
	Bases de datos Tema 4	
	Diseño de bases de datos relacionales Modelo Relacional	

1. El modelo Relacional

El modelo relacional fue desarrollado por Edgar Frank Codd en 1970. El modelo de Codd persigue al igual que la mayoría de los modelos de datos los siguientes objetivos:

1. **Independencia física** de los datos, esto es, el modo de almacenamiento de los datos no debe influir en su manipulación lógica.
2. **Independencia lógica** de los datos, es decir, los cambios que se realicen en los objetos de la base de datos no deben repercutir en los programas y usuarios que acceden a ella.
3. **Flexibilidad**, para representar a los usuarios los datos de la forma más adecuada.
4. **Uniformidad**, en la presentación de la lógica de los datos, que son tablas, lo que facilitan la manipulación de la base de datos por parte de los usuarios.
5. **Sencillez**, este modelo es fácil de comprender y utilizar por el usuario.

Para con estos objetivos Codd introduce el concepto de **relación (tablas)** como estructura básica del modelo, todos los datos de una base de datos se representa en forma de relaciones cuyo contenido varía en el tiempo.

1.1. Conceptos del modelo relacional

A continuación se definen los conceptos necesarios para transformar el modelo conceptual (diagrama entidad-relación) en el modelo lógico (modelo relacional).

1.1.1. Relación

El elemento básico del modelo relacional es la relación, que puede representarse en forma de tablas. Es una estructura de datos que se representa con un nombre y un conjunto de atributos o columnas junto con el tipo de datos de cada una de ellas. Por ejemplo, un jugador de una base de datos de una liga de baloncesto quedaría representado del siguiente modo:

Jugadores (id_jugador entero(4), cif carácter(9), nombre carácter(20), apellidos carácter (20), equipo carácter(20), edad entero(3), fech_alta fecha)

De modo que representamos la tabla o relación jugadores con siete campos o características de un jugador. Además, se incluye el tipo de datos para cada atributo especificando el tamaño o dominio de los mismo.

1.1.2. Dominio y atributo

Se define *dominio* como el conjunto finito de valores permitidos (todos del mismo tipo) y atómicos (son indivisibles) que puede tomar cada atributo. Todos los dominios tienen un nombre y un tipo de datos asociados. Por ejemplo, cadenas de caracteres, números enteros, los valores Sí o No, etc..

Se define *atributo* como el papel o rol que desempeña un dominio en una relación. Representa el uno de un dominio para una determinada relación. El atributo aporta un significado semántico a un dominio.

Relación: ImpuestosVehículos

Vehículo	Dueño	Teléfono	Matrícula	Cuota
Seat Ibiza TDI 1.9	Francisco	925123654	9918-FTV	92,24
Volkswagen Polo TDI 2.0	Pedro	918123456	4231-FHD	61,98
Fiat Punto 1.0	Ernesto	646987654	9287-BHF	45,77
Renault Laguna	María	876987543	7416-GSJ	145,32

Definición de relación

Relación: Corresponde con la idea de tabla.

Atributos: Corresponde con cada columna (Vehículo, Dueño, Teléfono,...)

Tupla: Corresponde con una fila.

Cardinalidad: Número de tuplas, 4 en ejemplo

Grado: Número de atributos, 5 en nuestro ejemplo.



Dominio: Para el atributo Vehículo carácter(20), ..., Cuota entero(5)

Clave primaria: Identificador único (no hay dos tuplas con igual identificador), en el ejemplo podría ser el *número de matrícula*.

1.1.3. Restricciones semánticas

También llamadas de usuario. Son condiciones que deben cumplir los datos para su correcto almacenamiento. Hay varios tipos:

- **Restricción de clave:** Es el conjunto de atributos que identifican de forma única a una entidad.
- **Restricciones de valor único (UNIQUE):** Es una restricción que impide que un atributo tenga un valor repetido. Todos los atributos claves cumplen esta restricción. No obstante, es posible que algún atributo que no sea clave y requiera una restricción de valor único. Por ejemplo el número de bastidor de un vehículo no es clave y sin embargo no puede haber ningún número de bastidor repetido.
- **Restricciones de integridad referencial:** Se da cuando una tabla tiene referencia a algún valor de otra tabla. En este caso, la restricción exige que exista el valor referenciado en la otra tabla.
- **Restricciones de dominio:** Esta restricción exige que el valor que puede tomar un campo esté dentro del dominio definido. Por ejemplo, si se

	Bases de datos Tema 4	
	Diseño de bases de datos relacionales Modelo Relacional	

establece que un campo DNI pertenece al dominio de los número de 9 dígito + 1 letra, no es posible insertar un DNI sin letra.

- **Restricciones de verificación (CHECK):** Esta restricción permite comprobar si un valor de un atributo es válido conforme a una expresión.
- **Restricción de valor nulo (NULL o NOT NULL):** Un atributo puede ser obligatorio si no admite valor NULO o NULL, es decir, el valor *falta de información o desconocimiento*.
- **Disparadores o triggers:** Son procedimientos que se ejecutan para hacer una tarea concreta en el momento de insertar, modificar o eliminar información de una tabla.
- **Restricciones genéricas adicionales o aserciones (ASSERT):** Permite validar cualquiera de los atributos de una o varias tablas.

1.1.4. Clave

Una clave es un conjunto de atributos que identifican de forma única una ocurrencia de entidad. En este caso, las claves pueden ser simples (atómicas) o compuestas. Además, hay varios tipos de clave:

- **Superclave:** Identifica a una entidad (puede ser no mínima). Por ejemplo, para un empleado, las superclaves posibles sin el DNI, o el DNI+Nombre o el DNI+Nombre+Número de la Seguridad Social, etc.
- **Clave candidata:** Es la mínima Superclave (en el caso anterior el DNI o el Número de la Seguridad Social).
- **Clave Primaria:** Es la clave candidata elegida por el diseñador como clave definitiva (en el ejemplo anterior se elegiría el DNI por ser la más representativa para el empleado).
- **Clave foránea:** Es un atributo de una entidad, que es clave en otra entidad. Por ejemplo, la nota de un módulo de una asignatura corresponde a un DNI, que es clave de otra entidad. En este caso el DNI es una clave foránea en la tabla notas.

2. Transformación de un diagrama E/R al modelo relacional

Una vez obtenido el esquema conceptual mediante el modelo E/R hay que definir el modelo de datos. Las reglas básicas para transformar un esquema conceptual E/R a un esquema relacional son las siguientes:

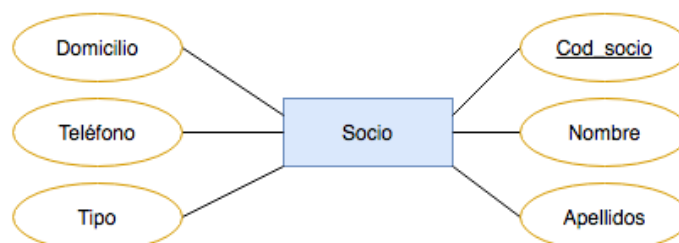
- Toda entidad se transforma en una tabla.
- Todo atributo se transforma en columnas dentro de una tabla.
- El identificador único de la entidad se convierte en clave primaria.
- Los atributos obligatorios se transforman en una columna en la cual no admiten valores nulos.
- Los atributos opcionales se transforman en una columna la cual admite valores nulos.
- Los atributos multivaluados, se transforman en una relación formada con la clave primaria de la entidad y el atributo multivaluado, siendo ambos clave primaria de la nueva relación (hay otras posibilidades).
- Los atributos compuestos, se transforman en los atributos simples (campos) que componen el atributo compuesto, desapareciendo esto como tal en la relación.

