



1º DAW

U.T. 3

Diseño lógico: el modelo relacional



Pablo Berciano Posada

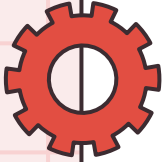


Tabla de contenidos

01

**El modelo
relacional**

02

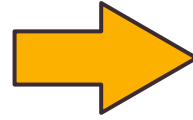
**Transformación de
entidades y atributos**

03

**Transformación de
relaciones**

04

**Teoría de la
normalización**



01

El modelo relacional





El modelo relacional

Edgar Frank Codd definió las bases del **modelo relacional** a finales de los 60. Codd quería conseguir los siguientes objetivos con su modelo:

- **Independencia física.** La forma de almacenar los datos, no debe influir en su manipulación lógica. Si el almacenamiento físico sufre algún cambio los usuarios no deben enterarse ya que seguirán funcionando sus aplicaciones.
- **Independencia lógica.** Las aplicaciones que utilizan la base de datos no deben ser modificadas aunque se modifiquen elementos de la base de datos.
- **Flexibilidad.** La base de datos ofrece fácilmente distintas vistas en función de los usuarios y aplicaciones.
- **Uniformidad.** Las estructuras lógicas siempre tienen una única forma conceptual.
- **Sencillez.**

Estas bases dieron lugar a las **12 reglas de Codd**, requisitos de un SGBD para ser relacional.





Las 12 reglas de Codd



1. **Representación de la información:** toda información debe representarse en forma de tabla, principio básico del modelo relacional.
2. **Acceso garantizado:** todo dato debe ser accesible mediante una combinación de un nombre de tabla, un valor de su clave y el nombre de una columna.
3. **Tratamiento sistemático de valores nulos:** estos valores han de ser tratados sistemáticamente por el sistema, el cual ha de ofrecer las posibilidades necesarias para su tratamiento.
4. **Catálogo en línea basado en el modelo relacional:** la representación de la descripción de la BD debe ser igual a la de los otros datos, y su acceso debe poder realizarse por medio del mismo lenguaje relacional. El Modelo de datos para la metabase debe ser también relacional.



Las 12 reglas de Codd



5. Sublenguaje de datos completo: debe existir un lenguaje que permita un completo manejo de la BD (definición y manipulación de datos, definición de vistas, restricciones de integridad, autorizaciones y gestión de transacciones).

6. Actualización de vista: toda vista teóricamente actualizable debe poder ser actualizada por el sistema.

7. Inserción, actualización y eliminación de alto nivel: todas las operaciones de manipulación de datos deben operar sobre conjuntos de filas.

8. Independencia física de los datos: el acceso lógico a los datos debe mantenerse incluso cuando cambien los métodos de acceso o la forma de almacenamiento.



Las 12 reglas de Codd



9. Independencia lógica de los datos: los programas de aplicación no deben verse afectados por cambios realizados en el esquema de la BD.

10. Independencia de integridad: las reglas de integridad de una BD deben ser definibles por medio del sublenguaje de datos relacional y habrán de almacenarse en el DD de la BD, no en los programas de aplicación.

11. Independencia de distribución: debe existir un sublenguaje de datos que pueda soportar BD distribuidas sin alterar los programas de aplicación cuando se distribuyan los datos.

12. Regla de la no subversión: si un SGBD soporta un lenguaje de bajo nivel que permita el acceso fila a fila, éste no puede utilizarse para saltarse las reglas de integridad expresadas por medio del lenguaje de más alto nivel.

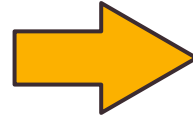


Diseño lógico: modelo relacional

El objetivo del **diseño lógico** es **convertir los esquemas conceptuales en un esquema lógico** que se ajuste al **modelo de SGBD** sobre el que se vaya a implementar el sistema. En nuestro caso el modelo elegido es el modelo relacional.

Ahora, en la fase de diseño lógico, debe **traducirse** el diseño conceptual a una estructura que pueda ser manejada por el ordenador. Al modelo de datos obtenido en el análisis se le denomina **modelo conceptual de datos** y al modelo obtenido en el diseño se le denomina **modelo lógico de datos**.

