Normalización.

Dentro del contexto del modelo relacional el objetivo de la normalización es conseguir un conjunto de relaciones óptimo que, una vez implementado físicamente, forme una estructura flexible, fácil de mantener, que elimine ciertos tipos de redundancia y evite anomalías en operaciones de actualización de datos (inserción, borrado y modificación).

Aunque un diseñador experimentado puede llegar a obtener relaciones normalizadas directamente, nosotros seguiremos un método formal.

Originalmente Codd definió tres niveles de normalización llamados primera forma normal (1FN), segunda forma normal (2FN) y tercera forma normal (3FN), existiendo una revisión de esta última propuesta por Boyce y el propio Codd que se llamó forma normal de Boyce-Codd (BCNF), posteriormente algunos autores definieron más niveles (4FN, 5FN...), en la práctica, normalizar hasta estos últimos niveles no suele tener sentido, ya que se generan gran cantidad de relaciones produciendo una implementación física difícil de mantener, teniéndose que realizar la operación contraria, es decir, la desnormalización.

Definiciones e ideas básicas.

Antes de seguir es importante definir algunos conceptos que posteriormente se utilizarán.

- **Dependencia funcional.** Dados dos atributos A y B de una relación R, se dice que B es funcionalmente dependiente de A, si cada valor de A tiene asociado un único valor de B. En otras palabras: si en cualquier instante, conocido el valor de A podemos conocer el valor de B. Tanto A como B pueden ser conjuntos de atributos. La dependencia funcional se simboliza del siguiente modo:

  $$R.A \longrightarrow R.B$$

  Por ejemplo, en la relación de la página siguiente: *Fecha* y *CodPro* dependen funcionalmente de *NumPed*.

  $$\text{NumPEd} \longrightarrow (\text{Fecha, CodPro})$$

- ***Dependencia funcional completa*** *se dice que un atributo B tiene dependencia funcional completa de un grupo de atributos A (al, a2,...) de la misma relación, si B depende funcionalmente de todo A, pero no de un subconjunto de A*.

  $$(R.a1, R.a2) \longrightarrow R.B$$

  Por ejemplo, en la relación PRECIOS del tema anterior, el precio de un artículo dependía funcionalmente completa del conjunto de atributos código de artículo, código del proveedor:

  $$(CodPro, CodArt) \longrightarrow Precio$$

- ***Dependencia transitiva:*** *sean A, B y C tres atributos o conjuntos de atributos de una relación. Si B depende funcionalmente de A y C de B, pero A no depende funcionalmente de B, se dice que C depende transitivamente de A*.

  Por ejemplo: en la relación de la página siguiente los atributos nombre y dirección del proveedor dependen transitivamente de la clave NumPed, es decir, no dependen funcionalmente de ella sino del atributo código del proveedor. Por otro lado NumPed no depende de CodPro.

  $$\underset{A}{NumPed} \longrightarrow \underset{B}{CodPro} \longrightarrow \underset{C}{(NomPro, DirPro)}$$

Primera forma normal.

Una relación está en primera forma normal si el valor de cada atributo de cada fila no se puede descomponer, es decir contiene valores atómicos. Dicho de otra forma, si en la relación cada intersección de fila columna contiene un solo valor y no un conjunto de ellos.

Para que una relación esté en 1FN no podrá contener atributos con varios valores para un valor concreto de la clave (grupos repetitivos). Veamos el siguiente ejemplo: podemos observar dos registros con claves NumPed 0001 y 0002, en las que los atributos correspondientes a cada línea de un pedido toman valores distintos para un valor concreto de la clave. En este caso se dice que este conjunto de valores está valuado en relaciones.

NumPed	Fecha	CodPro	NomPro	DirPro	CodArt	Descrip	Cantidad	Precio
0001	111098	314	InfoSet	Torla, 5	11	Teclado	3	1.600
					22	Impresora	5	28.000
					33	Monitor	2	19.900
0002	121098	112	OfiMicro	Aries, 8	11	Teclado	8	1.600
					66	Lector CD	5	15.000
					55	Altavoces	4	4.500

Relación pedido no normalizada: existen grupos repetitivos formados por los artículos de un pedido.

Anomalías: Aumento de la dificultad de tratamiento por los programas de aplicación: por ejemplo si quisiéramos modificar o añadir un artículo a un pedido.