Atributos derivados, no forman parte del modelo relacional resultante, quedando eliminados en esta parte del diseño.

2.1. Ejemplos de transformación

2.1.1. Transformación de una entidad

Para una entidad A, con atributos ($a_1, a_2, a_3, \dots, a_n$) se crea una tabla A (con el nombre en plural) con n columnas correspondientes a los atributos de A, donde cada fila de A corresponde a una ocurrencia de la entidad A. La clave primaria de la tabla A la forman los atributos claves de la entidad A.



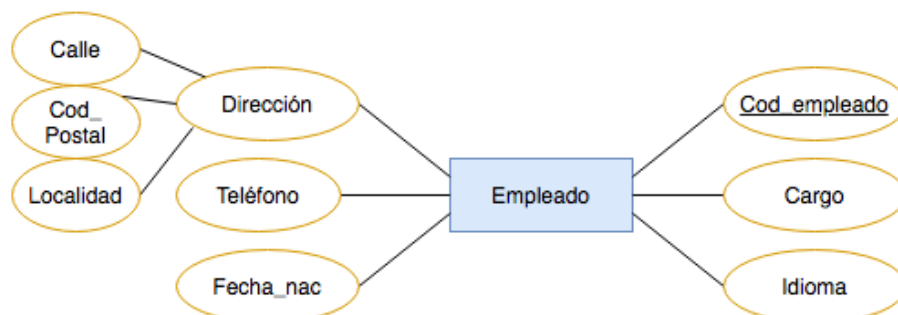
<i>Cod_socio</i>	<i>Nombre</i>	<i>Apellidos</i>	<i>Domicilio</i>	<i>Teléfono</i>	<i>Tipo</i>
00001	Susana	García	Rios 4	606211388	I
00002	Adolfo	Rguez	San Ben 22	712999577	P
00003	Antonio	Hdez	Pte 20	606234456	A
00004	Miguel	López	Los Silos 55	767989765	I
.
.
00007	Elena	Ledesma	Hemosa 14	666787542	I


 Clave Primaria

SOCIOS (Cod_socio, Nombre, Apellido, Domicilio, Teléfono, Tipo)

2.1.2. Transformación de una entidad con atributos compuestos

El modelo relacional en su forma básica contiene solo atributos simples, es por ello los atributos compuestos deben ser modelados en términos de atributos simples. Con cada atributo compuesto se tiene básicamente dos alternativas:



1. Eliminar el atributo compuesto considerando todos sus componentes como atributos individuales.

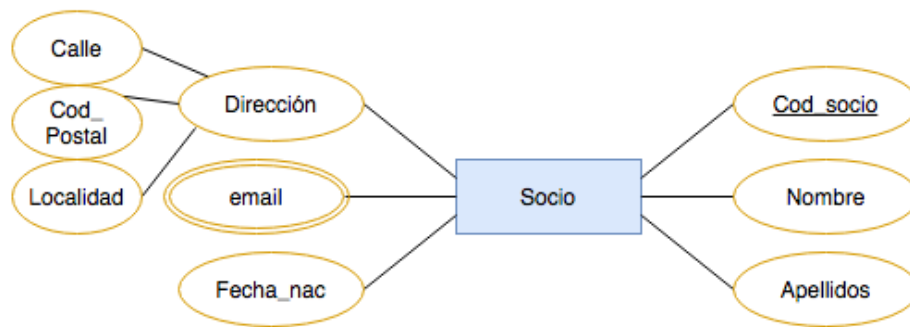
EMPLEADOS (Cod_Empleado, Cargo, Idioma, Calle, Cod_postal, Localidad, Teléfono, Fecha_nac)

2. Eliminar los componentes individuales y considerar el atributo compuesto entero en un solo atributo.

EMPLEADOS (Cod_Empleado, Cargo, Idioma, Dirección, Teléfono, Fecha_nac)

2.1.3. Transformación de una entidad con atributos multivaluados

Los atributos multivaluados requieren la introducción de entidades nuevas; cada atributo multivaluado distinto requiere en la entidad en la cual pueda estar representado como atributo sencillo (también llamado monovaluado). La nueva entidad contiene el atributo multivaluado más el identificador de la entidad original; el identificador de la nueva entidad es el conjunto de todos sus atributos.



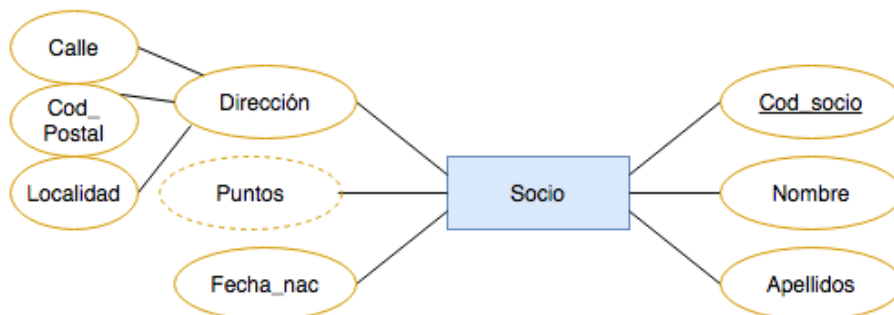
SOCIOS (Cod_socio, Nombre, Apellidos, Calle, Cod_postal, Localidad, Fecha_nac)

EMAILS (Id_email, email, Cod_socio (FK))

2.1.4. Transformación de una entidad con atributos derivados

No existe una representación directa. Por tanto se deben tratar como atributos normales, que pasarán a ser columnas de la tabla correspondiente, en el mejor de los casos evitarlos.

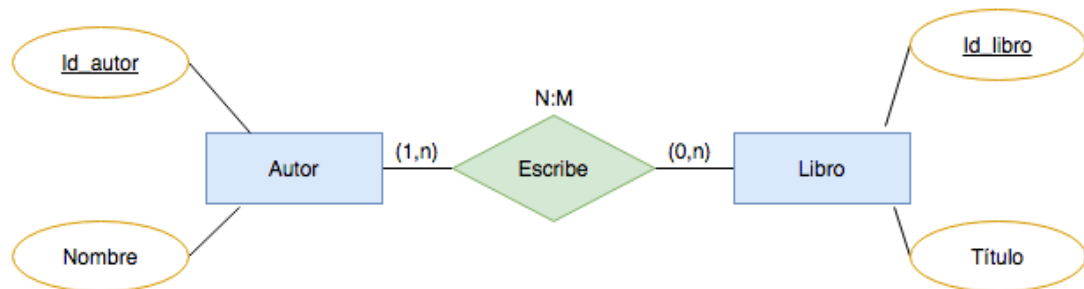
Se debe construir un **disparador (trigger)** que calcule el valor del atributo derivado cada vez que se inserten o borren las ocurrencias de los atributos que intervienen en el cálculo y añadir las restricciones correspondientes.



