# 4

# NORMALIZACIÓN Y ALGEBRA RELACIONAL.

## NORMALIZACIÓN

## Objetivos de la normalización

El objetivo de la normalización es conseguir un “buen” diseño en forma de tablas del modelo relacional de un sistema de información.

¿Qué significa un “buen” diseño?

Un buen diseño es aquel que permita que el modelo que representa las operaciones de la empresa sea flexible, es decir que permita que se extienda cuando necesite representar nuevos atributos, entidades o relaciones y que evite la redundancia, tanto para ahorrar espacio como inconsistencias de los datos.

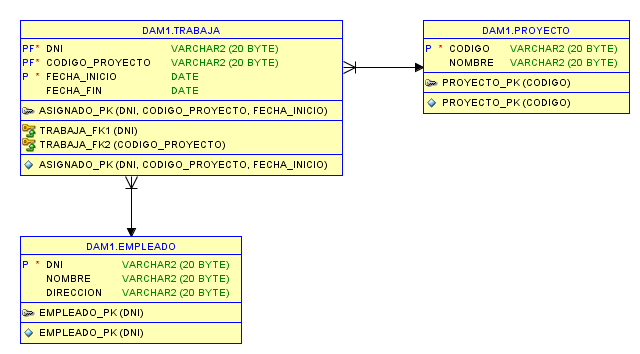
Este último punto, la evitar la redundancia, será la que evite que aparezcan tres tipos de anomalías, denominadas en conjunto anomalías de actualización, no deseables en cualquier base de datos: la anomalía de inserción, de borrado y de modificación.

#### Ejemplo:

Tras analizar la definición de un sistema de información, se ha creado su diagrama E-R siguiente:



Se han creado las siguientes tablas como transformación del modelo E-R al relacional:



Una tabla empleado, que recoge los datos de cada empleado, una tabla proyectos donde se reflejan todos los proyectos, y una tabla trabaja en la que se reflejan que empleados trabajan en cada proyecto, con sus correspondientes claves ajenas a las tablas empleado y proyecto, y con dos atributos que son las fechas de inicio y fin de trabajo en el proyecto.

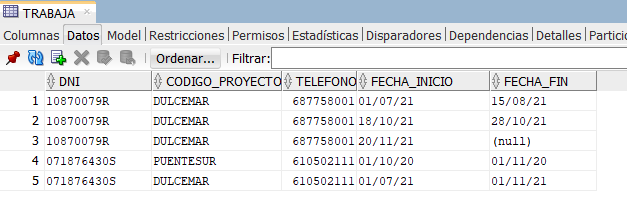
Adicionalmente, como se desea registrar que un empleado trabaja varias veces en el mismo proyecto, en fechas distintas, hemos añadido a la clave primaria la fecha de inicio.

Pero una vez hecho el diseño, se nos indica que los empleados que trabajen en proyectos deben de proporcionar un número de teléfono para que se les pueda localizar si surge algún imprevisto.

Y alguien, piensa que, esta es una buena solución:



Se comienzan a introducir datos y así queda la tabla trabaja, tras un tiempo:

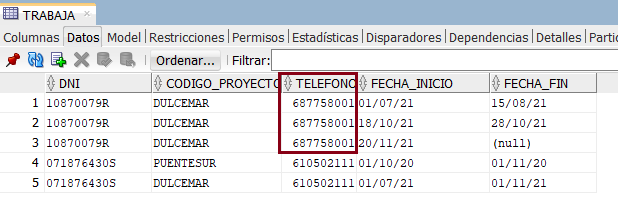


### 1.2 Anomalías de actualización: inserción, modificación y borrado

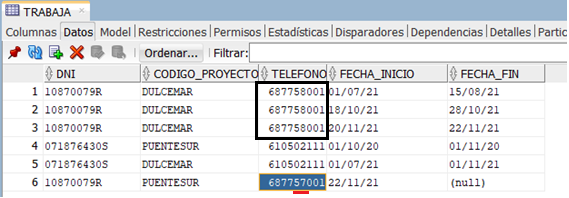
Se denominan así a las anomalías que aparecen en la información almacenada en la base de datos, debidas a la existencia de información redundante en la inserción, borrado y modificación de los datos.

#### Inserción

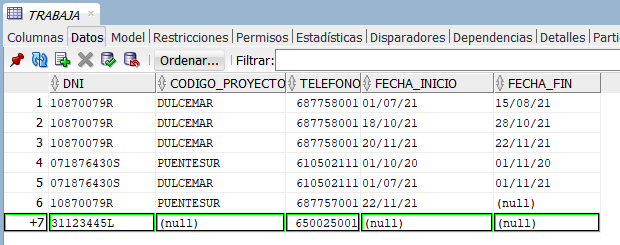
En la tabla TRABAJA, cada vez que un empleado trabaja en un proyecto hay que introducir su teléfono.