Para implementar estos casos con ficheros clásicos no planos, la estructura de almacenamiento solía constar de una estructura repetitiva de longitud prefijada (tabla, array, occurs) que almacenaba los distintos artículos de un pedido. Esto conllevaba el inconveniente que si un pedido sólo constaba de uno o dos artículos el resto de las posiciones quedaban libres. Por otro lado, podía darse el caso de un pedido que sobrepasará el total de artículos previstos.

Solución: Una solución para normalizar esta relación consistiría en "*aplanar"* la estructura, repitiendo los datos del pedido por cada artículo del pedido y tomando como clave primaria la concatenación de los atributos NumPed y CodArt (suponiendo que nunca se van a producir dos peticiones del mismo artículo en un mismo pedido en distintas líneas). Esta solución da lugar a un mayor número de tuplas.

NumPed	Fecha	CodPro	NomPro	DirPro	CodArt	Descrip	Cantidad	Precio
0001	111098	314	InfoSet	Torla, 5	11	Teclado	3	1.600
0001	111098	314	InfoSet	Torla, 5	22	Impresora	5	28.000
0001	111098	314	InfoSet	Torla, 5	33	Monitor	2	19.900
0002	121098	112	OfiMicro	Aries, 8	11	Teclado	8	1.600
0002	12109	112	OfiMicr	Aries,	66	Lector	5	15.00

	8		o	8		CD		0
0002	121098	112	OfiMicro	Aries, 8	55	Altavoces	4	4.500

LINEASPEDIDO(NumPed, Fecha, CodPro, NomPro, DirPro, CodArt, Descrip, Cantidad, Precio)

Esta solución aún deja mucho que desear, entre otros motivos, porque posee mucha información redundante. (*Imagina que ocurriría si cambiara el nombre o la dirección de un proveedor*).

Algunos autores (ni Codd, ni Date) a efectos prácticos, proponen como solución tomar los atributos que forman parte del grupo repetitivo y construir con ellos una nueva relación eliminándolos de la original. La clave principal de la nueva relación estará formada por la concatenación de uno o varios de sus atributos con la clave principal de la antigua. Pero como veremos, de todo esto se ocupa la segunda forma normal.

Si aplicamos esta segunda solución a nuestro ejemplo, consistiría en desdoblar la relación en dos, del siguiente modo:

PEDIDOS(NumPed, Fecha, CodPro, NomPro, DirPro)
LINEASPEDIDO(NumPed, CodArt, Descrip, Cantidad, Precio)

Evidentemente también se evitan grupos repetitivos. De todas formas, proseguiremos a partir de la primera solución para observar cómo llegamos a un resultado similar, que si aplicáramos la segunda forma normal.

Segunda forma normal.

Una relación estará en 2FN si ya está en 1FN y cada uno de sus atributos depende totalmente de la clave y no de parte de ella. De una forma más rigurosa podemos decir que una relación está en 2FN si está en 1FN y cada atributo que no pertenezca a una clave candidata concreta tiene una ***dependencia funcional completa*** de dicha clave candidata.

Partiendo de una relación en 1FN pasamos a 2FN identificando los atributos que no dependen completamente de alguna clave candidata y formando con ellos una nueva relación quitándolos de la antigua, la clave principal de la nueva relación estará formada por la parte de la antigua de la que dependen totalmente.

Partiendo de la siguiente relación en 1FN con clave principal NumPed+CodArt:

LINEASPEDIDO(NumPed, Fecha, CodPro, NomPro, DirPro, CodArt, Descrip, Cantidad, Precio)

Los atributos fecha del pedido, código, nombre y dirección del proveedor no tienen una dependencia funcional completa de la clave NumPed+CodArt, sino sólo de parte de ella (NumPed). Por lo que podríamos generar una nueva relación con estos atributos:

CABECERAPEDIDO(NumPed, Fecha, CodPro, NomPro, DirPro)
LINEASPEDIDO(NumPed, CodArt, Descrip, Cantidad, Precio)

Esta solución sigue teniendo anomalías, por ejemplo: Si observamos la relación LINEASPEDIDO:

- En altas: no se puede introducir la descripción y el precio de un material mientras no tengamos una línea de pedido que lo incluya.
- En bajas: si se borra una línea de pedido podemos perder la información de un material si sólo existía en esta línea.
- En modificaciones: para cambiar descripción y precio de un material tenemos que realizar un cambio en todas las líneas de pedidos en las que aparezca.

Estas anomalías se deben a que la relación LINEASPEDIDO, aún no están en segunda forma normal. Ya que los atributos descripción del material y precio unitario no tienen una dependencia funcional completa de la clave NumPed+CodArt, sino sólo de parte de ella (CodArt).