SOCIOS (Cod_socio, Nombre, Apellidos, Calle, Cod_postal, Localidad, Fecha_nac, Puntos)

2.1.5. Transformación de relaciones N:M

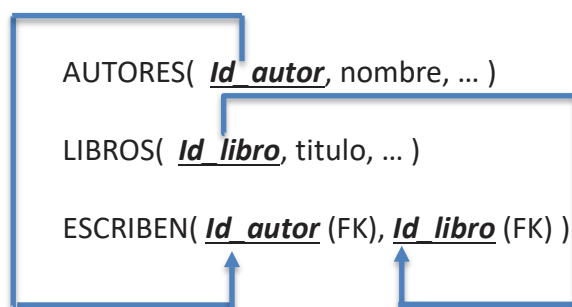
Cada entidad se convierte en un tabla y los atributos de las entidades se convierten en columna de las tablas:



AUTORES (Id_autor, nombre, ...)
 LIBROS (Id_libro, titulo, ...)

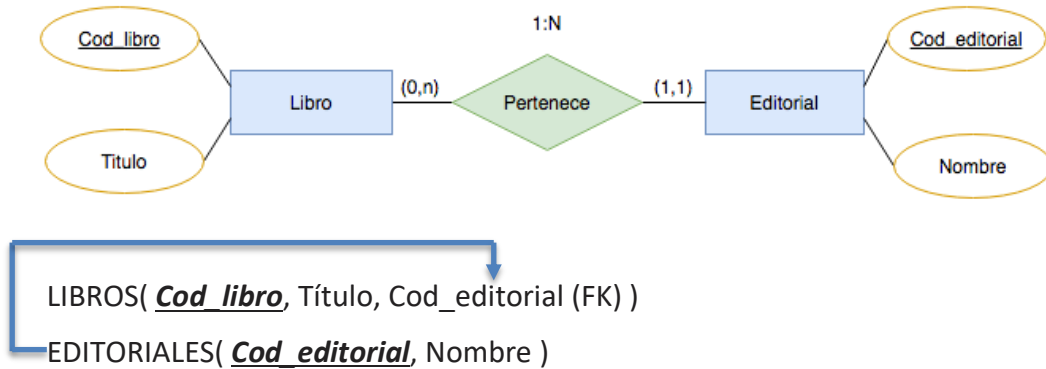
La relación N:M se transforma en una tabla que tendrá como clave primaria la concatenación de los identificadores únicos de los tipos de entidad que asocia.

Los atributos que forman la clave primaria de esta relación son clave ajena (Foreign Key) respecto a cada una de las tablas donde este atributo es clave primaria.

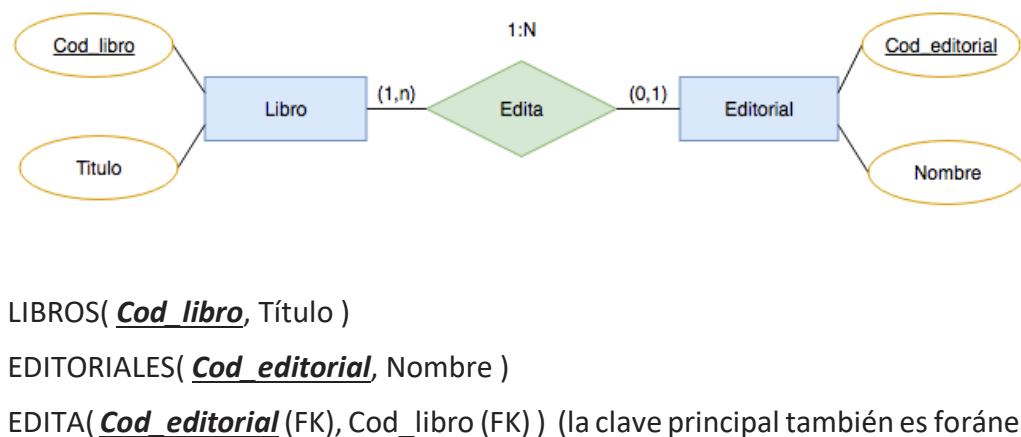


2.1.6. Transformación de relaciones 1:N

- a) Propagar el atributo identificador único del tipo de entidad que tiene cardinalidad máxima 1 al que tiene N. Si la relación tiene atributos éstos se pasarán a la entidad con cardinalidad N.



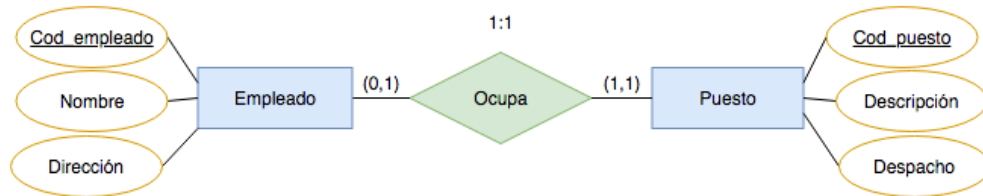
- b) En el caso de que la cardinalidad sea (0,1), entonces se transformará la relación como si se tratara de una relación N:M. Los atributos que tuviera la relación pasarían a la nueva tabla.



2.1.7. Transformación de relaciones 1:1

En la transformaciones de relaciones 1:1 se tiene en cuenta las cardinalidades de las entidades que participa. Existen dos soluciones:

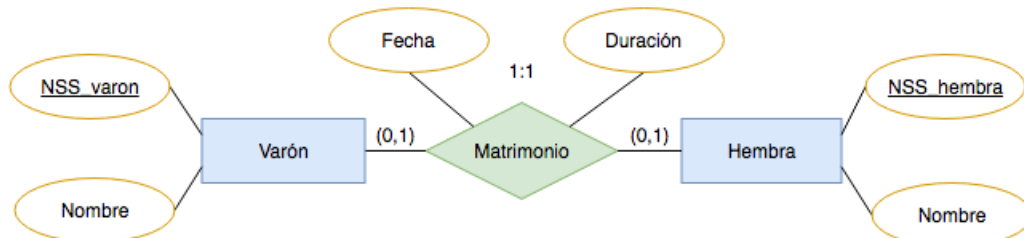
- a) Si una de las entidades que participa en la relación posee cardinalidades (1,1), mientras que en la otra son (0,1), conviene propagar la clave de la entidad con cardinalidades (1,1) a la tabla resultante de la entidad con cardinalidades (0,1). Si ambas entidades poseen cardinalidades (1,1) se puede propagar la clave de cualquiera de ellas a la tabla resultante de la otra.