Si nos equivocamos de teléfono, al incluir al empelado en un nuevo proyecto, aparece una **inconsistencia** en los datos, que hará imposible saber cuál es el teléfono real del empleado. Ejemplo:



Si quiero almacenar el teléfono del empleado, pero aún no tengo un proyecto al que asignarlo, no puedo porque, código proyecto y fecha\_inicio forma parte de la clave primaria.

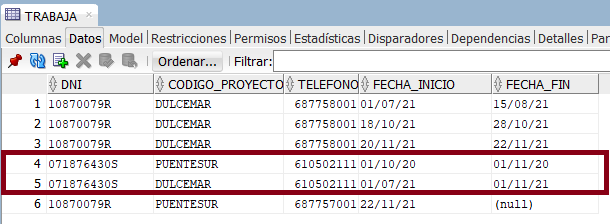


Entonces aparecen “proyectos ficticios”, **relaciones artificiales**, para solucionar esto como el 0000, que todavía añaden más inconsistencia a la base de datos, para solucionar un diseño incorrecto.

#### Borrado

Se produce cuando, al borrar información de una relación, se **borra información que no sería deseable** perder.

En el ejemplo anterior, si se borra la información marcada se pierde el teléfono del empleado.



#### Modificación

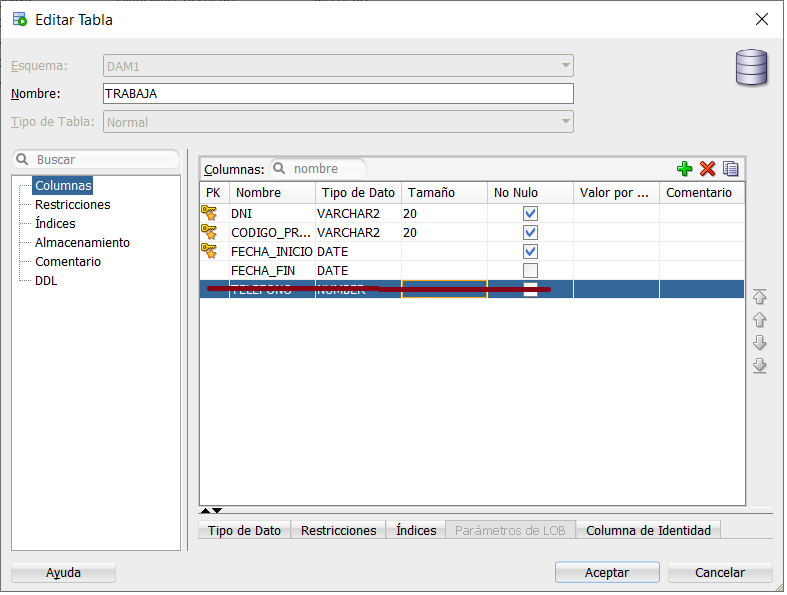
Actualizar un dato, tiene un **coste computacional muy alto**, pues si esta tabla tiene cien mil o un millón de registros, deberá comprobarse en cada uno, si el DNI es el de la persona a cambiar y si lo es cambiarlo, en todas las filas en las que se encuentre, cuando esto podría hacerse en una única operación, sobre una única fila.

Siempre, el origen de estas anomalías será la redundancia o repetición innecesaria de un determinado dato, y para evitarlas habrá que normalizar.

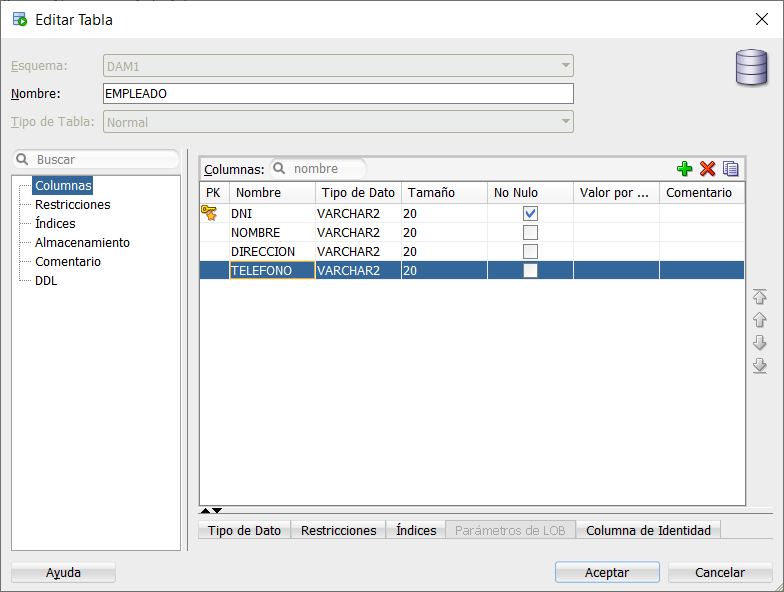
Normalizar es un proceso, un conjunto de pasos con los que se verifica que el diseño es “bueno” porque evita que aparezcan estas anomalías.