Para realizar este proceso iremos **transformando** las **entidades, atributos y relaciones** obtenidas en el el modelo E/R.



02

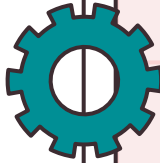
Transformación de entidades y atributos





Transformación de las entidades

Todas las **entidades fuertes** en el modelo E/R se transforman en **tablas** en el modelo relacional, manteniendo el número y tipo de los atributos. El **identificador de la entidad** se convierte en la **clave primaria**, se señala con un subrayado.

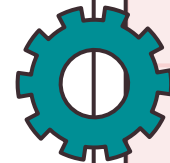


`HOTEL(código, nombre, dirección, num_plazas, ciudad)`

Las **entidades débiles** también se convierten en **tablas** en el modelo relacional, manteniendo el número y tipo de atributos. Aunque su **clave primaria** puede estar formada por la composición y su **identificador** y el **identificador de la entidad fuerte** de la cual depende.



Transformación de los atributos

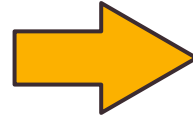


Todo **atributo simple** o **calculado** se transforma en **columna** de una tabla. Pero hay ciertos tipos de atributos que siguen otro tipo de transformación:

- Los **atributos compuestos** transforman los **campos** en los que se componen en nuevas **columnas** de la tabla, es decir se toman solo los atributos 'hoja'.
- Los **atributos multivaluados** generan una **nueva tabla** con dos columnas: el identificador único, llamado ahora clave primaria, de la tabla de la que surgen propagado como clave ajena y el valor del campo multivaluado.

Para los **atributos de las relaciones** existen dos casos:

- Si la relación **se resuelve propagando** la clave primaria, los atributos de la relación **se propagan junto con la clave primaria** de la entidad fuerte.
- Si la relación **se resuelve creando una tabla**, sus atributos se transforman en **columnas de la tabla** generada por dicha relación.



03

Transformación de relaciones





Transformación de relaciones

A la hora de **transformar una relación** del modelo E/R al modelo relacional vamos a tener **dos opciones**, entre las que elegiremos según las características de la relación.

- La relación puede resolverse **propagando la clave primaria**. En este caso lo que haremos será añadir la clave primaria de la entidad fuerte como una clave ajena en la tabla de la entidad débil, añadiendo tantas columnas como campos forman la clave.
- La relación puede resolverse **creando una tabla**. En este caso se crea una tabla totalmente nueva que represente la relación, y cuya clave primaria será la unión de las claves primarias de las entidades participantes.

Relaciones uno a uno (1:1)

Atendiendo a la **participación**, si las entidades que se asocian poseen participaciones **(0,1)**, en este caso la **relación 1:1** se transformará en una **tabla**.

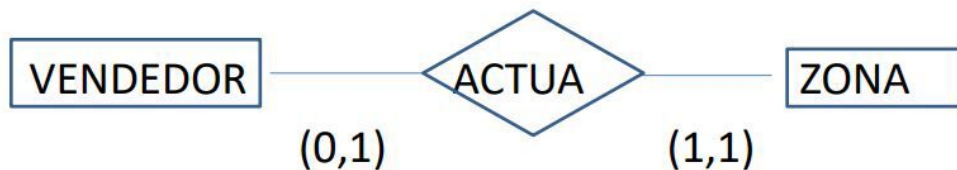


↓

VENDEDOR(Cod-vend, Nom-vend, Tel-vend)
ZONA(Cod-zon, Nom-zon)
ACTUA(Cod-vend, Cod-zon)

Relaciones uno a uno (1:1)

Si una de las entidades que participa en la relación posee participación (0,1), mientras que en la otra es (1,1), **se propaga la clave** de la entidad con participaciones (1,1) a la tabla resultante de la entidad de participaciones (0,1). Si ambas entidades tienen participaciones (1,1), se puede propagar la clave de cualquiera de ellas a la tabla resultante de la otra.