La solución sería, como antes, generar una nueva relación con estos atributos:

CABECERAPEDIDO(NumPed, Fecha, CodPro, NomPro, DirPro)

LINEASPEDIDO(NumPed, CodArt, Cantidad)
ARTICULOS(CodArt, Descrip, Precio)

Tercera forma normal.

Una relación estará en 3FN si ya está en 2FN y cada uno de sus atributos depende solamente de la clave y no de otros atributos. De una forma más rigurosa podemos decir que una relación estará en 3FN si está en 2FN y cada atributo que no pertenezca a una clave

candidata concreta no ***depende transitivamente*** de dicha clave candidata.

Partiendo de una relación en 2FN pasamos a 3FN identificando los atributos que dependen de otro atributo distinto de una clave candidata y formando con ellos otra relación, quitándolos de la antigua. La clave principal de la nueva relación será el atributo del cual dependen. Este atributo en la relación antigua pasará a ser una ***clave extranjera*** o ***ajena***.

Analicemos la relación:

CABECERAPEDIDO(NumPed, Fecha, CodPro, NomPro, DirPro)

Los atributos nombre y dirección del proveedor dependen transitivamente de la clave, es decir, no dependen funcionalmente de ella sino del atributo código del proveedor.

Esto ocasiona ciertas anomalías en la relación CABECERAPEDIDO:

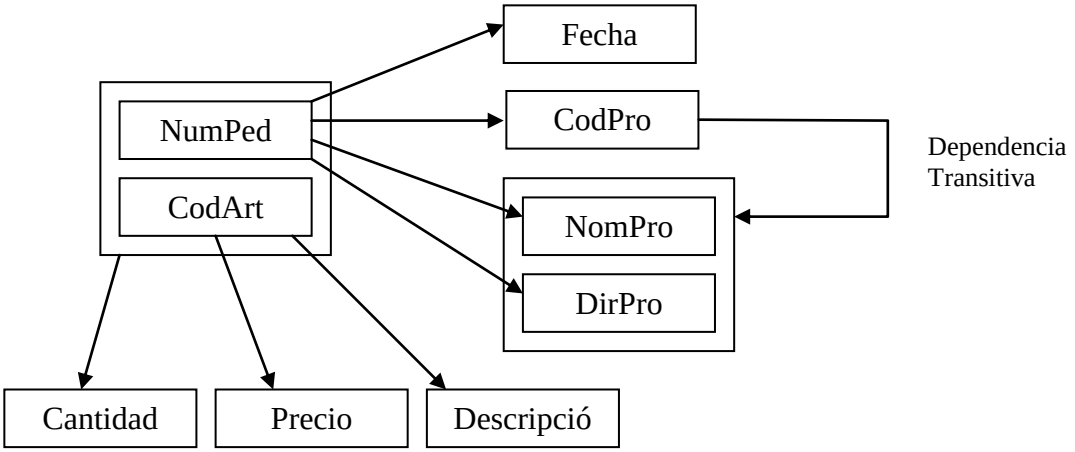
- En altas: no se puede introducir los datos de un proveedor mientras no tengamos un pedido para él (una cabecera de pedido que lo incluya).
- En bajas: si se borra una cabecera de pedido podemos perder la información de un proveedor si sólo existía en este pedido.
- En modificaciones: para cambiar los datos de un proveedor tengo que realizar un cambio en todas las cabeceras de pedidos en las que aparezca.

La solución consiste en tomar los atributos que dependen del código del proveedor y formar con ellos otra relación, quitándolos de la antigua. La clave principal de la nueva relación será el código del proveedor. El código de proveedor permanece es la relación CABECERAPEDIDO como clave extranjera o ajena.

CABECERAPEDIDO(NumPed, Fecha, CodPro)
PROVEEDOR(CodPro, NomPro, DirPro)

LINEASPEDIDO(NumPed, CodArt, Cantidad)
ARTICULOS(CodArt, Descrip, Precio)

Los diagramas de dependencias funcionales son una herramienta gráfica que aportan bastante claridad a la hora de normalizar. Veamos el diagrama de dependencias funcionales de este ejemplo:



Forma normal Boyce/Codd.

La definición original de Codd de 3FN tenía ciertas deficiencias. Concretamente no manejaba de manera satisfactoria el caso de una relación en la cual: Hay ***varias claves candidatas compuestas y solapadas***, es decir, tienen por lo menos un atributo común. Esta situación puede presentarse muy raras veces en la práctica.

