Normalización

Dada la siguiente estructura de datos relacional:

*LegajoEmpleado
NombreEmpleado
ApellidoEmpleado
CodCategoria
DescripcionCategoria
SueldoCategoria

Podremos notar algunas anomalías:

Anomalía de inserción: es fácil ver la imposibilidad de ingresar una categoría nueva junto con su descripción y sueldo si aún no hay al menos un empleado asignado a esa categoría, ya que la clave primaria de esta estructura es legajoEmpleado y por lo tanto no puede tener un valor nulo.

Anomalía de actualización: supongamos que hay 1287 empleados con categoría 1 y tenemos que modificar el sueldo de la categoría 1, habrá que modificar 1287 filas.

Anomalía de eliminación: consideremos ahora el caso de una categoría que tiene un solo empleado y ese empleado renuncia. Al dar de baja ese legajo, se pierde también toda la información de su categoría.

Estos 3 problemas quedarían resueltos si convertimos esta estructura inicial en las 2 siguientes estructuras:

*LegajoEmpleado NombreEmpleado ApellidoEmpleado CodCategoria (fk)	y	*CodCategoria DescripcionCategoria SueldoCategoria

De esta manera, se podría ingresar una nueva categoría aunque no hubiese empleados asignados, se modificaría solo el registro de la categoría 1 en vez 1287 registros de empleados con categoría 1 y también se podrían eliminar los datos de cualquier empleado sin perder los datos de alguna categoría, ya que los datos de las categorías se encuentran separados.

Para corregir estas anomalías, Codd desarrolló un procedimiento llamado Normalización.

Podemos definir a la normalización como un proceso de refinamiento de estructuras de datos que consiste en agrupar los datos siguiendo una serie de reglas o normas de manera repetitiva y ordenada tantas veces como sea necesario hasta que finalmente los datos queden normalizados cumpliendo ciertos criterios llamados FORMAS NORMALES.

Las formas normales más conocidas y empleadas en el diseño de Bases de Datos Relacionales son la 1FN, 2FN y 3FN que consisten en:

1FN: eliminar las estructuras repetitivas.

2FN: eliminar dependencias parciales de la clave primaria.

3FN: eliminar dependencias transitivas de la clave primaria.

Previamente a iniciar el proceso de normalización debemos conocer la semántica o significado de cada uno de los atributos, así como también las dependencias funcionales entre ellos. Comencemos por definir algunos términos nuevos como:
ATRIBUTOS PRIMOS que son aquellos atributos que forman parte de la llave o clave primaria.

Y obviamente los ATRIBUTOS NO PRIMOS: son los que NO forman parte de la clave primaria.

Dependencia funcional. Digamos entonces que dada una relación compuesta por un conjunto de atributos:

A = {a, b, c, d},

si el valor de a DETERMINA los valores de b, c y d, entonces

b, c y d DEPENDEN FUNCIONALMENTE DE a

Para el siguiente ejemplo, tenemos como atributos de la estructura de datos:

CodArticulo
DescripcionArt
PrecioArticulo

Dado un valor del atributo código de artículo quedan determinados los valores de los atributos DescripcionArt y PrecioArticulo. Es decir que para un cierto Código de artículo va a existir una sola descripción y un solo precio.

Esto es equivalente a decir que los valores de los atributos DescripcionArt y PrecioArticulo dependen funcionalmente del CodArticulo.

Para entenderlo mejor, pensemos en una base de datos que ya está creada y cargada con datos, si alguien me pide los datos del artículo con código 1352, solo tengo que acceder al registro con código de artículo 1352 y obtendré los datos solicitados. En cambio, con el valor del precio 100 seguramente podré obtener varios productos que cuesten $100. Por ello un precio NO DETERMINA un solo código de artículo.

También debemos entender que cuando se habla de DEPENDENCIA FUNCIONAL PARCIAL se refiere a la dependencia funcional de algún atributo no clave o no primo, que depende solo de una parte de la llave o clave primaria compuesta.

Veamos un ejemplo:

*LegajoProfesor
*CodigoCurso
NombreCurso

El atributo NombreCurso depende solo del código de curso independientemente de quien sea el profesor asignado.

La otra dependencia funcional existente es la DEPENDENCIA TRANSITIVA de la clave primaria, es decir que si un atributo c depende de otro atributo b, y a su vez este atributo b depende de la clave primaria a, entonces el atributo c depende de la clave primaria a.