En la práctica normalizar va a consistir en quitar aquello que está mal en una tabla y llevárnoslo a otra tabla.

En el ejemplo, lo quitaríamos de la tabla TRABAJA



Y lo llevaríamos a EMPLEADO.



## Formas normales

Una tabla o relación está en una forma normal específica, si satisface el conjunto de requisitos establecidos para esa forma.

Las formas normales son distintos grados o requisitos que les vamos a exigir a las tablas para decir si esa tabla es correcta o no.

Las formas normales están anidadas, esto quiere decir que las formas normales inferiores eliminan los fallos que ocurren en las superiores, pero satisfaciendo las condiciones de esa forma normal superior.

Normalizar significa tratar de poner en la forma normal más alta, todas las tablas que tienen o satisfacen unas características susceptibles de ser normalizadas.

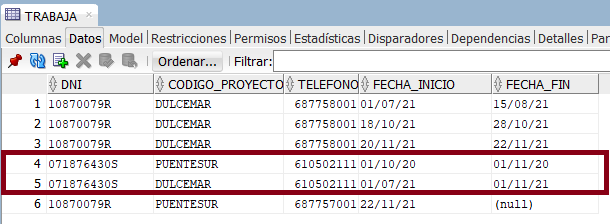
Al analizar porqué se producen las anomalías se detectaron que en las tablas existen un tipo de dependencias denominadas dependencias funcionales.

### 1.4 Dependencias funcionales

Las dependencias funcionales, nos indican que en todo sistema de información existen atributos que dependen de otros. Es decir que, si se conoce el valor de un atributo, sólo habrá un único valor de otro atributo.

Esto llevado a tablas, que es lo que queremos normalizar, mejorar, nos indica que en toda tabla hay atributos que dependen de otros.

En la tabla **TRABAJA**, el atributo *teléfono*, depende de *dni*. Eso es una dependencia funcional.



Se indica así:

NIF -> Teléfono

Y se dice que NIF **determina funcionalmente** teléfono o que Teléfono **depende funcionalmente** de NIF.

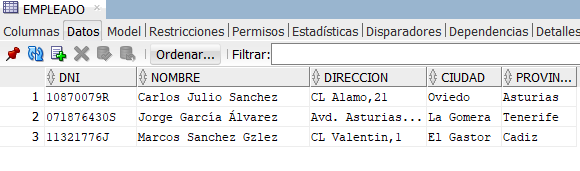
Y esta dependencia funcional ¿qué consecuencias tiene en la base de datos? Que cada *nif* tiene uno y sólo un *teléfono*.

### 1.5 Superclases, claves candidatas y claves primarias

Una **superclave** es el atributo o conjunto de atributos de una tabla que permiten distinguir una fila de otra.

Las superclaves pueden tener atributos que no son necesarios para distinguir una fila de otra. Interesa especialmente encontrar superclaves que no contengan tales atributos adicionales.

En la siguiente tabla:

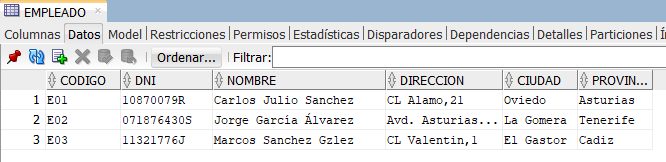


{DNI} es una superclave, {DNI, nombre} es una superclave, {DNI, nombre, dirección, ciudad} es una superclave, ya que introducidos valores para esos atributos, obtendremos **una y sólo una** fila de la tabla. Observamos que, si le añadimos nuevos atributos a una superclave, sigue siendo superclave.

Una **clave candidata**, es una superclave en la que ningún subconjunto de atributos es una superclave, es decir es un identificador mínimo.

En el ejemplo anterior, {DNI, nombre} no es una clave candidata, pues tiene un subconjunto, {DNI} que es superclave.

Una relación (tabla) puede tener varias claves candidatas, imaginemos que incluimos en la tabla anterior un código de empleado.