Para definir BCNF vamos a introducir un término nuevo. Definimos ***determinante*** a un atributo del cual depende funcionalmente completa otro atributo.

Por ejemplo en la relación:

LINEASPEDIDO(NumPed, Fecha, CodPro, NomPro, DirPro, CodArt, Descrip, Cantidad, Precio)

Serían determinantes los siguientes atributos:

- NumPed, ya que de él dependen funcionalmente Fecha y CodPro.
- CodPro, ya que de él dependen funcionalmente NomPro y DirPro.
- CodArt, ya que de él dependen funcionalmente Descrip y Precio.
- NumPed+CodArt, ya que de ellos depende de forma funcionalmente completa Cantidad.

Definimos BCNF del siguiente modo: *Una relación está en forma normal Boyce/Codd (BCNF) si y sólo si todo determinante es una clave candidata.*

Es interesante notar la simplicidad de esta definición, que no se apoya en la 1FN ni en la 2FN, ni en la idea de dependencia transitiva.

Una relación que no está en BCNF debe descomponerse en otras dos de la siguiente forma: Sea una relación R(A,B,C,D,E,...) que no está en BCNF, siendo C un determinante, de forma que C D, pero no una clave candidata. Se forman dos nuevas relaciones, las proyecciones: R1(A,B,C,E,...) y R2(C,D). De nuevo se comprueba si R1 y R2 están en BCNF; en caso de no ser así se vuelve a repetir el mismo proceso.

Como ejercicio se podría aplicar esta regla de descomposición a la relación LINEASPEDIDO, observando que se llega a una solución similar que aplicando la 2FN y la 3FN.

Otra idea interesante es que, una relación en BCNF está siempre en 3FN, sin embargo, una relación en 3FN, no tiene porqué estar en BCNF. Veamos un ejemplo, supongamos la relación siguiente:

ALUMNO	ASIGNATURA	PROFESOR
Carlos Pérez	Unix	Pedro
Elvira Gómez	Unix	Victoria
Carlos Pérez	Lenguaje C	Luis
Elvira Gómez	Lenguaje C	Lola
Ana López	Lenguaje C	Lola

El significado de una tupla es que "*un profesor imparte una asignatura a un alumno determinado*". Suponiendo además, las siguientes restricciones:

- Para cada asignatura, cada estudiante de esa asignatura tiene un solo profesor, es decir, tanto Pedro como Victoria dan la asignatura de Unix, pero a distintos alumnos.
- Cada profesor imparte una sola asignatura, pero cada asignatura es impartida por varios profesores. Cada profesor da clase a más de un alumno.

Si analizamos las dependencias funcionales llegamos a las siguientes conclusiones:

Por la primera restricción tenemos que:

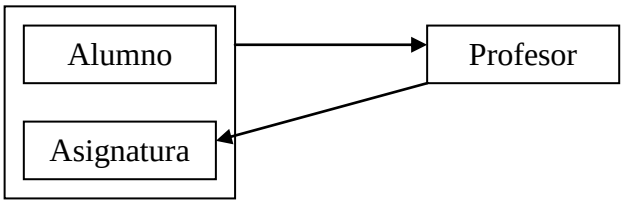
(alumno, asignatura) ----→ profesor

Por la segunda restricción tenemos que:

Profesor ---→ asignatura

Según esto podríamos tomar como claves candidatas a (alumno, asignatura) o bien (alumno, profesor), se trata de dos claves candidatas compuestas y solapadas.

Veamos el diagrama de dependencias funcionales:



Esta relación, tomando como clave primaria (alumno, asignatura), está en 2FN ya que el atributo no clave (profesor) es totalmente dependiente de la clave primaria. Además está en 3FN ya que el atributo no clave (profesor) no depende transitivamente de ninguna clave candidata. Pero no está en BCNF ya que existe un determinante (profesor) que no es clave candidata, lo que origina redundancias. Además plantea ciertos problemas de actualización. Por ejemplo, si deseamos eliminar la información "Carlos Pérez estudia Unix", perderíamos al mismo tiempo la información de "Pedro enseña Unix".

