Bases de Datos

Tema 4: Normalización

 objetivos

El proceso de normalización consiste en optimizar las tablas para eliminar toda redundancia y mal diseño posible.

Debemos ser conscientes, sin embargo, que si la base de datos está bien diseñada desde el principio, estará prácticamente normalizada, más si tenemos en cuenta que la Forma Normal Boyce-Codd elimina una redundancia muy rara de encontrar, y que la mayor parte de las ocasiones no nos molestará esta poquito de redudància (antes al contrario, nos supondría más molestia tener otra tabla).

La utilidad real de la normalización será cuando partimos de una base de datos directamente importada de un sistema de archivos (un archivo → una mesa), donde ahora sí normalmente estará muy mal diseñado.

Los objetivos mínimos serán, por tanto:

- Entender el problema de la redundancia y el mal diseño de una Base de Datos.
- Detectar cuando una tabla no está en Primera Forma Normal.
- Saber pasar una tabla no normalizada en Primera Forma Normal.
- Detectar todas las dependencias funcionales entre los atributos de una tabla.
- Saber pasar una tabla en Primera Forma Normal en Tercera Forma Normal (directamente).

Los objetivos de ampliación (y que no nos interesan tanto) serán:

- Saber poner en Forma Normal Boyce-Codd

1. Introducción

El proceso de normalización se encarga de seguir una serie de pasos o normas. Después de aplicar todas, se obtienen los datos agrupados en diferentes mesas, por lo que es la estructura óptima para su implementación, gestión y explotación desde diferentes aplicaciones futuras.

La normalización se basa en que los datos deben ser independientes de las aplicaciones que las utilizan, y por tanto no serán optimizadas para una aplicación determinada sino que serán optimizadas, en general, para cualquier aplicación futura. Su objetivo es obtener el **mayor número de mesas posibles** . Cada una de estas mesas estará formada por los atributos imprescindibles para representar a la entidad o la relación entre entidades.

Ventajas que se obtienen después de la normalización:

- **Facilidad de uso.** Los datos están agrupadas en tablas que identifican claramente una entidad o una relación.
- **Flexibilidad.** La información que necesitan los usuarios se puede obtener de las tablas relacionales para medio de las operaciones del álgebra relacional. No se tienen unas estructuras rígidas de datos sino que se pueden combinar de manera que se obtengan múltiples vistas.
- **Facilidad de Implementación.** Las tablas resultantes son simples.
- **Claridad.** La representación de la información es clara y sencilla para el usuario. Son tablas sencillas.
- **Redundancia mínima.** La información no estará duplicada innecesariamente dentro de las estructuras.
- **Máximo rendimiento de las aplicaciones.** Sólo se trata aquella información que será de utilidad a cada aplicación.

Dentro de todo el proceso de creación de una Base de Datos, la normalización se haría después de construir todo el esquema relacional, es decir, primero se construiría el esquema con el Modelo E / R, luego se traduciría al Modelo Relacional , y posteriormente

se normalizarían estas tablas del esquema relacional. La verdad, sin embargo, es que normalmente las tablas que nos saldrán en el esquema relacional estarán bastante normalizadas, aunque puede ser muy conveniente analizar todas las tablas.

La verdadera utililitat y necesidad de la normalización está en la conversión o actualización de sistemas de información antiguos . Así, si queremos actualizar una aplicación basada en ficheros tradicionales, la normalización de las tablas será absolutamente necesaria.

2. Primero Forma Normal (1FN)

Una tabla está en 1FN si y sólo si los valores que componen cada atributo de una tupla son atómicos. Es decir, en un atributo no deben aparecer valores repetitivos.

Por ejemplo, en la siguiente tabla hay una serie de tipos de materiales existentes en una ferretería. Un material tiene un código que lo identifica, su descripción y sus tamaños (la clave principal es el campo subrayado)

MATERIALES

COD-MAT	DESCRIPCIÓN	MEDIDAS
039	tornillo	3,5 - 5 - 7-9
067	arandela	2-5
461	broca	2,5 - 3 - 3,5

Los problemas que plantea son los siguientes:

- La falta de espacio en el campo para los valores que puedan aparecer o, por el contrario el desperdicio del atributo cuando existen pocos valores.
- La dificultad del tratamiento para actualizaciones, consultas y búsquedas de un valor determinado.