PUESTOS(Cod_puesto, Descripción, Despacho)

EMPLEADOS(Cod_Empleado, Nombre, Dirección, Cod_Puesto (FK))

- b) **Transformar la relación en una tabla.** Si las entidades poseen cardinalidades (0,1), la relación se convierte en una tabla.



VARONES(NSS_varon, Nombre)

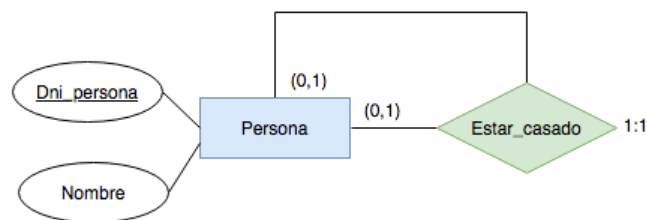
HEMBRAS(NSS_hembra, Nombre)

MATRIMONIO(NSS_varon (FK), NSS_hembra (FK), Fecha, Duración)

2.1.8. Transformación de relaciones reflexivas

Son relaciones binarias en las que participa un tipo de entidad. En el proceso de convertir una relación reflexiva a tabla hay que tener en cuenta sobre todo la cardinalidad. Lo normal es que toda relación reflexiva se convierta en dos tablas, una para la entidad y la otra para la relación. Se pueden presentar los siguientes casos:

- a) **Si la relación es 1:1**, la clave de la entidad se repite, con lo que la tabla resultante tendrá dos veces ese atributo, una como clave primaria y otra como clave foránea de ella misma. No se crea la segunda tabla.



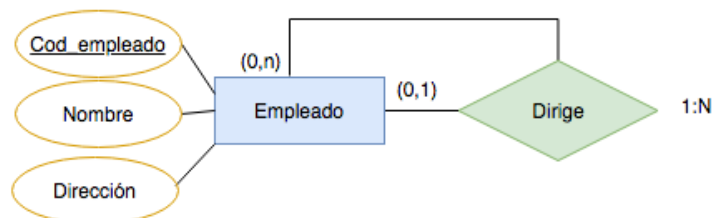
PERSONAS(Dni_persona, Nombre, Dni_persona_C (FK))

- b) **Si la relación es 1:N**, podemos tener dos casos:

- I. Caso de que la entidad muchos sea siempre obligatoria se procede como en el caso 1:1.

EMPLEADOS(Cod_empleado, Nombre, Dirección, Cod_dirección (FK))

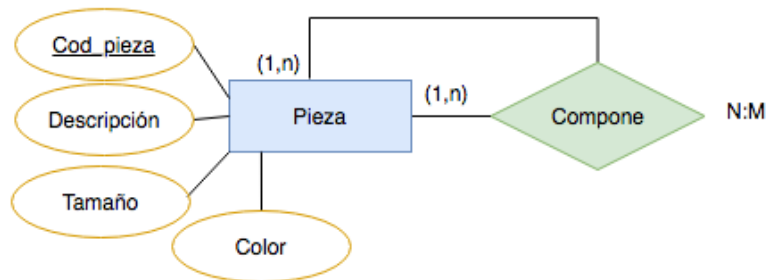
- II. Si no es obligatoria, se crea una nueva tabla cuya clave será la de la entidad del lado muchos, y además se propaga la clave a la nueva tabla como clave ajena.



EMPLEADOS(Cod_empleado, Nombre, Dirección)

DIRIGE(Cod_empleado (FK), Cod_direccion (FK))

- c) Si la relación es N:M, se trata igual que en la relaciones binarias. La tabla resultante de la relación contendrá dos veces la clave primaria de la entidad del lado muchos, más los atributos de la relación si los hubiera. La clave de esta nueva tabla será la combinación de las dos.



PIEZA(Cod_pieza, Descripción, Tamaño, Color)

COMPONE_PIEZA(Cod_pieza_com (FK), Cod_pieza (FK))

2.1.9. Generalizaciones, transformación de jerarquías al modelo relacional

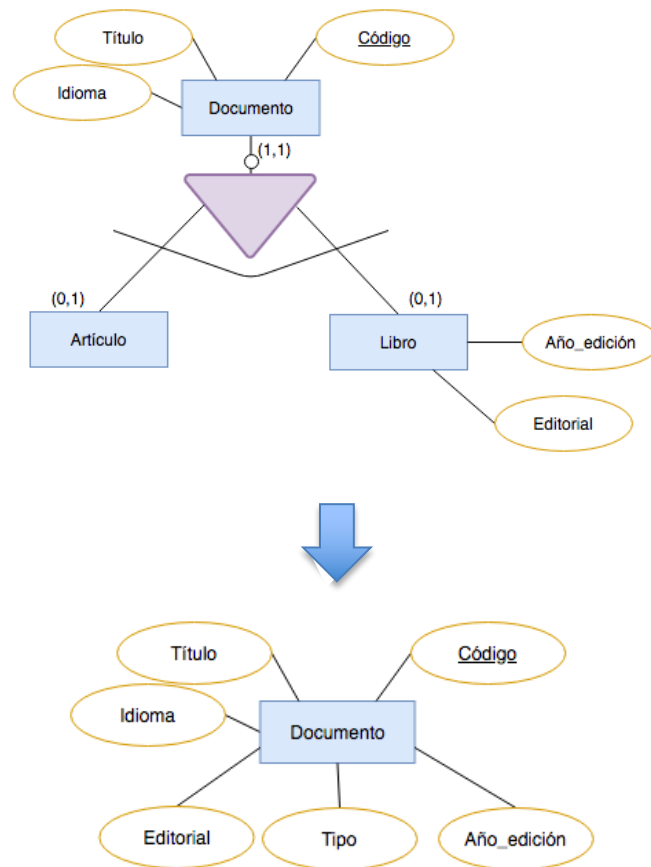
El modelo relacional no dispone de mecanismos para la representación de las relaciones jerárquicas, así pues, las relaciones jerárquicas se tiene que eliminar. Para ello hay que tener en cuentas:

- La especialización que los subtipos tiene respecto a los supertipos, es decir, los atributos diferentes que tengan asociados cada uno de los subtipos, que son los que se diferencian con el resto de atributos de los otros subtipos.
- El tipo de especialización que representa el tipo de relación jerárquica: *total o parcial exclusiva* y *total o parcial solapada*.
- Otros tipos de relación que mantengan tanto los subtipos como el supertipo.
- La forma en la que se va a acceder a la información que representan tanto el supertipo como el subtipo.