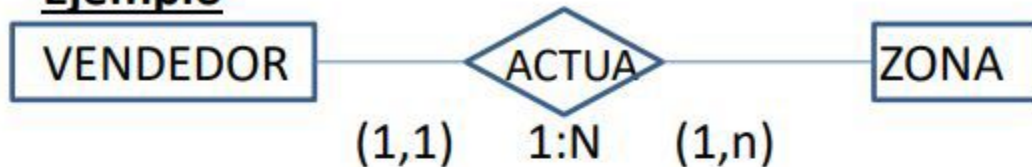
↓

VENDEDOR(Cod-vend, Nom-vend, Tel-vend, **Cod-zon**)
ZONA(Cod-zon, Nom-zon)

Relaciones uno a muchos (1:N)

En la transformación de **relaciones 1:N** se puede optar por las dos soluciones posibles. **Normalmente se propaga la clave.** Se propaga el atributo principal de la entidad que tiene participación máxima 1, llamada entidad fuerte, a la que tiene participación máxima n, llamada entidad débil.

Ejemplo

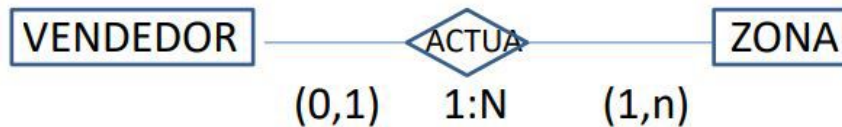


VENDEDOR(Cod-vend, Nom-vend, Tel-vend)

ZONA(Cod-zon, Nom-zon, **Cod-vend**)

Relaciones uno a muchos (1:N)

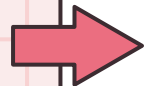
A la hora de transformar relaciones 1:N también se puede crear una nueva tabla, recomendable cuando la participación es (0,1) y (1,n) y en el caso en que la relación tiene atributos propios, ya que pasan a formar parte de la nueva tabla.



VENDEDOR(Cod-vend, Nom-vend, Tel-vend)

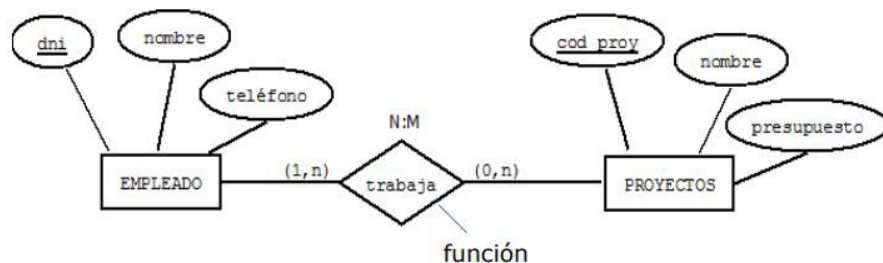
ZONA(Cod-zon, Nom-zon)

ACTUA(Cod-zon, Cod-vend)



Relaciones muchos a muchos (N:M)

En el caso de las relaciones de muchos a muchos N:M se construye una nueva tabla correspondiente a la relación.



EMPLEADO(dni, nombre, teléfono)

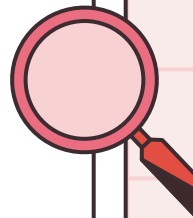
PROYECTO (cod_proy, nombre, presupuesto)

TRABAJA (dni,cod_proy, función)

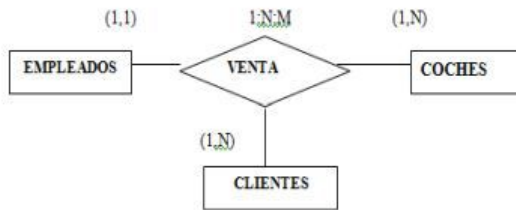


Relaciones N-aria

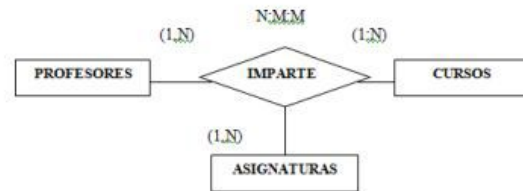
En este tipo de relaciones se **agrupan 3 o más entidades**, y para pasar al modelo de datos relacional cada entidad se convierte en tabla, así como la relación, que va a contener los atributos propios de ella más las claves de todas las entidades. La clave de la tabla resultante será la concatenación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