Poner en 1FN

Para pasar a 1FN una tabla que no lo estaba **se descompone** en **dos** distintas:

A) La **primera mesa** será la proyección de la tabla original sobre los siguientes atributos:

- La clave de la tabla original.
- Los atributos atómicos (los que contienen valores únicos).

MATERIALES

COD-MAT	DESCRIPCIÓN
039	tornillo
067	arandela
461	broca

B) La **segunda mesa** será la proyección de la tabla original sobre los siguientes atributos:

- La clave de la tabla original.
- Los atributos que tienen valores múltiples, distribuyendo estos valores múltiples en tuplas distintas y por tanto en una fila existirá un único valor elemental.

  La clave de esta segunda mesa estará formada por todos los atributos.

MAT-MEDIDAS

COD-MAT	MEDIDA
039	3,5
039	5
039	7
039	9
067	2
067	5
461	2,5
461	3
461	3,5

Nota

También deberemos ser capaces de detectar que no está en 1FN cuando tenemos unos atributos multivaluados "encubiertos". Por ejemplo, una variante del ejemplo anterior podría ser:

MATERIALES

COD-MAT	DESCRIPCIÓN	MIDA1	MIDA2	MIDA3	MIDA4
039	tornillo	3,5	5	7	9

067	arandela	2	5		
461	broca	2,5	3	3,5	

Se ve que se trata de una manera de "disimular" los atributos multivaluados. Estamos ante el mismo caso que en el ejemplo de arriba y tendremos los mismos problemas, y por lo tanto la solución es la misma.

3.1 DEPENDENCIA FUNCIONAL

> Un atributo o conjunto de atributos **Y** depende funcionalmente del atributo o conjunto de atributos **X** , y se representa como **X** → **Y** , si y sólo si cada valor de X se corresponde con un único valor de Y.

Por ejemplo, entre DNI y NOMBRE existe una dependencia funcional, ya que el valor DNI se corresponde con un único nombre.

DNI → NOMBRE

En este caso (suponiendo que el NOMBRE es único y no existen dos nombres iguales) se cumple también la dependencia

NOMBRE → DNI

y lo podríamos abreviar por

DNI ← → NOMBRE

No siempre se da de forma biunívoca la dependencia funcional entre dos atributos, es más, en pocos casos sucede. Por ejemplo, entre los atributos DNI y DIRECCIÓN existe una dependencia funcional ya que una persona identificada por su DNI vive en una única DIRECCIÓN.

DNI → direcciones

Pero en este caso no se da la dependencia en sentido inverso, porque en una DIRECCIÓN viven varias personas. Además, la dirección no nos dice ni siquiera la ciudad donde se encuentra (por ejemplo, el "C / Mayor, 7" se encuentra en muchas ciudades)

Hay atributos que no tienen entre ellos una dependencia funcional como es el caso de DIRECCIÓN y fecha_nacimiento.

En muchas ocasiones, para determinar un único valor de un atributo, no es suficiente con conocer el valor de otro atributo, sino que es necesario encontrar los valores de varios atributos. Esto es lo que pasa, si tenemos los atributos: DNI, EMPRESA y SOU, y sabemos que una persona puede trabajar en más de una empresa. Entre los atributos DNI y SOU no existe ninguna dependencia funcional, ya que un individuo puede ganar sueldos distintos en empresas distintas. Pero si conocemos la empresa en la que trabaja, sí podemos decir que:

DNI. EMPRESA → SUELDO

El operador punto "." representa la expresión "junto con" o "y" entre los dos atributos y el operador barra "|" hace referencia a la expresión "o también".