{dni} es una clave candidata y {código} es otra clave candidata. {dni, código} no sería una clave candidata, sería una superclave, pues tienen un subconjunto que permite identificar de forma única a la fila.

En resumen, una clave candidata es una superclave formada por los atributos mínimos para identificar una fila de forma unívoca.

Una clave primaria es una de las claves candidatas, que se decide, se usará para identificar la tuplas, porque es la que, en el mundo real, en la empresa, se usa de forma habitual para identificarlas o porque, por privacidad es mejor no usarlos o porque un extranjero no tendría por qué tenerlo.

Las claves candidatas tienen una dependencia funcional recíproca de una a la otra. Es decir

**código -> dni** y **dni -> código**

Y además el resto de atributos son dependientes funcionalmente de ellos.

**código** **-> {dni, nombre, dirección, ciudad, provincia}**

**dni** **-> {código, nombre, dirección, ciudad, provincia}**

### 1.6 Dependencias funcionales y claves en el diagrama E-R

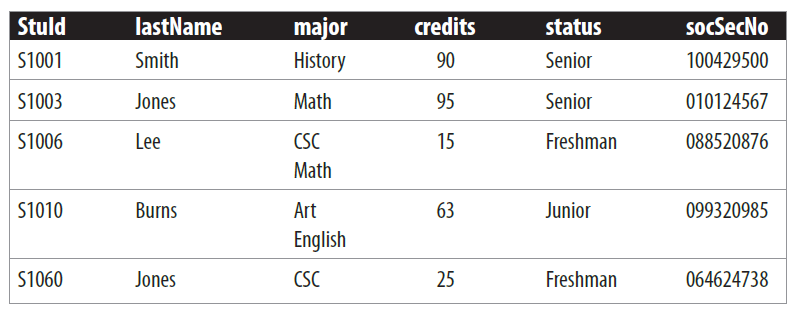
Cuando realiza un trabajo de diseño de un diagrama E-R y posteriormente se mapea en el conjunto de relaciones o tablas del modelo relacional, habitualmente, el diseñador ya ha identificado las dependencias funcionales y ha identificado las claves primarias, y por tanto las tablas ya están bastante normalizadas. Para completar el proceso, se debe comprobar cada tabla, identificando dependencias funcionales por si existe alguna y normalizarlas más si es necesario.

### 2 Primera forma normal (1FN)

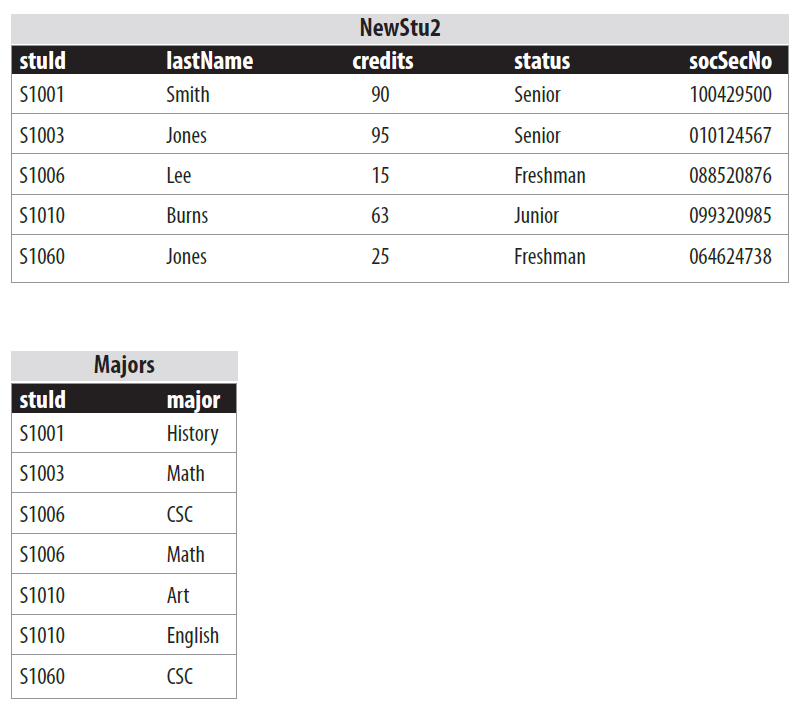
Una tabla está en primera forma normal si, cada atributo tiene un valor sencillo en cada fila.

Esto se consigue utilizando dominios simples para los atributos como enteros, caracteres, fechas, etc.

En el ejemplo siguiente, en esa tabla, el atributo *major*, si pudiese almacenar dos valores, no estaría en primera forma normal.