Teniendo en cuenta los puntos señalados para pasar estas relaciones al modelo relacional se aplicará una de las siguientes reglas:

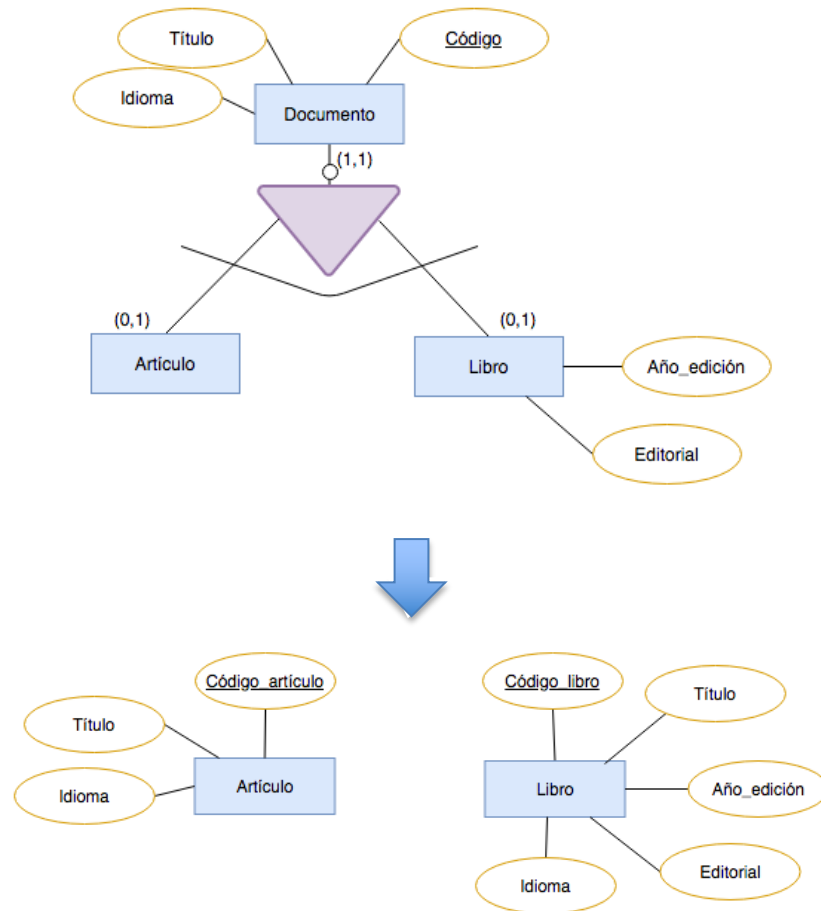
- a) **Integrar todas las entidades en una única, eliminando a los subtipos.** Esta nueva entidad contendrá todos los atributos del supertipo, todos los de los subtipos. Se añade un atributo discriminativo para distinguir a qué subentidad pertenece cada atributo. Todas las relaciones se mantiene con la nueva entidad.

Esta regla puede aplicarse a cualquier tipo de jerarquía. La gran ventaja es la simplicidad pues todo se reduce a una entidad. El gran inconveniente es que se generan demasiados valores nulos en los atributos que se aplican solo a las subentidades y todas las operaciones que tenían acceso solo a las subentidades tiene que buscar ahora el caso correspondiente dentro del conjunto completo de casos de la subentidad.



DOCUMENTOS(Código, Título, Idioma, Editorial, Año_edición, Tipo)

- b) **Eliminación del supertipo.** Transfiriendo los atributos a cada uno de los subtipos. Las relaciones del supertipo se consideran para cada uno de los subtipos. La clave genérica del supertipo pasa a cada uno de los subtipos. Sólo puede ser aplicada para *jerarquías totales y exclusivas*. Inconvenientes que presenta esta transformación:
- Se crea redundancia en la información, pues los atributos del supertipo se repiten en cada uno de los subtipos.
 - El número de relaciones aumenta, pues si el supertipo tiene relaciones, éstas pasan a cada uno de los subtipos.



ARTICULOS(Código_artículo, Título, Idioma)

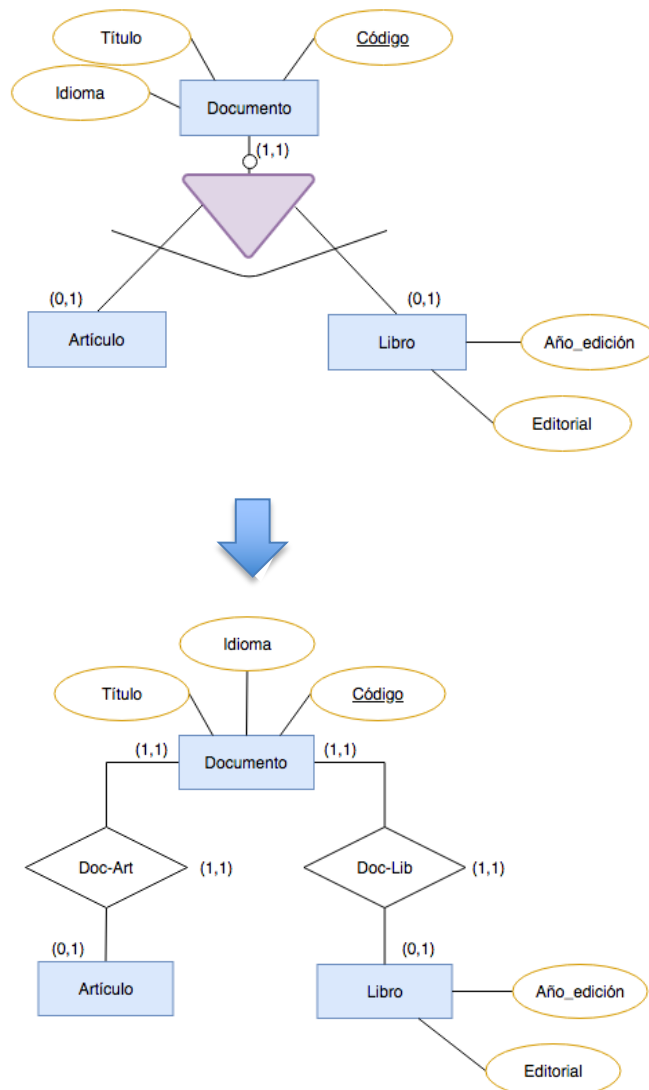
LIBROS(Código_libro, Título, Idioma, Editorial, Año_edición)

- c) **Insertar una relación 1:1 entre el supertipo y cada uno de los subtipos.** Los atributos se mantienen y cada subtipo se identificará con la clave ajena del supertipo. El supertipo mantendrá una relación 1:1 con cada subtipo. Los subtipos mantendrán, si la relación es exclusiva la cardinalidad mínima 0, y si es solapada 0 ó 1.

Esta alternativa tiene dos desventajas:

- El esquema resultante es bastante complejo: por ejemplo para insertar un nuevo caso de subentidad se requiere insertar dos casos adicionales uno para la subentidad u otro para la interrelación con la subentidad.
- Hay una redundancia inherente (al menos en el nivel conceptual). Al representar cada eslabón ISA en la jerarquía a través de una interrelación explícita.

La principal ventaja es que modela las 4 combinaciones de jerarquías parcial/total y exclusiva/superpuesta.