Por tanto, podemos decir que DNI "junto con" EMPRESA determinan el SUELDO. Y para la dependencia:

DNI → NOMBRE | direcciones

diremos que con el DNI se conoce el NOMBRE "o también" la DIRECCIÓN.

3.2 DEPENDENCIA FUNCIONAL TOTAL

Se dice que el atributo **Y** tiene una **dependencia funcional total** del atributo **X** si tiene una dependencia funcional de X y NO depende funcionalmente de ningún subconjunto de X.

Por ejemplo, una dependencia funcional sería:

DNI. EMPRESA → NOMBRE

Pero, lógicamente, esta dependencia no es total ya que NOMBRE depende funcionalmente de DNI. Por eso, en esta dependencia se denomina **dependencia parcial** .

La dependencia funcional total sería:

DNI. EMPRESA → SUELDO

Ahora sí, el SOU no depende funcionalmente de ningún subconjunto.

Es evidente que si X está formado únicamente por un atributo, la dependencia funcional será total.

Las dependencias que nos interesan para la normalización son siempre las dependencias funcionales totales.

3.3 GRAF DE LAS DEPENDENCIAS FUNCIONALES

Sirve para mostrar gráficamente la relación existente entre todos los atributos y es una forma clara de tener una visión general de los datos y de la cohesión existente entre ellas. La clave principal se representa dentro de una caja de líneas continuas con sus atributos primarios. El resto de los atributos se representa fuera de la caja.

Si existen dependencias de un conjunto de atributos que no son la clave, irán dentro de una caja de líneas discontinuas para no confundirlo con la clave principal de la tabla.

ejemplo:

A. B. C → M | S

M → N

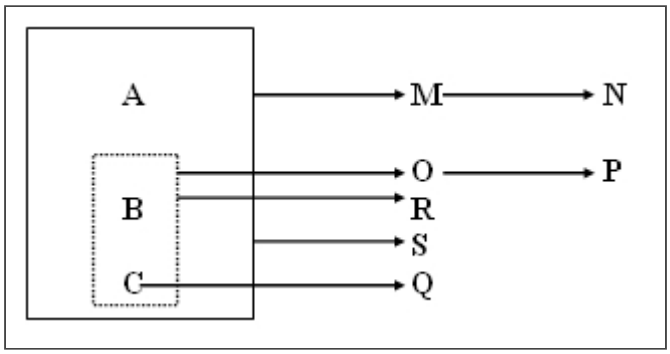
B. C → O | R

O → P

C → Q

(clave = A. B. C)

El grafo asociado es:



4. Segunda Forma Normal (2FN)

Una mesa se dice que está en 2FN si y sólo si cumple dos condiciones:

- **Se encuentra en 1FN.**
- **Todo atributo secundario (aquellos que no pertenecen a la clave principal, los que se encuentran fuera de la caja) depende totalmente (tiene una dependencia funcional total) de la clave completa y, por tanto, no de una parte de ella.**

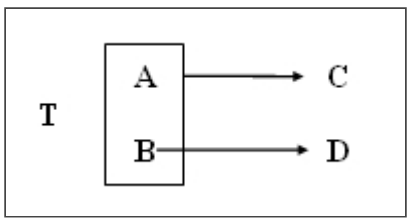
Esta forma normal sólo se considera si la clave principal es compuesta, y por tanto, está formada por varios atributos.

Si una tabla **T** tiene como atributos **A, B, C, D** y la clave es **A. B** cumpliéndose las dependencias:

A. B → C

B → D

Se observa que la mesa no se encuentra en 2FN ya que el atributo D no tiene una dependencia funcional total con la clave completa A. B, sino con una parte de la llave (B). El grafo de las dependencias funcionales sería:

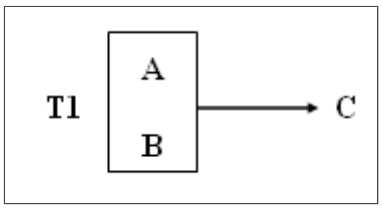


Si existe una flecha que sale del interior de la caja que engloba la clave, entonces la mesa no está en 2FN.