- Si en la relación **todas las entidades tienen participación máxima N**, la clave de la tabla resultante es la unión de las claves de las entidades que relaciona.
 - Si en la relación **una de las entidades tienen una participación 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.
- 

Relaciones N-aria



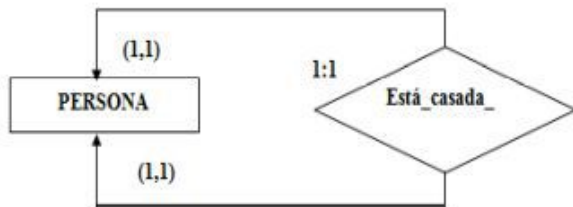
CLIENTES (Cod-cliente, nombre, teléfono)
EMPLEADOS (Cod-empleado, nombre, teléfono, salario, fecha-alta)
COCHES (Cod-coche, matricula, modelo, precio)
VENTA (Cod-coche, Cod-cliente, Cod-empleado, forma-pago, fecha-venta)



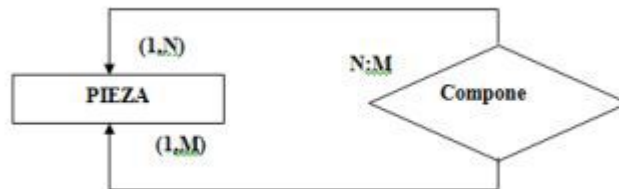
PROFESORES (Cod-profesor, dirección, nombre, teléfono, especialidad)
CURSOS (Cod-curso, descripción, nivel, turno)
ASIGNATURAS (Cod-asignatura, nombre)
IMPARTE (Cod-profesor, Cod-curso, Cod-asignatura)

Relaciones reflexivas

Las **relaciones reflexivas** son relaciones binarias pero con la particularidad de que solamente tienen una tabla. Por lo tanto se tratarán como el resto de relaciones binarias.



PERSONA (DNI, Nombre, Apellidos, Dirección, Teléfono,..., DNI-cónyuge)

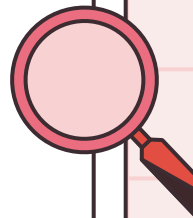


PIEZA (Cod-pieza, Descripción, Tamaño, Color)
COMPONE-PIEZA (Cod-pieza-con, Cod-pieza)

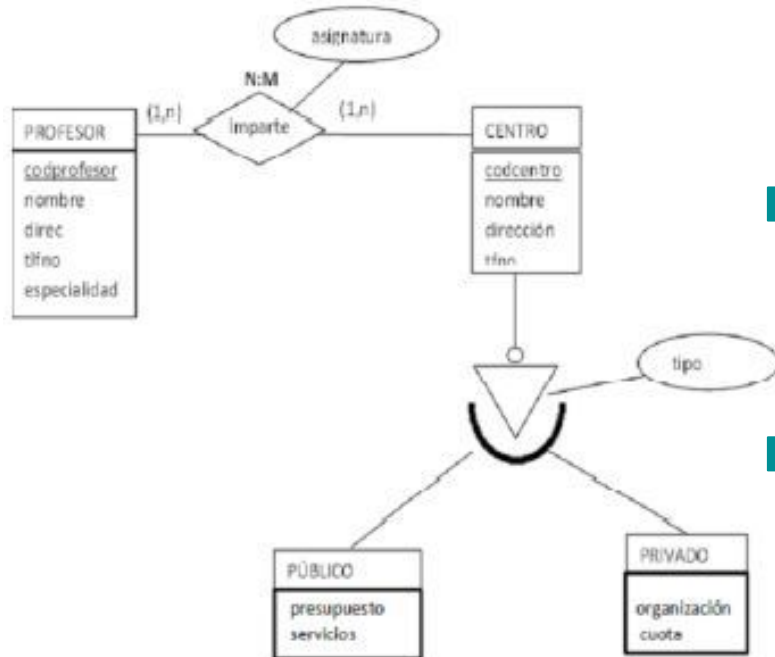


Transformación de jerarquías

El modelo relacional no dispone de mecanismos para la representación de las **relaciones jerárquicas**, así pues, las relaciones jerárquicas se tienen que eliminar. Como son un caso especial, se pueden dar algunas reglas que sirvan de referencia:

- **Integrar todas las entidades en una única eliminando los subtipos.** Esta entidad contendrá todos los atributos de la entidad supertipo, todos los de las subtipos y un atributo con el tipo.
 - **Eliminación de la jerarquía.** La relación jerárquica se transforma en tantas relaciones como subtipos tenga, manteniéndose las relaciones que tenían el supertipo y los subtipos. En la entidad que anteriormente era la supertipo se añade el tipo. Si la relación es inclusiva se crea una nueva tabla cuya clave estará formada por la clave de la de la entidad supertipo y el tipo . El esquema mantiene la representación de las relaciones existentes entre el supertipo y los subtipos de entidad.
- 

Transformación de jerarquías



PROFESOR(codprofesor, nombre, direc, tfno, especialidad)
CENTRO(codcentro, nombre, dirección, tfno, tipo, presupuesto,
servicios, cuota, presupuesto)
IMPARTE(codprofesor, codcentro, asignatura)

PROFESOR(codprofesor, nombre, direc, tfno, especialidad)
CENTRO(codcentro, nombre, dirección, tfno, tipo)
PÚBLICO(codcentro, presupuesto, servicios)
PRIVADO(codcentro, organización, cuota)
IMPARTE(codprofesor, codcentro, asignatura)



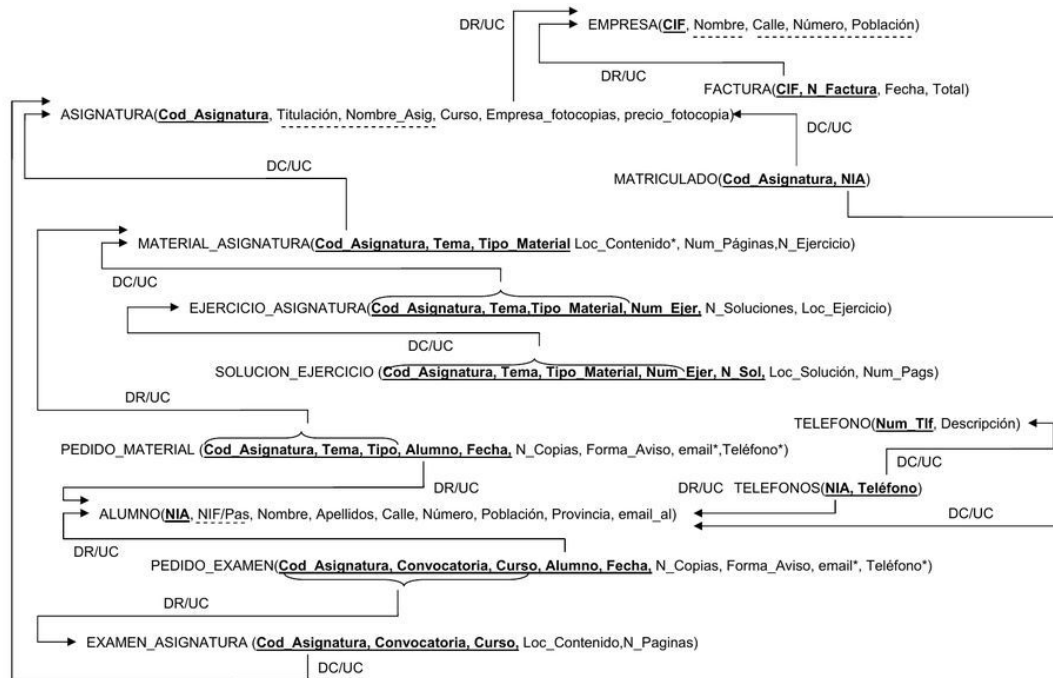
Grafo relacional



Un esquema relacional se representará mediante un **grafo**. Se trata de un grafo **dirigido** cuyos **nod**os son las **tablas** de la base de datos y los **arcos** representan las **restricciones de clave ajena**. Las convenciones empleadas para la representación de este grafo son:

- El **nombre de las tablas** está representado en **mayúsculas**.
- Las **claves primarias** aparecen **subrayadas**.
- Las **claves alternativas** están representadas en letra **cursiva**.
- Las **claves ajenas** aparecen **negrita** y referencian a la relación en la que son clave primaria mediante una **flecha**.