La solución sería, como propusimos anteriormente, desdoblar del siguiente modo:

R1(<u>alumno, profesor</u>)
R2(<u>profesor</u>, asignatura)

ALUMNO	PROFESOR
Carlos Pérez	Pedro
Elvira Gómez	Victoria
Carlos Pérez	Luis
Elvira Gómez	Lola
Ana López	Lola

PROFESOR	ASIGNATURA
Pedro	Unix
Victoria	Unix
Luis	Lenguaje C
Lola	Lenguaje C

La dependencia funcional (alumno,asignatura) -----→ profesor, se ha perdido, pero se puede generar por la reunión natural de ambas relaciones por el atributo profesor.

Esta división aunque resuelve algunos problemas, trae otros distintos. Concretamente las dos relaciones no se pueden actualizar de manera independiente. Por ejemplo se deberá rechazar cualquier intento de insertar una tupla en R1, para Elvira Gómez y Pedro,

porque Pedro enseña Unix y Elvira Gómez ya tiene profesor de Unix. La única forma de que el sistema se dé cuenta de esto, es consultando la relación R2.

Cuarta forma normal.

A veces BCNF es insuficiente para eliminar las redundancias y anomalías de actualización si una relación contiene ***dependencias de valores múltiples***, lo cual conlleva una normalización adicional: la cuarta forma normal.

Veamos un ejemplo: Supongamos que tenemos una "*relación no normalizada*" con información acerca de las asignaturas, profesores y textos recomendados. Cada registro de la relación se compone de un nombre de asignatura, más un grupo repetitivo de nombres de profesores, más un grupo repetitivo de nombres de textos. Cada registro se interpretaría como que una determinada asignatura puede ser impartida por cualquiera de los profesores especificados y emplea todos los libros de textos especificados. Para una asignatura puede existir cualquier número de profesores y de textos recomendados, además suponemos que los profesores y los textos son totalmente independientes entre sí. Y por último, también suponemos que un profesor o un texto pueden estar asociados a cualquier número de asignaturas.

Veamos una representación de dicha relación:

ASIGNATURA	PROFESOR	TEXTO
Programación en C	Javier Krahe Alberto Pérez	Lenguaje C Metodología de la programación.
Telemática	Javier Krahe	Lenguaje C Comunicaciones con Linux Teleinformática

Si intentamos convertir esta estructura en una forma normalizada equivalente, lo único que podremos hacer hasta el momento es "aplanar" la estructura, ya que no existe ninguna dependencia funcional.

ASIGNATURA	PROFESOR	TEXTO
Programación en C	Javier Krahe	Lenguaje C
Programación en C	Javier Krahe	Metodología de la programación.
Programación en C	Alberto Pérez	Lenguaje C
Programación en C	Alberto Pérez	Metodología de la programación.
Telemática	Javier Krahe	Lenguaje C
Telemática	Javier Krahe	Comunicaciones con Linux
Telemática	Javier Krahe	Teleinformática

Es evidente que esta relación implica bastante redundancia, lo cual conduce a ciertas anomalías de actualización. Por ejemplo, si queremos añadir la información de que la asignatura de Telemática puede ser impartida por un nuevo profesor, es necesario añadir tres nuevas tuplas, una para cada texto. Sin embargo esta relación está en BCNF, porque la clave está formada por todos los atributos.

Las ***dependencias multivaluadas*** (DMV) se definen como siguen: *Dada una relación R con los atributos A,B, y C, la dependencia multivaluada R.A --------→ R.B, se cumple en R si y sólo si existe un conjunto de valores de B que depende de un valor de A y es independiente del valor de C. (A,B y C pueden ser compuestos)*

Observa que las DMV, sólo pueden existir si la relación tiene al menos tres atributos. Además en una relación R(A, B, C), la DMV R.A -----→ R.B se cumple, si y sólo si también se cumple R.A --------→ R.C. Las DMV siempre se presentan en pares como éste, y por ello es común representarlas con la siguiente notación:

$$R.A \longrightarrow\!\!\!\!\!\rightarrow R.B \mid R.C$$

Ahora podemos definir la 4FN: *Una relación R está en 4FN si y sólo si, está en BCFN y no contiene dependencia multivaluadas.*

Fagin demuestra que una relación R(A,B,C) donde se cumplen las dependencias multivaluadas R.A ----→ R.B|R.C, se puede descomponer sin pérdidas en dos relaciones en 4FN, mediante sus dos proyecciones R1(A,B) y R2(A,C).

