieta)*
donde {*codigoViaje*} se refiere a *VIAJE*
{codigoViaje, codigoConductor} → {importeDieta}

La nueva relación *DIETA* ya está en FNBC; en cambio, la relación *VIAJE* viola la 3FN, dado que hay dependencias funcionales entre atributos que no son clave ni forman parte de claves alternativas. Para normalizar la relación *VIAJE*, tenemos que descomponerla en dos relaciones, como se muestra a continuación:

VIAJE (codigoViaje, matriculaCamion, ciudadDestino)
donde {*ciudadDestino*} se refiere a *CIUDAD {ciudad}*
{codigoViaje}→ {*matriculaCamion*}
{*codigoViaje*} → {*ciudadDestino*}

CIUDAD (*ciudad*, *pais*)
{*ciudad*} → {*pais*}

Ahora, tanto la relación *VIAJE* como la relación *CIUDAD* están en FNBC.

2. En este caso, y tras examinar detenidamente la relación *DIETA*, nos damos cuenta de que en la relación se representan tres hechos diferentes del mundo real: los viajes, las dietas de los distintos viajes y las ciudades. Para conseguir separar estos hechos, normalizamos directamente hasta la FNBC, que nos garantiza que cualquier determinante a una dependencia funcional debe ser una clave candidata de la relación donde se da la dependencia funcional. Como consecuencia, obtenemos tres relaciones: la relación *VIAJE*, la relación *CIUDAD* y, finalmente, la relación *DIETA*:

VIAJE (codigoViaje, matriculaCamion, ciudadDestino)
donde {*ciudadDestino*} se refiere a *CIUDAD* {ciudad}
{*codigoViaje*} → {*matriculaCamion*}
{*codigoViaje*} → {*ciudadDestino*}

CIUDAD (*ciudad*, *pais*)
{*ciudad*} → {*pais*}

DIETA (codigoViaje, *codigoConductor*, *importeDieta)*
donde {*codigoViaje*} se refiere a *VIAJE*
{*codigoViaje*, *codigoConductor*} → {*importeDieta*}

Glosario

atributo de una relación *m* Nombre del rol que ejerce un dominio en un esquema de relación.

cardinalidad *f* En una relación, el grado es el número de tuplas que pertenecen a su extensión.

clave candidata de una relación *f* Superclave C de la relación en la que ningún subconjunto propio de C es superclave.

clave foránea de una relación *f* Subconjunto de los atributos del esquema de la relación CF, en la que existe una relación S (que puede ser ella misma) que tiene por clave primaria CP, y se cumple que, para toda tupla *t* de la extensión de R, los valores para CF de *T* son o bien valores nulos, o bien valores que coinciden con los valores para CP de alguna tupla *s* de S.

clave primaria de una relación *f* Clave candidata de la relación escogida para identificar las tuplas de la relación.

diseño lógico *m* Proceso de transformación de un esquema conceptual en un esquema lógico de base de datos.

dominio *m* Conjunto de valores atómicos.

esquema de la relación *m* Componente de una relación que consiste en un nombre de relación R y en un conjunto de atributos {A1, A2, …, An}.

extensión de una relación de esquema R (A *f* Conjunto de tuplas t_i (i = 1, 2, ..., m) en el que cada tupla t_i es un conjunto de pares t_i = {<A_1: V_{i1}>, <A_2: V_{i2}> ..., <A_n: V_{in}>:} y, por cada par <A_j: V_{ij}>, se cumple que v_{ij} es un valor que domina (A_{ij}) o bien un valor nulo.

grado *m* En una relación, su grado es el número de atributos que pertenecen a su esquema.

inserción *f* Hecho de añadir una o más tuplas a una relación.

normalización *f* Propiedad de los datos de corresponder a representaciones plausibles del mundo real.

integridad *f* Proceso por el que, a partir de un conjunto de relaciones, se obtiene un conjunto de relaciones equivalente que satisface la condición de la forma normal deseada.

relación *f* Elemento de la estructura de los datos de una BD relacional formado por un esquema (o intensión) y una extensión.

superclave de una relación *f* Subconjunto de los atributos del esquema en el que no puede haber dos tuplas en la extensión de la relación que tengan la misma combinación de valores para los atributos del subconjunto.