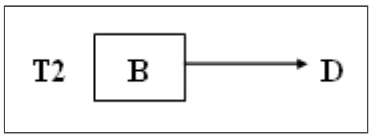
Poner en 2FN

Para convertir una tabla que no está en segunda forma normal en 2FN, se realiza una proyección y se crea:

A) Una **primera mesa** con la clave y todas sus dependencias totales con los atributos secundarios afectados:



B) Una **segunda mesa** con la parte de la clave que tiene dependencias, y los atributos secundarios implicados:



La clave de la nueva tabla T2 será la antigua parte de la clave.

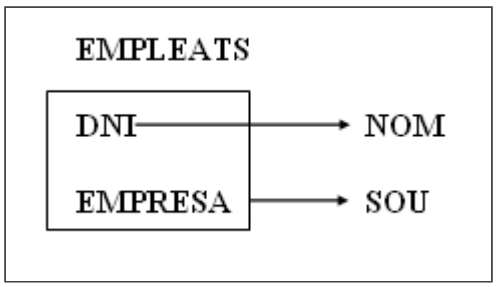
Ejemplo : Tabla con las personas que trabajan en varias empresas con el sueldo correspondiente, con los atributos: **DNI, NOMBRE, EMPRESA, SOU**

Entre los atributos existen las dependencias:

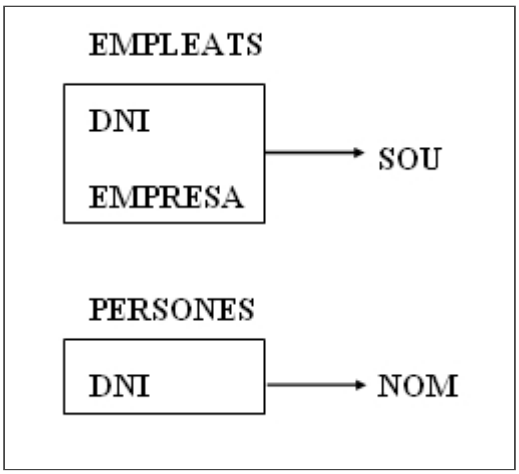
DNI → NOMBRE

DNI. EMPRESA → SUELDO

El grafo que muestra las dependencias es el siguiente:



Es evidente que la mesa no se encuentra en 2FN, tras normalizar obtiene:


EMPLEATS
DNI
EMPRESA
SOU
PERSONES
DNI
NOM

5. Dependencia Funcional Transitiva

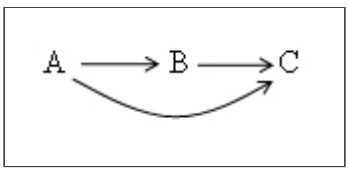
La dependencia funcional transitiva aplica para analizar las tablas en tercera forma normal (3FN). Consiste básicamente en considerar que **un atributo no primario sólo debe conocerse a través de la clave principal o claves secundarias** . En otro caso, estará produciendo redundancia de información con las anomalías típicas que lleva con ella.

Supongamos tres subconjuntos distintos de atributos A, B y C que pertenecen a una tabla T, por lo que se cumplen las condiciones:

A → **B** y **B** - / → **A**

Se dice que C tiene una **dependencia funcional transitiva** con A o que es transitivamente dependiente de A si se cumple que **B** → **C**

Gráficamente se puede mostrar:



Por tanto, un atributo C es transitivamente dependiente de otro A si se conoce por diferentes vías, una directamente, y otra a partir de otro atributo intermedio B.

Por ejemplo, consideremos tres atributos que forman parte de la tabla ALUMNOS:

Nummi = núm. de matrícula.

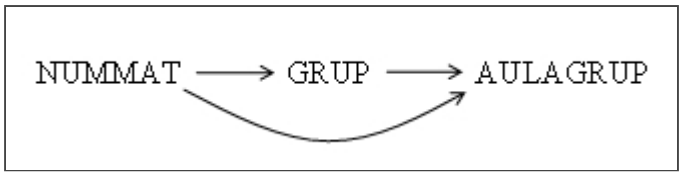
GRUPO = Grupo asignado.