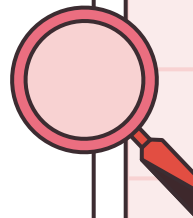
Grafo relacional

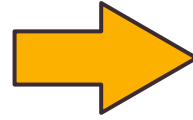




Acciones referenciales

Debido a que el SGBD hace cumplir las **restricciones de referencia**, se debe garantizar la **integridad** de los datos si las filas de la tabla de referencia se van a eliminar o van a ser actualizadas. Si todavía existen filas dependientes en tablas referenciadas, esas referencias tienen que ser consideradas. Hay 5 posibles **acciones referenciales** diferentes que se ejecutarán en estos casos:

- **CASCADE**: se replica la acción realizada en las tablas referenciadas.
 - **RESTRICT**: si hay una referencia en alguna tabla hija al registro a tratar, se anula la acción.
 - **NO ACTION**: similar a RESTRICT, realiza la comprobación en otro momento.
 - **SET NULL**: el valor de la referencia afectada se cambia a NULL.
 - **SET DEFAULT**: el valor de la referencia afectada se cambia por un valor por defecto.
- 



04

Teoría de la normalización



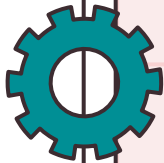


Teoría de la normalización

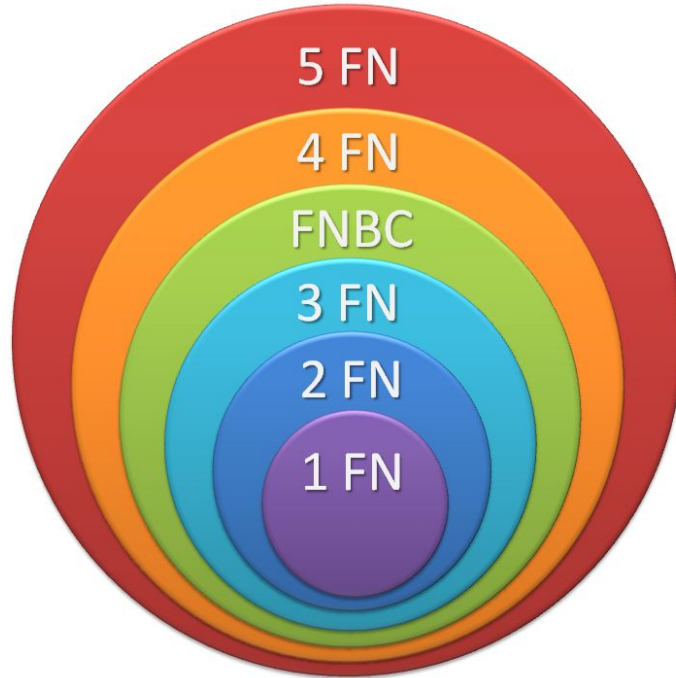
El proceso de **normalización** de bases de datos consiste en aplicar una serie de **reglas** a las relaciones obtenidas tras el paso del modelo entidad-relación al modelo relacional con la finalidad de:

- **Evitar la redundancia** de los datos.
- **Evitar problemas de actualización** de los datos en las tablas.
- **Proteger la integridad** de los datos.

Cada regla que se cumple aumenta el **grado de normalización**. Si una regla no se cumple, la relación se debe descomponer en varias relaciones que sí la cumplan. Conforme se va avanzando en la normalización, las relaciones tienen un formato más estricto y, por lo tanto, son menos vulnerables a las anomalías de actualización.



Teoría de la normalización





Dependencias funcionales



Dependencia funcional: Se dice que un atributo Y depende funcionalmente de otro X de la misma relación si y sólo si, cada valor de X tiene asociado en todo momento un único valor de Y, lo que se representa como:

$$X \rightarrow Y \quad \left\{ \begin{array}{l} Y \text{ depende funcionalmente de } X \\ X \text{ determina a } Y \end{array} \right.$$

X = determinante
Y = implicado

Ejemplo:

el **nombre de una persona** depende funcionalmente del **DNI**
es decir para un DNI concreto sólo hay un nombre posible.