Habría que transformarla en varias tablas donde el atributo *major* sólo permitiese un valor atómico:



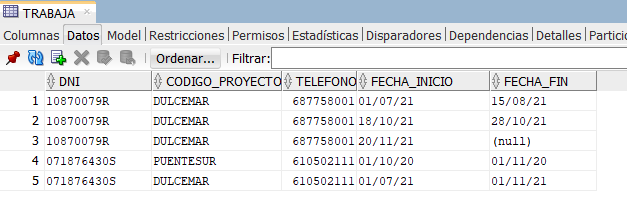
Además, la tabla debe de tener una clave primaria.

Desde un punto de vista práctico, como estudiantes que ya conocemos la tecnología donde vamos a almacenar nuestra información, es muy difícil que diseñemos tablas que no estén en 1FN.

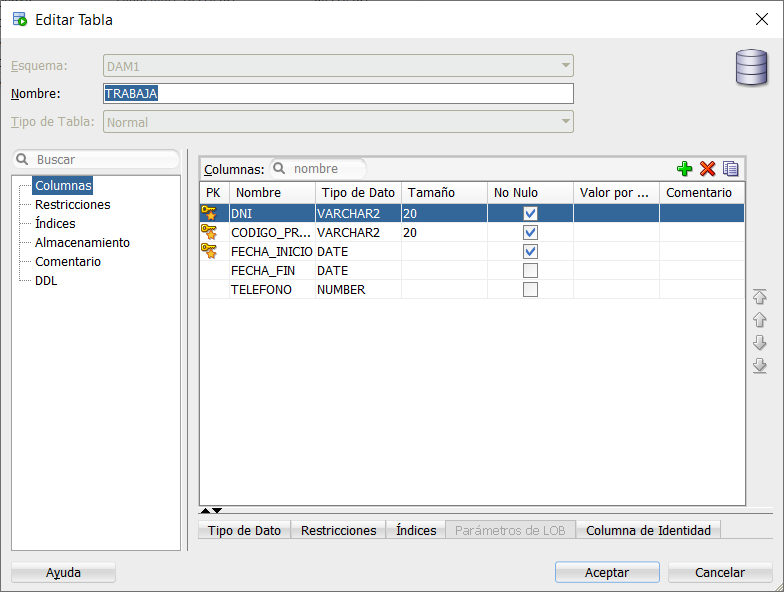
### Segunda forma normal (2 FN)

Una tabla o relación, está en segunda forma normal si está en primera forma normal y si todos los atributos no clave, dependen funcionalmente, completamente de la, o de las, claves candidatas.

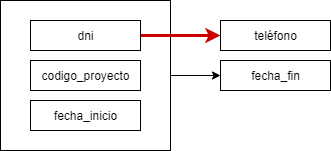
En nuestro ejemplo:



La tabla está definida así:



Podríamos hacer el gráfico siguiente de dependencias funcionales:



*fecha\_fin*, depende de forma completa de la clave primaria (sólo hay una clave candidata, es la clave primaria) mientras que teléfono sólo depende de dni que no es la clave primaria, forma parte de ella, pero no es la clave primaria.

Encontrar este tipo de dependencias, no completas, hace que esta tabla no cumpla con la segunda forma normal (2FN)

Esa tabla no estará en segunda forma normal, aparecerán redundancias.

Para solucionarlo, aquello que está mal, se saca a una tabla nueva o se lleva a una tabla ya existente, como hemos hecho con teléfono, que lo hemos llevado a donde debería estar, en la tabla empleado.

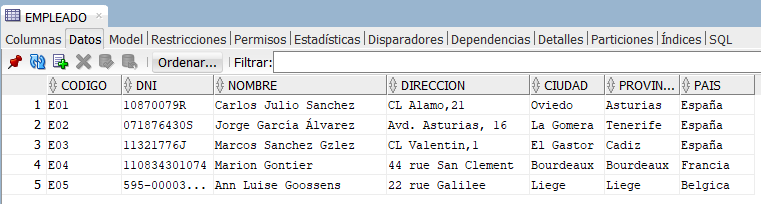
Cuando una relación tiene sólo un atributo como clave primaria, la relación está automáticamente en segunda forma normal (2FN). Sólo hay que preocuparse por la 2FN cuando la clave primaria es compuesta, por ejemplo, en las tablas que derivan de relaciones del modelo E-R o en tablas que nos encontramos por ahí que no sabemos de dónde derivan.

### Tercera forma normal (3 FN)

Aunque las tablas en segunda forma normal ya son mejores que las que están en primera, aún pueden aparecer anomalías, redundancias.

Decimos que una tabla está en tercera forma normal, cuando todos los atributos **no clave** dependen de manera **no transitiva** de cualquier clave candidata.