AULAGRUP = Aula asignada al grupo.

Nummi → GRUPO | AULAGRUP

GRUPO → AULAGRUP

El atributo AULAGRUP es transitivamente dependiente de Nummi, ya que se puede conocer por medio del atributo nummi ya través del atributo GRUPO



6. Tercera Forma Normal (3FN)

Una mesa se dice que está en 3FN si y sólo si se cumplen dos condiciones:

- **Se encuentra en 2FN.**
- **No existen atributos no primarios (atributos que no forman parte de la clave principal) que son transitivamente dependientes de cada clave candidata de la tabla.**

Esto quiere decir que un atributo secundario sólo se puede conocer a través de la clave principal o claves candidatas de la mesa y no por medio de otro atributo no primario.

En el grafo de dependencias sólo deben mostrarse las dependencias transitivas y no aquellas dependencias funcionales a partir de las claves candidatas, porque se sabe que para ser claves ya conocen todos los atributos.

Ejemplo: A es la clave principal, B es una clave candidata y se dan las siguientes dependencias:

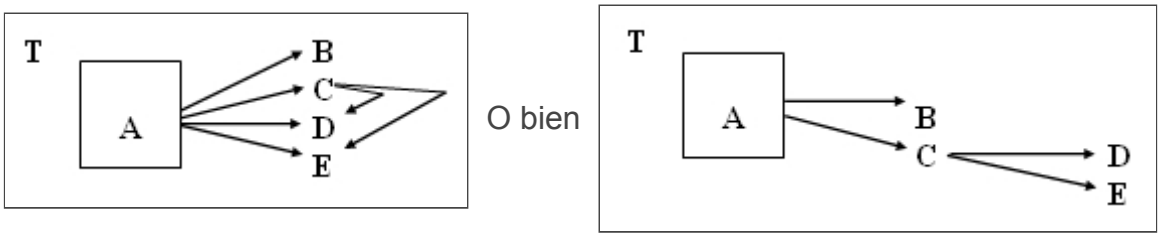
A → BB → AC → D

A → CB → CC → E

A → DB → D

A → EB → E

El grafo queda del siguiente modo:



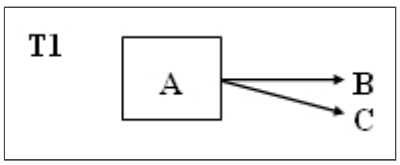
Las flechas que muestran las dependencias funcionales que tiene la clave candidata B no se representan (como hemos dicho anteriormente) porque son evidentes y no simplifican la visión del grafo. Además, para la normalización, no se necesitan para nada; por el contrario, suelen complicar el análisis.

La tabla T no está en 3FN ya que los atributos D y E son transitivamente dependientes respecto de la llave A.

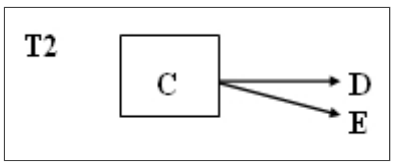
Poner en 3FN

Para normalizar una tabla que no esté en tercera forma normal, es decir, que tenga dependencias transitivas, descomponer la mesa en más de una tabla:

A) Una **primera mesa** con la clave principal más los atributos que no dependen transitivamente

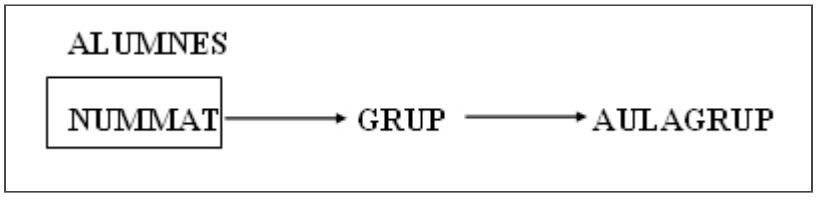


B) Una **segunda mesa** con los atributos que dependen transitivamente, más el atributo de quien dependen, que será clave principal