En nuestro ejemplo: R1(asignatura, profesor) y R2(asignatura,texto)

R1

ASIGNATURA	PROFESOR
Programación en C	Javier Krahe
Programación en C	Alberto Pérez
Telemática	Javier Krahe

R2

ASIGNATURA	TEXTO
Programación en C	Lenguaje C
Programación en C	Metodología de la programación.
Telemática	Lenguaje C
Telemática	Comunicaciones con Linux
Telemática	Teleinformática

Desde un punto de vista práctico, nos basta saber que cuando estamos intentando normalizar una relación no normalizada, el primer paso debe ser "*separar los grupos repetitivos independientes*", una regla de sentido común.

5.1.1. Normalización práctica.

Veamos algunos aspectos a tener en cuenta si queremos ser prácticos cuando estamos normalizando:

- En ocasiones habrá razones válidas para no normalizar por completo. El ejemplo clásico de un caso en el cual la normalización completa podría no ser conveniente es la relación de nombres y direcciones:

  NOMDIR(Nombre, Calle, CódigoPostal, Localidad, Provincia)

  Donde encontramos las siguientes dependencias funcionales:

  Nombre ⟶ (Calle, CódigoPostal, Localidad, Provincia)
  CódigoPostal ⟶ (Localidad, Provincia)

  Según esto, esta relación estaría en 2FN, ya que existe una dependencia transitiva. Según lo visto convendría descomponerla en las relaciones:

  NOMDIR(Nombre, Calle, CódigoPostal)
  CODIGOS(CódigoPostal, Localidad, Provincia)

  Sin embargo, como Calle, CódigoPostal, Localidad, y Provincia, casi siempre se necesitan juntos, y como los códigos postales no

se modifican con mucha frecuencia, no es muy probable que una descomposición así valiera la pena.

- Cuando normalizamos, es posible que nos encontremos con relaciones degeneradas que contengan como único atributo una clave; estas relaciones, generalmente pueden eliminarse. También pueden aparecer relaciones con la misma clave, que podrían combinarse siempre que el resultado final sea lógico.

- Veamos cómo resolvería una B.D. relacional el típico problema de estructura en árbol de la ***lista de materiales***, en la que cualquier componente fabricado está a su vez compuesto por varios componentes menores y así sucesivamente hasta los componentes más elementales. Este problema lleva al establecimiento de una serie de niveles jerárquicos de componentes:

PRODUCTO (01)
GRUPO (02)
SUBGRUPO (03)
PIEZA (99)
PIEZA (99)
...
SUBGRUPO (03)
PIEZA (99)
PIEZA (99)
...
GRUPO (02)
...

Además pueden existir enlaces directos desde un nivel superior (PRODUCTO) a un nivel inferior (SUBGRUPO o PIEZA). La solución mediante un modelo lógico relacional sería la siguiente:

ARTICULOS (CódigoArtículo, Nivel, Descripción, ...)
DESCOMPOSICIÓN (CódigoArtículo, CódigoArtComponente, Cantidad)

Para cada artículo, bien sea producto, grupo, subgrupo, o pieza, existe una tupla en la relación de ARTÍCULOS. La descomposición de cada artículo se expresa mediante una serie de tuplas de la relación DESCOMPOSICIÓN.

Veamos un ejemplo de la descomposición de un artículo "*PC multimedia*":

(01) PC
- (99) Monitor
- (99) Teclado
- (99) Ratón
- (99) Altavoces

(02) Torre
- (99) Disco duro
- (99) CD-ROM
- (99) Disquetera
- (99) Tornillos

(03)Placa base
- (99) Microprocesador
- (99) Memoria

ARTÍCULOS:

CódigoArtículo	Nivel	Descripción
PC01	01	PC Multimedia
A013	99	Monitor
A014	99	Teclado
A015	99	Ratón
A016	99	Altavoces
TR17	02	Torre
A018	99	Disco duro
A019	99	CD-ROM
A022	99	Disquetera
A023	99	Tornillos
PB24	03	Placa base
A025	99	Microprocesador
A026	99	Memoria 32 M

DESCOMPOSICIÓN:

CódigoArtículo	CodArtComponente	Cantidad
PC01	A013	1
PC01	A014	1
PC01	A015	1
PC01	A016	1
PC01	TR17	1
TO17	A018	1
TO17	A019	1
TO17	A022	1
TO17	A023	12
TO17	PB24	1
PB24	A025	1
PB24	A026	2

- Como comentario final decir que una vez obtenido el modelo conceptual mediante el modelado entidad-relación, si transformamos los objetos conceptuales en relaciones (modelo lógico) según las reglas vistas, es habitual obtener relaciones totalmente normalizadas.