DNI \rightarrow NombrePersona

DNI	NombrePersona	FechaNacimiento	EstadoCivil
80094920	Ángel Morales	25-7-1977	soltero
80454659	María López	12-4-1967	casado



Dependencias funcionales

Dependencia funcional completa: Si el atributo X es compuesto $X(X1, X2)$, se dice que Y tiene dependencia funcional completa o plena respecto de X si depende funcionalmente de X pero no depende de ningún subconjunto del mismo.

$X \rightarrow Y$

$X1 \not\rightarrow Y$

$X2 \not\rightarrow Y$

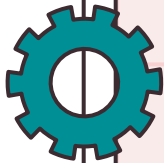
Un empleado puede trabajar en varios proyectos, realizando una sola función en cada uno de ellos (consultor, analista, programador, etc.) aunque ésta podrá ser distinta según el proyecto, tendríamos que **Función depende funcionalmente y de forma completa** de **DNIEmpleado+CodProyecto** y además no depende de DNIEmpleado o CodProyecto por separado.

(DNIEmpleado, CodProyecto) \rightarrow Función

DNIEmpleado	CodProyecto	Función
80094920	PROY-53	ANALISTA
80454659	PROY-53	PROGRAMADOR
80094920	PROY-54	PROGRAMADOR

DNIEmpleado $\not\rightarrow$ Función

CodProyecto $\not\rightarrow$ Función





Dependencias funcionales

Dependencia funcional transitiva: Sean X,Y,Z tres atributos de una relación, si se cumple que Z depende de Y e Y depende de X, se dice que Z tiene una dependencia transitiva respecto de X a través de Y, lo que se representa como :

$$X - \rightarrow Z$$

Ejemplo:

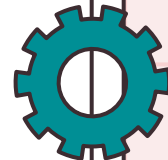
Consideremos una relación que asocia **cursos** con **departamento** y con **escuela**.

Curso \rightarrow **Departamento**

Departamento \rightarrow **Escuela**

Curso - \rightarrow **Escuela**

Curso	Departamento	Escuela
canto	música	ARTES
piano	música	ARTES
dibujo	plástica	ARTES
cerámica	plástica	ARTES
hidráulica	Ingeniería civil	INGENIERÍA





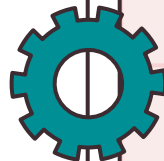
Primera Forma Normal (1FN)

Una relación está en primera forma normal si, y sólo si los valores que componen cada atributo de una tupla son atómicos, es decir, cada atributo de la relación toma un único valor del dominio correspondiente, o lo que es lo mismo no existen grupos repetitivos.

La tabla siguiente no cumple la primera forma normal, porque el atributo Idiomas tiene 3 datos para una misma fila.

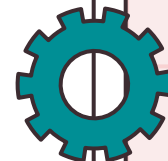
<u>DNI</u>	NombrePersona	FechaNacimiento	Idiomas
80094920	Ángel Morales	25-7-1977	Español Inglés Alemán
80454659	María López	12-4-1967	Español

Hay tres formas de eliminar los grupos repetitivos.





Primera Forma Normal (1FN)



<u>DNI</u>	NombrePersona	FechaNacimiento	<u>Idiomas</u>
80094920	Ángel Morales	25-7-1977	Español
80094920	Ángel Morales	25-7-1977	Inglés
80094920	Ángel Morales	25-7-1977	Alemán
80454659	María López	12-4-1967	Español

<u>DNI</u>	NombrePersona	FechaNacimiento
80094920	Ángel Morales	25-7-1977
80454659	María López	12-4-1967

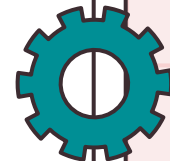
<u>DNI</u>	<u>Idiomas</u>
80094920	Español
80094920	Inglés
80094920	Alemán
80454659	Español

<u>DNI</u>	NombrePersona	FechaNacimiento	Idioma1	Idioma2	Idioma3
80094920	Ángel Morales	25-7-1977	Español	Inglés	Alemán
80454659	María López	12-4-1967	Español		



Segunda Forma Normal (2FN)

Una relación está en segunda forma normal si, y sólo si, está en 1FN y, además, cada atributo que no forma parte de la clave primaria es completamente dependiente de la clave **primaria**. La 2FN se aplica a las relaciones que tienen claves primarias compuestas por dos o más atributos. Si una relación está en 1FN y su clave primaria es simple (tiene un solo atributo), entonces también está en 2FN.