En nuestro ejemplo, vemos que aparecen anomalías y redundancias en el atributo *país*.



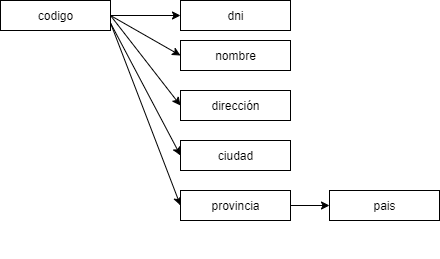
Es debido a que, pese a que todos los atributos dependen funcionalmente de código, hay una dependencia transitiva

provincia -> país

es decir, el país de un empleado se puede obtener por el código de empleado, pero también se puede obtener a partir de la provincia del empleado.

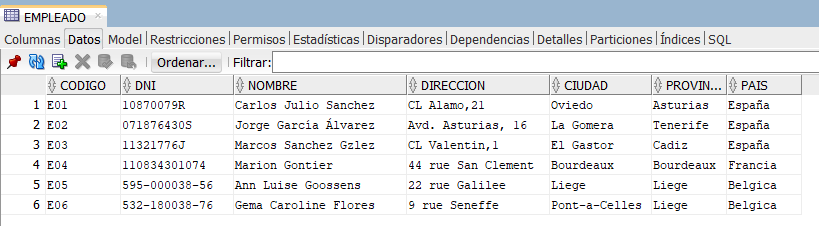
¿Como podemos definirla de forma informal? Como atributos a los que se puede llegar a través de otros atributos.

Si pintamos el gráfico de dependencias funcionales



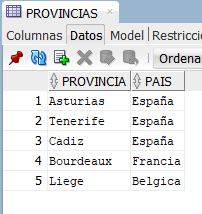
Cada vez que digo la Provincia tengo que introducir el país, aunque se repita. Esa información la puedo indicar una única vez.

Por ejemplo, a medida que introducimos más y más empleados, se van a ir repitiendo países cuando la provincia sea la misma.



Esto no está bien, general anomalías de inserción, borrado y modificación.

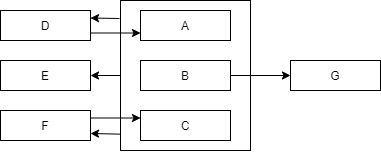
¿Qué es lo que se debe hacer? Sacar una nueva tabla, donde se registrarán las provincias y poner una clave ajena que las relacione.



### Forma normal de Boyce-Codd

Para saber si una tabla está en la forma normal de Boyce-Codd debemos recordar que son las claves candidatas y como reconocerlas en un diagrama de dependencias funcionales.

Dado este diagrama de dependencias funcionales, cuya clave primaria son los atributos {A, B, C},



también son claves alternativas {D,B,C}, {D, B, F} o {A, B, F}

Es necesario también introducir el concepto de **determinante**, como aquellos atributos o conjuntos de atributos de los que depende funcionalmente por completo algún otro atributo.

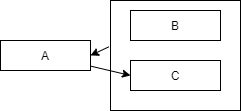
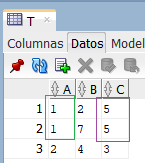
En el diagrama de dependencias funcionales anterior serían determinantes los atributos de los que salen flechas: (A,B,C), (B), (D) y (F).

El problema está en que algunos de estos determinantes no sean clave candidata.

Una relación está en forma normal de Boyce-Codd (FNBC) si y solo si, todo determinante es una clave candidata.

En el ejemplo, sólo el determinante (A,B,C) es clave candidata, (B), (D) y (F) no son claves candidatas por lo que la tabla no está en FNBD.

Otro ejemplo con datos:

Un valor de A determina un único valor de C. Por eso cuando A vale 1, C siempre tiene que valer lo mismo. El 1 siempre va acompañado del 5. Y eso es una redundancia.

Sin embargo, la clave primaria es (A,B), está en 1FN, está en 2FN porque como A es intercambiable por C, (B,C) es una clave alternativa, así que no hay atributos que no formen parte de alguna clave alternativa. Y lo mismo para 3FN. En estas formas los que generan problemas son los atributos que no son claves y aquí todos lo son, así que hasta 3FN no hay nada que normalizar porque ya están en 3FN.

Para la FNBC, los determinantes son (A) y (B,C). No está en forma Boyce-Codd porque (A) no es una clave candidata.

Y la FNBC dice que para estar en dicha forma normal, no deben existir determinantes que no sean clave candidata.

Para solucionarlo hay que sacar fuera lo que sobra.

En esta tabla lo que sobra es C, así que se hace como en el resto de formas normales, se quita de la tabla, se lleva a una tabla donde la clave primaria es o son los atributos de los que depende y se crea una clave ajena en la tabla normalizada.