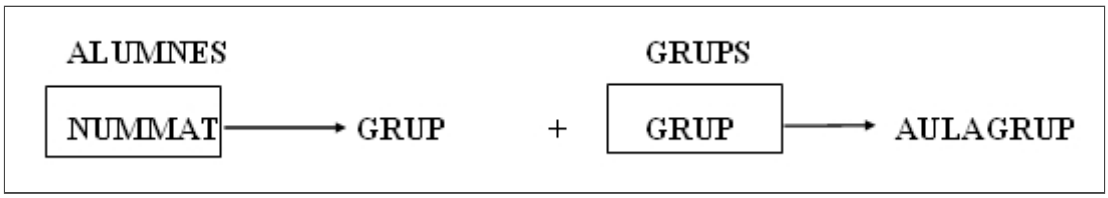


Se hará una descomposición por cada dependencia transitiva que haya que afecte a campos distintos.

Para el ejemplo de los atributos nummi, GRUPO y AULAGRUP tenemos el siguiente grafo:



Una vez descompuesta la mesa en dos segundos el algoritmo anterior, tendremos dos tablas

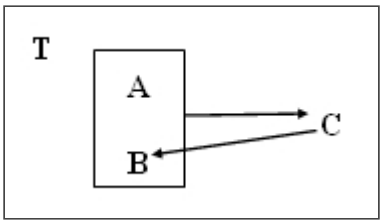


Las dos tablas resultantes sí se encuentran en 3FN.

7. Forma Normal Boyce-Codd (FNBC)

Tras la creación de la 3FN se observó posteriormente que podían haber anomalías que no eran abordadas. De todas formas son unas redundancias ya muy extrañas, y que a veces no valdrá la pena considerarlas.

Son casos de tablas que aunque están en 3FN, mantienen una dependencia de un atributo secundario con parte de la clave. Es el único caso de dependencia transitiva que se nos podía haber escapado. Gráficamente es el siguiente caso:



Una mesa T está en FNBC si y sólo si está en 1FN y las únicas dependencias funcionales elementales son aquellas en las que la clave principal (y claves candidatas) determinan un atributo.

La definición engloba la 3FN ya que las dependencias transitivas existen por medio de atributos secundarios que no eran clave.

Si la clave está formada por un único atributo y ya estaba en 3FN, la mesa está en FNBC (como sucedía con la 2FN).

Ejemplo : Tabla de un callejero

CALLEJERO

DIRECCIÓN	CIUDAD	CODPOST
C / Pez, 2	Benicarló	12580
C / Luz, 5	Benicarló	12580
C / Mar, 4	Castellón	12005
C / Sol, 4	Vinaròs	12500
C / Sal, 9	Castellón	12004

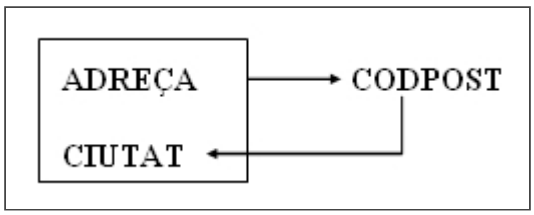
C / Mar, 4	Vinaròs	12500

Las dependencias funcionales que nos encontramos son:

DIRECCIÓN. CIUDAD → CODPOST

CODPOST → CIUDAD

gráficamente:

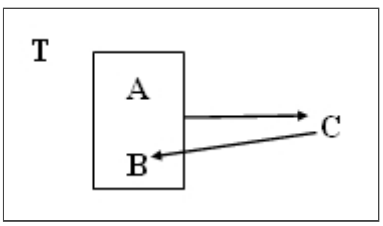


Si observamos atentamente las tuplas de una tabla como esta, veremos que para un mismo código postal existen multitud de tuplas que se corresponden con la misma ciudad (tantas como direcciones existan diferentes), por tanto existe información duplicada.