Esto entonces lo podemos simplificar diciendo que existe dependencia transitiva de la clave primaria cuando existe algún atributo No primos que depende de otro atributo no primo.

Veamos ahora un ejemplo para analizar su semántica:

CodArticulo
DescripcionArt
CodRubro
DescripRubro
CodSucursal
NombreSucursal
Stock

Esta estructura de datos está compuesta por atributos que contendrán los valores correspondientes para cada instancia de la entidad Articulo, entonces:

CodArticulo es el atributo que contiene el código de un artículo.

DescripcionArt es el atributo que contiene la descripción de un artículo.

CodRubro es el atributo que contiene el código de un rubro.

DescripRubro es el atributo que contiene la descripción de un rubro.

CodSucursal es el atributo que contiene el código de una sucursal.

NombreSucursal es el atributo que contiene el nombre de una sucursal.

Stock es el atributo que contiene el stock de un cierto artículo en una determinada sucursal.

Veamos ahora las dependencias funcionales:

El código de artículo determina la descripción del artículo y el código de rubro.

Esto se escribe:

CodArticulo $\longrightarrow$ DescripcionArt, CodRubro

El código de rubro determina la descripción del rubro.

CodRubro → DescripRubro

El código de sucursal determina la descripción de la sucursal.

CodSucursal → NombreSucursal

Y el código de artículo junto con el código de sucursal determinan el stock.

CodArticulo, CodSucursal → Stock

Existen varios métodos para normalizar, por ejemplo en el libro de nuestra bibliografía obligatoria explican claramente un método para normalizar hasta la tercera forma normal.

Yo les voy a explicar otro método más que pueden usar para normalizar.

Volviendo a la estructura de datos anterior:

CodArticulo
DescripcionArt
CodRubro
DescripRubro
CodSucursal
NombreSucursal
Stock

Lo primero que hacemos es buscar el atributo dominante, es decir el atributo con el que se relacionan los demás atributos de la estructura y lo marcamos como futura clave primaria.

Podrían existir varios atributos dominantes o claves candidatas, en ese caso elegimos uno de ellos.

En este caso, código de artículo es el atributo dominante porque:

- Para un código de artículo existe una descripción de ese artículo.
- Un código de artículo corresponde a un rubro determinado, entonces se relaciona con un código de rubro y una descripción del rubro.
- Un código de artículo se vende en algunas sucursales, entonces se relaciona con códigos y nombres de sucursales.
- Un código de artículo junto con una determinada sucursal tienen cierto stock.

Una vez identificado el atributo dominante lo marcamos como clave primaria transitoria de esta estructura.

*CodArticulo
DescripcionArt
CodRubro
DescripRubro
CodSucursal
NombreSucursal
Stock

Ahora sí, comencemos a aplicar el proceso de 1Fn que consiste en eliminar las estructuras repetitivas, es decir aquellas que contienen atributos multivaluados.

Para ello veamos que:

Un artículo tiene asociado una descripción, un código de rubro, una descripción de rubro, pero puede tener asociados.

Varios códigos de sucursales.

Varias descripciones de sucursales.

Y varios stocks.

Por lo tanto:

CodSucursal

NombreSucursal

Stock

Es la estructura repetitiva que debemos eliminar.

Para ello deberemos aplicar de manera consecutiva tantas veces como sea necesario los siguientes pasos hasta obtener estructuras que estén en 1fn, es decir sin estructuras repetitivas.

Paso 1: copiar la parte de la estructura que ya está en 1FN (todos los atributos tienen un único valor).

*CodArticulo
DescripcionArt
CodRubro
DescripRubro

Paso 2: transformar las estructuras repetitivas aplicando los siguientes pasos tantas veces como sean necesarios para obtener nuevas estructuras con atributos con un solo valor.

2 a) Copiar la estructura repetitiva.
CodSucursal
NombreSucursal
Stock

2 b) Elegir su clave primaria transitoria.
* CodSucursal
NombreSucursal
Stock

2 c) Bajar la clave de la que depende la estructura repetitiva (CodArticulo) como clave foránea.

CodArticulo (fk)
* CodSucursal
NombreSucursal
Stock

2 d) Verificar que la fk no quede multivaluada, y en caso de que estuviese multivaluada hacerla también parte de la clave primaria es decir, atributo primo.