DOCUMENTOS(Código, Título, Idioma)

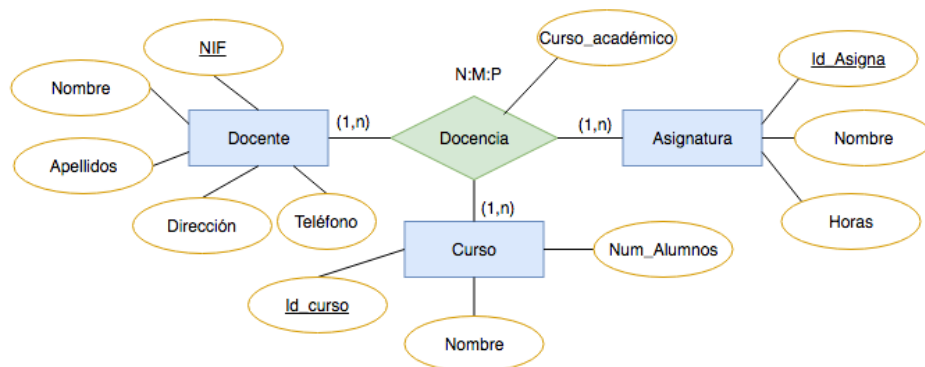
ARTICULOS(Código (FK), ...)

LIBROS(Código (FK), Editorial, Año_edición)

2.1.10. Transformación relaciones N-arias

En este tipo de relaciones se agrupan 3 o más entidades, y para pasar al modelo de datos relacional cada entidad se convierte en tabla, así como la relación, que va a contener los atributos propios de ella más las claves de todas las entidades. La clave de la tabla resultante será la concatenación de las claves de las entidades. Hay que tener en cuenta:

- Si la relación es N:M:P, es decir, si todas las entidades participan con cardinalidad máxima N, la clave de la tabla resultante es la unión de las claves de las entidades que relaciona. Esta tabla incluirá los atributos de la relación si los hubiera.



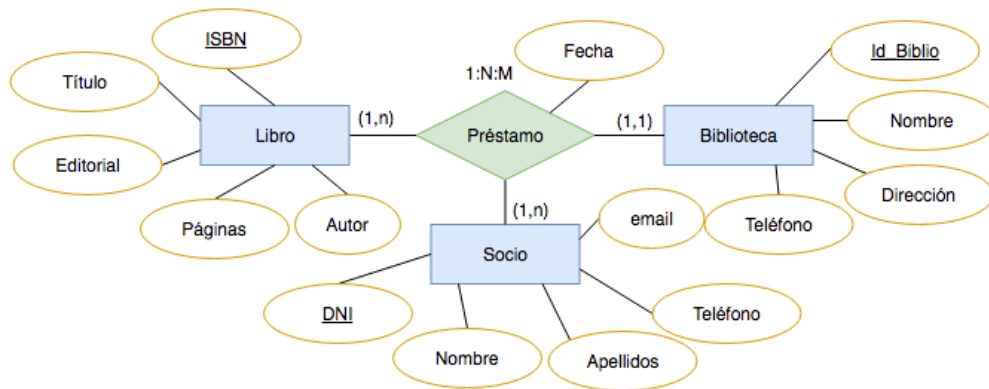
DOCENTES(NIF, Nombre, Apellidos, Dirección, Teléfono)

ASIGNATURAS(Id asigna, Nombre, Horas)

CURSOS(Id curso, Nombre, Num_alumnos)

DOCENCIA(NIF (FK), Id asigna (FK), Id curso (FK), curso académico)

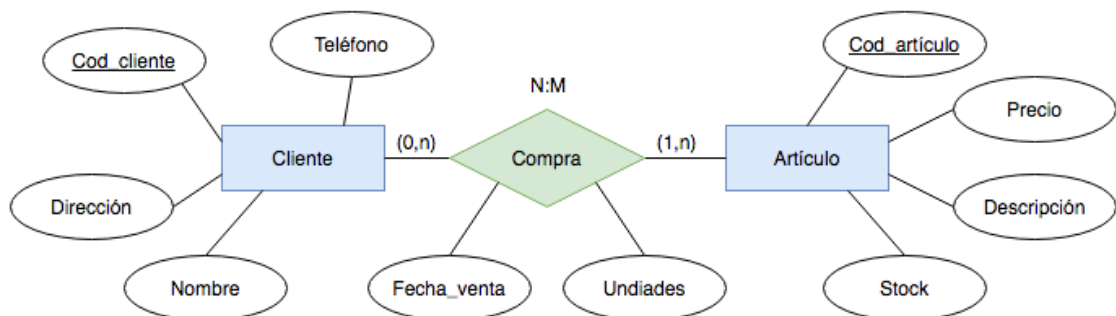
- Si la relación es 1:N:M, es decir, una de las entidades participa con cardinalidad máxima 1, la clave de esta entidad no pasa a formar parte de la clave de la tabla resultante, pero forma parte de la relación como un atributo más.



LIBROS(ISBN, Título, Autor, Páginas, Editorial)
 BIBLIOTECAS(Id_biblio, Nombre, Dirección, Teléfono)
 SOCIOS(DNI, Nombre, Apellidos, Teléfono, Email)
 PRESTAMOS(ISBN (FK), DNI (FK), Id_Biblio (FK), Fecha)

3. Del modelo relacional a las tablas

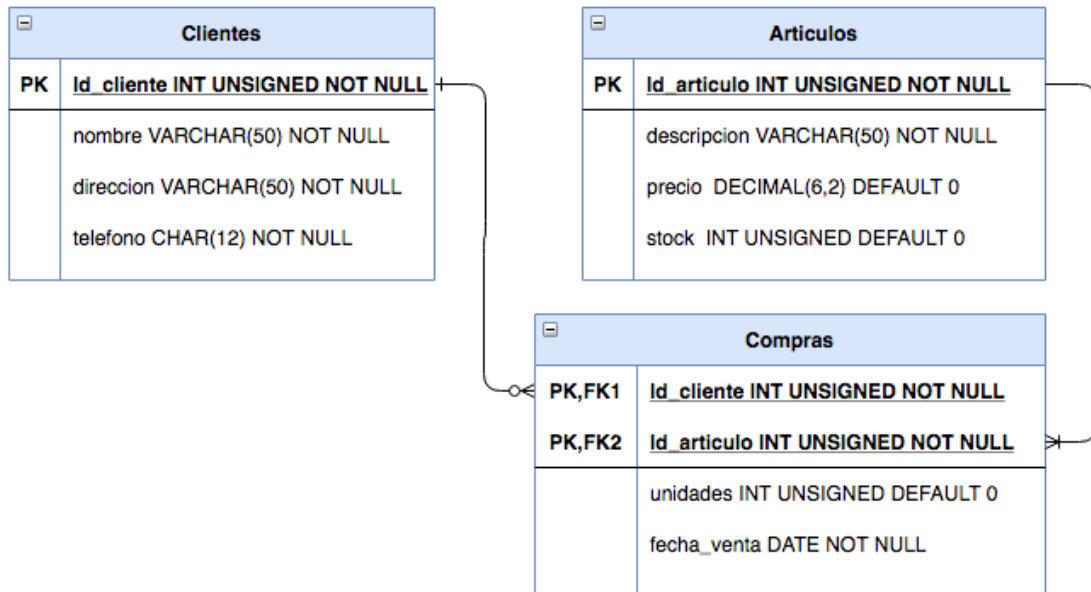
Veamos el siguiente ejemplo de modelo entidad-relación:



Obtenemos las tablas a partir del modelo ER anterior:

CLIENTES(Id_cliente, Nombre, Dirección, Teléfono)
 ARTICULOS(Cod_articulo, Descripción, Precio, Stock)
 COMPRA(Id_cliente (FK), Cod_articulo (FK), Unidades, Fecha_venta)

A partir de las tablas obtenidas creamos el modelos relacional de las mismas con el dominio asignado a cada uno de los campos:



Creamos el modelo físico de las tablas a través del lenguaje SQL, visto en temas anteriores:

CREATE TABLE IF NOT EXISTS CLIENTES (

Id_cliente INT UNSIGNED NOT NULL PRIMARY KEY,
 nombre VARCHAR(50) NOT NULL,
 direccion VARCHAR(50) NOT NULL,
 telefono CHAR(12) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE IF NOT EXISTS ARTICULOS (

Id_articulo INT UNSIGNED NOT NULL,
 descripcion VARCHAR(50) NOT NULL,
 precio DECIMAL(6, 2) DEFAULT 0,
 stock INT UNSIGNED DEFAULT 0,
 PRIMARY KEY (Id_articulo)) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

CREATE TABLE IF NOT EXISTS COMPRAS(

Id_cliente INT UNSIGNED NOT NULL,
 Id_articulo INT UNSIGNED NOT NULL,
 unidades INT UNSIGNED DEFAULT 0,
 fecha_venta DATE NOT NULL,

```

PRIMARY KEY (Id_cliente, Id_articulo),
INDEX (Id_cliente), INDEX (Id_articulo),
FOREIGN KEY (Id_cliente)
    REFERENCES CLIENTES(Id_cliente),
CONSTRAINT fk_articulo FOREIGN KEY (Id_articulo)
    REFERENCES ARTICULOS( Id_articulo )
ON UPDATE CASCADE
ON DELETE RESTRICT ) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

```



4. Normalización de modelos relacionales

Habitualmente, el diseño de una base de datos termina en el paso del modelo entidad-relación al modelo relacional. No obstante, siempre que se diseña un sistema, no solo una base de datos, sino también cualquier tipo de solución informática, se ha de medir la calidad de la misma, y si no cumple determinados criterios de calidad, hay que realizar, de forma iterativa, sucesivos refinamientos en el diseño para alcanzar la calidad deseada. Uno de los parámetros que mide la calidad de una base de datos es la *forma normal* en la que se encuentra. El proceso de obligar a los atributos de un diseño a cumplir ciertas formas normales se llama **normalización**.

Las formas normales pretenden alcanzar dos objetivos:

1. Almacenar en la base de datos cada hecho solo una vez, es decir, evitar la redundancia de datos. De esta manera se reduce el espacio de almacenamiento.
2. Que los hechos distintos se almacenen en sitios distintos. Esto evita ciertas anomalías a la hora de operar con los datos.

Hay definidas 6 formas normales, cada una agrupa a las anteriores, de forma que, por ejemplo, la forma normal 3 cumple la forma normal 2 y la forma normal 1.

 salesianos COLEGIO SAN JUAN BOSCO LA CUESTA	Bases de datos Tema 4	 EDUCATIA 130 años 1887-2017 ACREDITADO POR ENAC
	Diseño de bases de datos relacionales Modelo Relacional	

4.1. Tipos de dependencias

Vamos a desarrollar los conceptos sobre los que se basa el análisis de dependencias entre atributos, que se lleva a cabo en el proceso de normalización antes indicado.

- **Dependencia funcional:** Dados los atributos A y B, se dice que B depende funcionalmente de A, sí y solo sí, para cada valor de A solo puede existir un valor de B. La dependencia funcional siempre se establece entre atributos de una misma tabla. El atributo A se denomina determinante, ya que A determina el valor de B. Para representar esta dependencia funcional utilizamos la siguiente notación: $A \rightarrow B$. Hay que indicar que A y B podrían ser un solo atributo o un conjunto de ellos.

PRODUCTOS(CódigoProducto, Nombre, Precio, Descripción)

CódigoProducto \rightarrow Nombre, puesto que un código de producto solo puede tener asociado un único nombre, dicho de otro modo, a través del código de producto se localiza un único nombre.

- **Dependencia funcional completa:** Dados los atributos A1, A2, ...Ak y B, se dice que B depende funcionalmente de forma completa de A1, A2, ...Ak, sí y solo sí B depende funcionalmente del conjunto de atributos A, A2, ...Ak, pero no de ninguno de sus posibles subconjuntos. representar esta dependencia funcional utilizamos la siguiente notación: $A \Rightarrow B$

PRODUCTOS(CódigoProducto, CódigoProveedor, Cantidad, FechaCompra)

CódigoProducto, CódigoProveedor \Rightarrow FechaCompra, puesto que la FechaCompra es única para la combinación de CódigoProducto y CódigoProveedor (se puede hacer un pedido al día de cada producto del mismo proveedor), y sin embargo, se pueden hacer varios pedidos del mismo producto a diferentes proveedores, es decir CódigoProducto \nrightarrow Fecha.

- **Dependencia funcional transitiva:** Dados tres atributos A, B y C, se dice que existe una dependencia transitiva entre A y C, si B depende funcionalmente de A y C depende funcionalmente de B. Es decir, $A \rightarrow B$, $B \rightarrow C$ y $B \nrightarrow A$. Se dice que A depende transitivamente de C, o que, $A \twoheadrightarrow C$

PRODUCTOS(CódigoProducto, Nombre, Fabricante, País)

CódigoProducto \rightarrow Fabricante

Fabricante \rightarrow País

CódigoProducto → País, es decir, CódigoProducto depende transitivamente de País.

4.2. Primera forma normal (1FN)

En esta forma normal se prohíbe que una tabla haya atributos que puedan tomar más de un valor. Esta forma es inherente al modelo relacional, puesto que las tablas gestionadas por un SGBD relacional, están construidas de esta forma. Dicho de otra forma, una tabla estará en 1FN si los atributos no clave, dependen funcionalmente de la clave.

Películas	Año	Actor	Películas	Año	Actor
La amenaza fantasma	1999	Ewan McGregor	La amenaza fantasma	1999	Ewan McGregor
		Liam Neeson	La amenaza fantasma	1999	Liam Neeson
		Natalie Portman	La amenaza fantasma	1999	Natalie Portman
Blade Runner	1982	Harrison Ford	Blade Runner	1982	Harrison Ford
		Sean Young	Blade Runner	1982	Sean Young
		Rutger Hauer	Blade Runner	1982	Rutger Hauer
Avatar	2009	Sam Worthington	Avatar	2009	Sam Worthington
		Zoe Saldana	Avatar	2009	Zoe Saldana
		Sigourney Weaver	Avatar	2009	Sigourney Weaver

No cumple la 1FN



Cumple la 1FN

¿Cómo se normaliza a Primera Forma Normal?