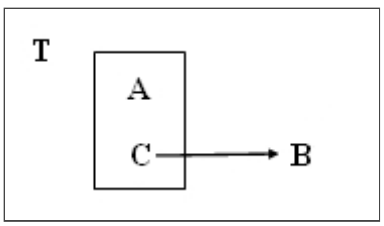
Si la información, una vez que se da de alta no varía, es más rentable que la dependencia funcional **CODPOST → CIUDAD** se encuentre en otra mesa y que exista una sola tupla para cada código postal.

Además, ¿qué sucede si se elimina la tupla con dirección "C / Sol, 4" de "Vinaròs" y la tupla "C / Mar, 4" de "Vinaròs"? Lo que ocurre es que desaparece la relación entre el código postal "12500" y "Vinaròs" y quizás estos datos deberían mantenerse.

Si analizamos con más detalle la mesa, veríamos que en realidad se puede sustituir la clave principal para **A + C** (en el ejemplo **DIRECCIÓN + CODPOSTAL**), ya que si A + B ya era clave principal, como para cada valor de C sólo podemos tener uno de B, la combinación A + C también podrá identificar unívocamente cada ocurrencia de la tabla. Por tanto, si sustituiremos la clave principal, ya no tendríamos dudas de cómo normalizar la mesa, que será justamente como veremos a continuación:



es equivalente a



Poner en FNBC

El algoritmo de descomposición que se aplica a una tabla que no está en FNBC es el siguiente:

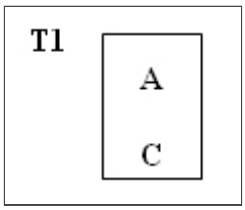
Si tenemos una dependencia funcional **C** → **B** donde **C** y **B** son disjuntos, **C** es una atributo no primario, y **B** forma parte de la clave.

Se obtienen las proyecciones:

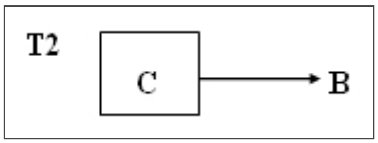
> **A)** Una **primera mesa T1** con todos los atributos, excepto **B** (lo que formaba parte de la clave principal); ahora formará parte de la clave principal **C** .
>
> **B)** Una **segunda mesa T2** con los atributos **C** y **B** , y será la clave principal **C**

En el ejemplo inicial de esta pregunta quedará:

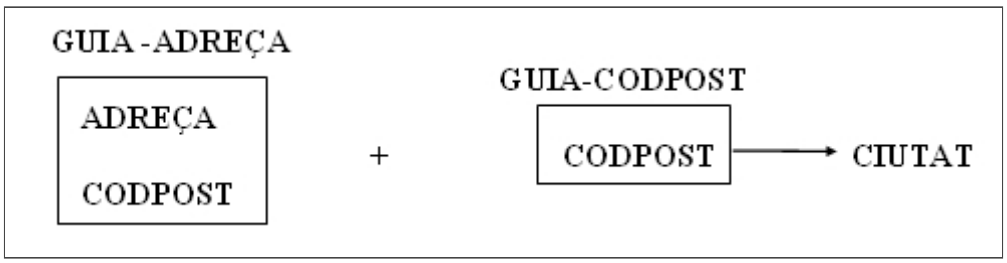
A)



B)



En el ejemplo de la CALLEJERO:



Y quedarían con la siguiente información:

GUÍA-DIRECCIÓN

DIRECCIÓN	CODPOST
C / Pez, 2	12580

GUÍA-CODPOST

CODPOST	CIUDAD
12004	Castellón

C / Luz, 5	12580
C / Mar, 4	12005
C / Sol, 4	12500
C / Sal, 9	12004
C / Mar, 4	12500

12005	Castellón
12500	Vinaròs
12580	Benicarló

Por último, observamos las tablas que nos quedan. Querremos tener una tabla de códigos postales? Si el diseño es para Correos o Telefónica, o una empresa grande que tenga muchos cliente y los quiere tener distribuidos por códigos postales, pues seguro que sí.