Ejemplo: El atributo **Descripción** depende funcionalmente de **CodProyecto**.

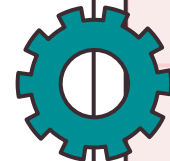
<u>DNIEmpleado</u>	<u>CodProyecto</u>	Descripción	Función
80094920	PROY-53	Financiera	ANALISTA
80454659	PROY-53	Financiera	PROGRAMADOR
80094920	PROY-54	Importación	PROGRAMADOR

Está en 1FN porque no tiene grupos repetitivos.
No está en 2FN porque hay atributos que no dependen completamente de la clave primaria.



Segunda Forma Normal (2FN)

Para pasar una relación en 1FN a 2FN hay que **eliminar las dependencias parciales** de la clave primaria. Para ello, se **eliminan los atributos** que son funcionalmente dependientes y se ponen en una **nueva tabla** con una copia de su determinante, los atributos de la clave primaria de los que dependen.



<u>CodPedido</u>	<u>CodProducto</u>	DescripProd	Cantidad	Precio
800	Lib-18	cuaderno	45	8.30
800	Lib-20	carpeta	25	18.50



<u>CodPedido</u>	<u>CodProducto</u>	Cantidad
800	Lib-18	45
800	Lib-20	25

Está en 2FN porque todo atributo no clave depende completamente de la clave primaria
 $CodPedido + CodProducto \rightarrow Cantidad$

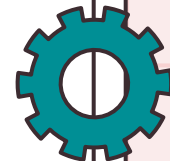
<u>CodProducto</u>	DescripProd	Precio
Lib-18	cuaderno	8.30
Lib-20	carpeta	18.50

Está en 2FN porque todo atributo no clave depende completamente de la clave primaria.
 $CodProducto \rightarrow DescripProd$
 $CodProducto \rightarrow Precio$



Tercera Forma Normal (3FN)

Una relación está en tercera forma normal si, y sólo si, está en 2FN y cada atributo no clave de la relación no depende funcionalmente de forma transitiva de la clave primaria de esa **relación**. Para convertir una tabla que no está en 3FN se realizará una proyección de la clave a los elementos que no tengan dependencia funcional transitiva y otra tabla con una nueva clave a los elementos que anteriormente tenían esta dependencia.



<u>CodPed</u>	FechaPedido	CodProveedor	NombreProveedor	DirecciónProveedor
800	25-7-2010	PR-10	Ángel Morales	Plaza Constitución, 1
810	27-7-2010	PR-10	Ángel Morales	Plaza Constitución, 1



<u>CodPed</u>	FechaPedido	<u>Cod_Proveedor</u>
800	25-7-2010	PR-10
810	27-7-2010	PR-10

<u>CodProveedor</u>	NombreProveedor	DirecciónProveedor
PR-10	Ángel Morales	Plaza Constitución, 1



Más Formas Normales

En entornos profesionales se considera que se tiene un buen diseño si se encuentra en 3FN. Existen **más formas normales** que, al igual que las anteriores, **requieren que se cumpla la anterior** forma normal. En orden son:

- La **forma normal de Boyce-Codd (FNBC)**: una relación está en FNBC si, y sólo si, está en 3FN y todo determinante es clave candidata. Es decir, si todas las partes izquierdas de las dependencias son claves candidatas.
- La **cuarta forma normal (4FN)**: una relación está en 4FN si, y sólo si, está en FNBC y siempre que existe un hecho multivaluado ($A \twoheadrightarrow B$) todos los atributos de la relación son funcionalmente dependientes de A. Es decir, no existe entre sus atributos ninguna dependencia multivaluada que no sea combinación de dependencias funcionales.
- La **quinta forma normal (5FN)**: una relación está en 5FN si, y sólo si, está en 4FN y toda dependencia en ella es una consecuencia de sus claves candidatas. Es decir, su información no puede ser reconstruida si la dividimos en fragmentos más pequeños.