- Se crea, a partir de la tabla inicial, una nueva tabla cuyos atributos son los que representan dependencia funcional de la clave primaria. La clave de esta tabla será la misma clave primaria de la tabla inicial.
- Con los atributos restantes se crea otra tabla y se elige entre ellos uno que será la clave primaria de dicha tabla. Comprobaremos si esta segunda tabla está en 1FN. Si es así la tabla inicial ya estará normalizada a 1FN y el proceso termina. Si no está en 1FN, tomaremos la segunda tabla como tabla inicial y repetiremos el proceso.

4.3. Segunda forma normal (2FN)

Un diseño se encuentra en 2FN si está en 1FN y además, cada atributo que no forma parte de la clave tiene dependencia completa de la clave principal. Es obvio que una tabla que esté en 1FN y cuya clave esté compuesta por un único atributo, estará en 2FN.

	Bases de datos Tema 4	
	Diseño de bases de datos relacionales Modelo Relacional	

Ejemplo:

COMPRAS(CódigoProducto, CódigoProveedor, NombreProducto, Cantidad, FechaCompra)

CódigoProducto → NombreProducto, por tanto, al no ser dependencia funcional completa, **NO** está en 2FN.

¿Cómo se normaliza a Segunda Forma Normal?

- Se crea, a partir de la tabla inicial, una nueva tabla con los atributos que dependen funcionalmente de forma completa de la clave. La clave de esta tabla será la misma clave primaria de la tabla inicial. Esta tabla ya estará en 2FN.
- Con los atributos restantes, se crea otra tabla que tendrá por clave el subconjunto de atributos de la clave inicial de los que depende de forma completa. Se comprueba si esta tabla está en 2FN. Si es así, la tabla inicial ya está normalizada y el proceso termina. Si no está en 2FN tomamos esta segunda tabla como tabla inicial y repetimos el proceso.

4.4. Tercera forma normal (3FN)

Un diseño se encuentra en 3FN si está en 2FN y además, no hay ningún atributo no clave que depende de forma transitiva de la clave primaria.

Ejemplo:

PRODUCTOS(CódigoProducto, Nombre, Fabricante, País)

CódigoProducto → Fabricante

Fabricante → País

CódigoProducto – ➔ País

País depende transitivamente de CódigoProducto, por tanto, **NO** está en tercera forma normal.

¿Cómo se normaliza a Tercera Forma Normal?

- Se crea, a partir de la tabla inicial, una nueva tabla con los atributos que no poseen dependencias transitivas de la clave primaria. Esta tabla ya estará en 3FN.
- Con los atributos restantes, se crea otra tabla con los atributos no clave que intervienen en la dependencia transitiva, y se elige uno de ellos como clave primaria, si cumple los requisitos para ello. Se comprueba si esta tabla está en

3FN. Si es así, la tabla inicial ya está normalizada y el proceso termina. Si no está en 3FN tomamos esta segunda tabla como tabla inicial y repetimos el proceso.

4.5. Forma normal Boyce-Codd (FNBC)

Una tabla está en Forma Normal de Boyce-Codd (FNBC o BCFN) si está en 3FN y además, todo determinante de la tabla, es una clave candidata. Un determinante será todo atributo simple o compuesto del que depende funcionalmente de forma completa algún otro atributo de la tabla.

Ejemplo:

NOTAS(**DNIAlumno**, **DNIProfesor**, NombreProfesor, Nota)

DNIProfesor → NombreProfesor

NombreProfesor → DNIProfesor

DNIProfesor, DNIAlumno → Nota

En este caso, la tabla está en 3FN porque no hay dependencias funcionales transitivas, y sin embargo no está en FNBC porque NombreProfesor y DNIProfesor son implicantes, y no son clave candidatas. Para obtener la tabla en FNBC habría que quitar de la tabla los atributos DNIProfesor y NombreProfesor.

Ejemplo de normalización:

Sea la siguiente tabla. Se pide normalizarla a FNBC.

COMPRAS(**cod_compra**, **cod_prod**, nomb_prod, fecha, cantidad, precio, fecha_rec, cod_prov, nomb_prov, tfno)

Comprobamos 1FN:

Las tablas COMPRAS está en 1FN ya que todos sus atributos son atómicos y todos los atributos no clave dependen funcionalmente de la clave.

Comprobamos 2FN:

Nos preguntamos, ¿todo atributo depende de todo el conjunto de atributos que forman la clave primaria, o solo de parte?. Como vemos, existen atributos que dependen solo de una parte de la clave, por lo que esta tabla no está en 2FN.



Veamos las dependencias:

Cod_prod → nomb_prod, y cod_prod es parte de la clave primaria.

Al no estar en 2FN, hemos de descomponer la tabla COMPRAS en:

COMPRA1(**cod_compra**, **cod_prod**, fecha, cantidad, precio, fecha_rec, cod_prov, nomb_prov, tfno)

PRODUCTO(**cod_prod**, Nomb_prod)

	Bases de datos Tema 4	
	Diseño de bases de datos relacionales Modelo Relacional	

Una vez hecha esta descomposición, ambas tablas están en 2FN. Todos los atributos no clave dependen de toda la clave primaria.

Comprobamos 3FN:

PRODUCTO está en 3FN, ya que por el número de atributos que tiene no puede tener dependencias transitivas.

¿COMPRA1 está en 3FN? Hemos de preguntarnos si existen dependencias transitivas entre atributos no clave.

Veamos las dependencias:

$\text{Cod_prod} \rightarrow \text{nomb_prod}$

$\text{Cod_prod} \rightarrow \text{tfno}$

(siendo cod_prov el código el proveedor y nomb_prov el nombre del proveedor)

COMPRA1 no está en 3FN porque existen dependencias transitivas entre atributos no clave, por tanto hemos de descomponer:

COMPRA2(cod_compra, cod_prod, fecha, cantidad, precio, fecha_rec, cod_prov)

PROVEEDOR(cod_prov, nomb_prov, tfno)

Comprobamos FNBC:

PRODUCTO está en FNBC, ya que está en 3FN y todo determinante es clave candidata.

COMPRA2 está en FNBC, ya que está en 3FN y todo determinante es clave candidata.

PROVEEDOR está en FNBC, ya que está en 3FN y todo determinante es clave candidata.

La tabla inicial COMPRAS queda normalizada hasta FNBC del siguiente modo:

PRODUTO(cod_prod, nomb_prod)

COMPRA2(cod_compra, cod_prod, fecha, cantidad, precio, fecha_rec, cod_prov)

PROVEEDOR(cod_prov, nomb_prov, tfno)