Pero si se trata de una empresa no demasiado grande, y que tampoco interesa demasiado la distribución por códigos postales, seguramente mantener una tabla de códigos postales puede parecer incluso ridículo. Entonces, mantener la mesa en 3FN y asumir la poquito de redundancia que supone no tenerla en FNBC, puede ser incluso saludable. Por ello se ha comentado desde el principio del tema la importancia de normalizar hasta la 3FN, y la FNBC tiene una importancia relativa.

Resumen

1FN

Una tabla está en 1FN si y sólo si los valores que componen los atributos de una tupla son atómicos.

Se descompone la mesa en dos

1ª.- Proyección de la clave junto con los atributos que tienen valores atómicos.

2ª.- Nueva clave con los atributos que tienen valores múltiples (hay que idear una nueva clave).

2FN

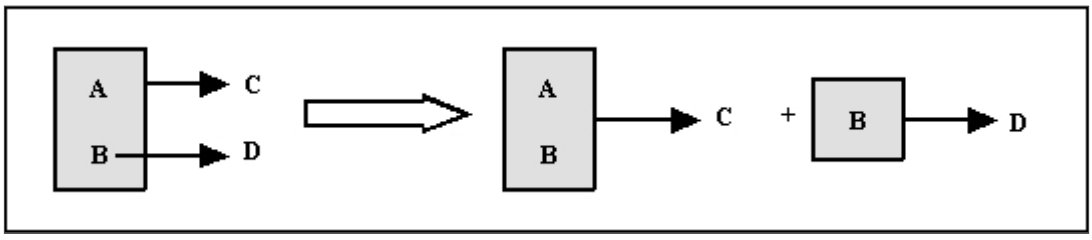
Una tabla está en 2FN si y sólo si cumple dos condiciones:

- **Está en 1FN.**
- **Todo atributo secundario (los que no pertenecen a la clave principal) depende totalmente (tiene una dependencia funcional total) de la clave completa, y por tanto, no de una parte de ella.**

Se descompone la mesa en dos

1ª.- Una tabla con la clave y todas sus dependencias totales.

2ª.- Otra mesa con la parte de la clave que tiene dependencias, y los atributos secundarios implicados.



3FN

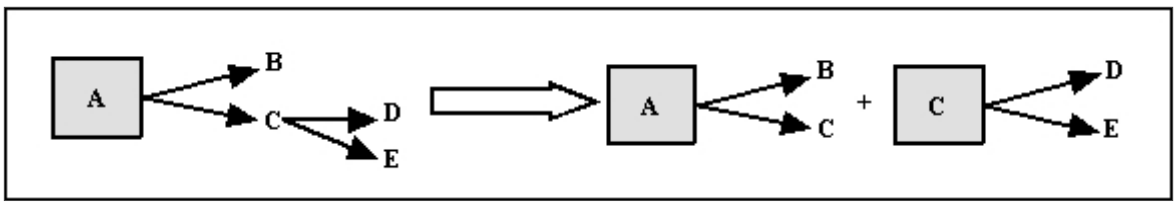
Una tabla se encuentra en 3FN si y sólo si se cumplen dos condiciones:

- **Está en 2FN.**
- **No existen atributos no primarios (que no forman parte de la clave) que son transitivamente dependientes de una clave candidata (cada posible clave de la tabla).**

Se descompone la mesa en dos

1ª.- Una tabla con la clave y todos los atributos no primarios que no son transitivos.

2ª.- Otra tabla con los atributos transitivos y el atributo no primario (que será la clave de la nueva tabla).



FNBC

Una tabla está en FNBC si y sólo si está en 1FN y las únicas dependencias funcionales son aquellas en que la clave principal (y claves candidatas) determinan un atributo.

Se descompone la mesa en dos

1ª.- Una mesa con todos los atributos menos la parte de la clave dependiente del atributo secundario. La clave está formada por el resto de la llave y el atributo secundario del que dependía parte de la clave.

2ª.- Otra mesa en la que el atributo de la que depende parte de la clave será la nueva clave y esta parte de la clave como atributo secundario.