T (A, B,C ) nos queda:

T (A, B) donde {A} referencia T1.

T1 (A, C)

Lo que importa para esta forma normal son los determinantes y las claves candidatas. Si algún determinante no es clave candidata, ese determinante está mal. En este caso para que deje de ser determinante, hay que sacar C fuera.

Es importante llegar a la forma normal de Boyce-Codd si hay varias claves candidatas y se solapan en atributos si hay que hacerla, sino ya estará en FNBC.

Se dice que se solapan en atributos si las claves candidatas comparten algún atributo.

En este caso como las claves candidatas (A, B) y (B, C) comparten B si hay que normalizar hasta la FNBC.

### Cuarta forma normal

Las tablas que no están en cuarta forma normal, están en FNBC, es decir, están en 1FN, 2FN y 3FN además de FNBC. No hay dependencias funcionales de atributos no clave de una parte de la clave primaria ni dependencias transitivas de atributos no clave.

Sin embargo sigue habiendo redundancia, que como vimos es una posible fuente de anomalías.

En las tablas de una base de datos, además de las dependencias funcionales, es decir que existan atributos que dado un valor impliquen un valor y solo un valor en otro atributo, existen atributos que dado un valor implican un conjunto de valores de un atributo, todos los posibles o no todos.

Este tipo de depedencias se llaman dependencias funcionales multivalor.

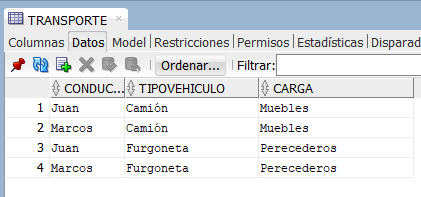
Sea R una relación que tiene atributos o conjuntos de atributos A, B y C. Existe una dependencia multivaluada del atributo B sobre el atributo A si y sólo si el conjunto de valores B asociados con un valor A dado es independiente de los valores C.

Hay que determinar si existen este tipo de dependencias en las tablas y especialmente si alguna de las columnas es independiente de dicha dependencia, pues esa columna se estará repitiendo.

Ejemplo:

Dada la tabla Transporte

**TRANSPORTE** (conductor, tipovehículo, carga)



En este caso se puede observar como para poder configurar que Juan y Marcos pueden conducir camiones y furgonetas se tiene que repetir que los Muebles van en camiones y los perecederos en furgoneta dos veces, es decir se repite esa información.

Y al revés, si se quiere configurar que los muebles van en camiones y los perecederos en furgonetas, tenemos que indicar que tanto Juan como Marcos los pueden conducir.

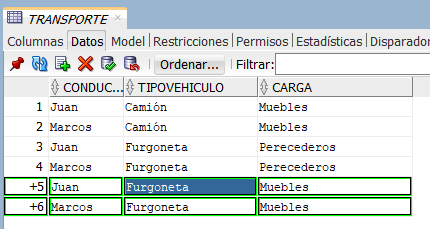
Una tabla (relación) está en 4FN si está en FNBC y además no tiene dependencias funcionales multivaluadas no triviales.

En la tabla transporte hay dos dependencias multivaluadas, los tipos de vehículos que puede conducir el conductor y los tipos de vehículos que pueden llevar una determinada carga. Se representa como

**Conductor ->> TipoVehículo**

**Carga ->> TipoVehículo**

El problema aparece cuando hay que indicar además que los muebles pueden ir en furgonetas y camiones, ahí si que se ven las redundancias.

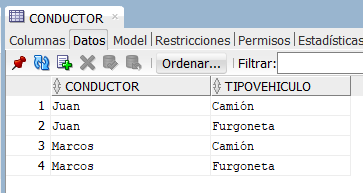
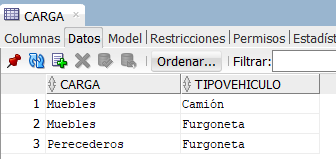


Para expresar que los Muebles pueden ir en Furgoneta, es obligatorio incluir una nueva tupla para indicar que además de Juan, Marcos también puede conducir Furgonetas para llevar Muebles.

Para los muebles no interesa el conductor que los lleva sino el vehículo en el que pueden ir. Y no interesa el conductor que lleva los muebles, sino los vehículso que puede conducir.

Para normalizarlo se crean tablas separadas para cada uno de los atributos que multideterminan y con los atributos multideterminados. Ambos atributos son clave primaria. En 4FN no hay claves ajenas.

En el ejemplo anterior, se lleva cada dependencia funcional a una nueva tabla. La clave primaria son las dos columnas y no hay claves ajenas.

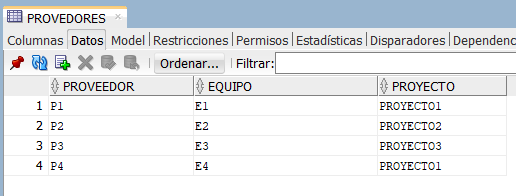
La tabla Transportes estaba en FNBC porque todas las columnas forman parte de la clave primaria, y por tanto no hay dependencias funcionales de campos no clave, pero no estaba en 4FN.

### Quinta forma normal

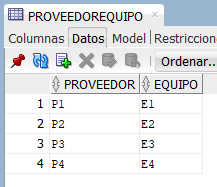
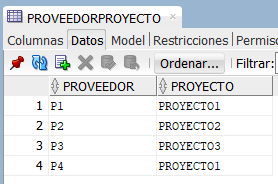
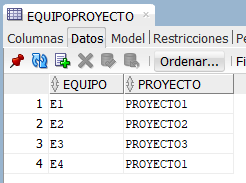
La quinta forma normal, se denomina también de proyección – unión.

Cuando tenemos una tabla en 4FN con n atributos (en la que no hay dependencias multivaluadas de un atributo con otro), y no podemos descomponerla en n relaciones más pequeñas (proyecciones), que después al combinarlas (uniones) nos devuelva la relación original.

Para ilustrarlo, vamos a usar un ejemplo.

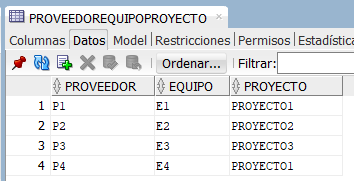


Si lo descomponemos en n tablas más pequeñas (proyecciones). En este caso 3 por tener 3 atributos:

Y volvemos a combinar las tres tablas, si nos da la tabla original, eso quiere decir que podríamos tenerlas en tablas más pequeñas sin perder información y no estaría en 5FN. Si al unirlas nos aparece información nueva, que no está en la tabla original, entonces si que estaría en 5FN, porque no puede ser descompuesta en tablas que al unirlas la regenere.

Se hace la unión entre PROVEEDOREQUIPO y PROVEEDORPROYECTO

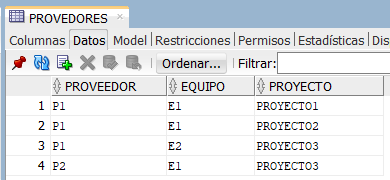


Y la resultante con EQUIPOPROYECTO, comprobando que no haya que insertar ninguna tupla nueva. Tras hacerlo no se inserta ninguna tupla más pues ya están en la relación anterior.

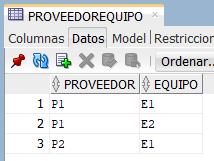
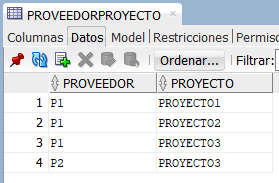
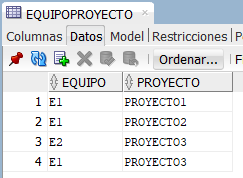
Al hacerlo se ve que se no se cumple la condición de 5FN, es decir que si se puede descomponer en tablas que al unirlas recuperen la información de la tabla original.

Para normalizarlas se transformarían en las tres tablas anteriores.

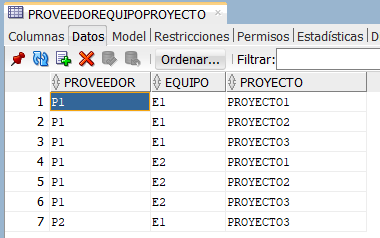
Si repetimos el ejemplo con la siguiente información



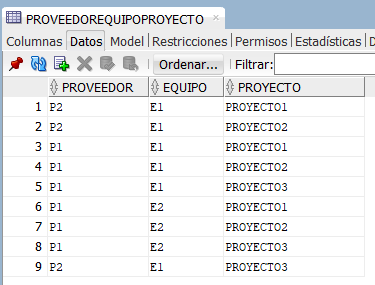
Y como hemos hecho antes, creamos las tres tablas más pequeñas

Se hace la unión de PROVEEDOREQUIPO con PROVEEDORPROYECTO



Y a continuación con EQUIPOPROYECTO y se comprueba si se deben incluir tuplas nuevas, que son las de P2 y E1 que tienen que aparecer con los tres proyectos.



Si se observa esta tabla, tiene más filas que la original, es decir después de hacer la unión no se puede llegar a la tabla original. Por eso la tabla original está en 5FN.