En este caso, en una sucursal puede haber VARIOS artículos, por lo tanto debemos convertirla en atributo primo.

*CodArticulo (fk)
* CodSucursal
NombreSucursal
Stock

Este proceso terminó, solo queda verificar que no haya más estructuras repetitivas, y en caso de haberlas repetir estos pasos desde el paso 1 tantas veces como sea necesario hasta que no queden estructuras repetitivas.

En este caso:

Cada código de artículo en cada sucursal tiene una sola descripción de sucursal y un solo stock, por lo tanto terminamos la primera forma normal y las 2 nuevas estructuras refinadas serán:

1FN
*CodArticulo
DescripcionArt
CodRubro
DescripRubro
* CodArticulo (fk)
* CodSucursal
NombreSucursal
Stock

Para la segunda forma normal: 2FN se deben eliminar todas las dependencias parciales de la clave primaria.

Se deben tomar cada una de las estructuras obtenidas en la 1FN y repetir los siguientes pasos ordenados tantas veces como sea necesario para que todas las estructuras queden en 2FN.

En el caso de la estructura:

*CodArticulo
DescripcionArt
CodRubro
DescripRubro

Ya se encuentra en 2FN porque no hay clave primaria compuesta y ya está en 1FN.

En cambio la estructura:

* CodArticulo (fk)
* CodSucursal
NombreSucursal
Stock

Tiene una dependencia funcional parcial que es:

El nombre de sucursal depende solamente del código de sucursal, por lo tanto debemos separar la dependencia parcial en una estructura nueva y dejar también en la estructura sin la dependencia parcial el atributo determinante como fk para que se sigan relacionando los datos. Por el contrario, el stock depende del artículo y de la sucursal, por ello debe quedar en la estructura de la clave primaria compuesta.

Es decir que quedará

*codSucursal
NombreSucursal

Y:

* CodArticulo (fk)
* CodSucursal (fk)
Stock

Como este proceso es repetitivo, debemos analizar las 3 nuevas estructuras para ver si sigue habiendo dependencias parciales de la pk.

En este caso, las 3 estructuras están en 2FN, porque las 2 primeras tienen clave primaria simple y en la tercera estructura el único atributo no primo depende de toda la clave primaria. Y por lo tanto nos quedarían las siguientes estructuras en 2FN:

*CodArticulo
DescripcionArt
CodRubro
DescripRubro

*codSucursal
NombreSucursal

* CodArticulo (fk)
* CodSucursal (fk)
Stock

Entonces recién ahora podemos pasar a la 3FN que consiste en eliminar dependencias transitivas de la clave primaria, es decir atributos no primos que dependen de otros atributos no primos.

En la estructura:

*CodArticulo
DescripcionArt
CodRubro
DescripRubro

Encontramos que la descripción del rubro depende del código de rubro, entonces debemos eliminar esa dependencia transitiva y para ello el método consiste en separar el atributo no primo que depende del otro atributo no primo juntos, tomando como nueva clave primaria de esta nueva estructura al atributo determinante y dejando el resto de la estructura con el atributo no primo dominante como fk para mantener la relación entre las nuevas estructuras:

*CodRubro
DescripRubro

*CodArticulo
DescripcionArt
CodRubro(fk)

Aquí concluye el método de la 3FN, entonces solo nos queda analizar si las nuevas estructuras ya están en 3FN o si debemos comenzar de nuevo a aplicar el método de 3FN a cada una de ellas hasta que no haya dependencias transitivas.

En este caso vemos que en:

*CodRubro
DescripcionRubro

Hay un solo atributo no primo, por lo tanto no puede depender de ningún otro.

Y en el caso de:

*CodArticulo
DescripcionArt
CodRubro(fk)

Los atributos no primos no dependen entre ellos, ya que una descripción de artículo no determina el código de rubro ni el código de rubro determina la descripción del artículo.

Por lo tanto está en 3FN.

En cuanto a la estructura:

*codSucursal
NombreSucursal

Ya está en 3FN porque tiene un solo atributo no primo.

Y la estructura:

* CodArticulo (fk)
* CodSucursal (fk)
Stock

También tiene un solo atributo no primo.

Por lo tanto todas nuestras nuevas estructuras están en 3FN.

Por último podemos ver que con la normalización se pueden corregir las 3 anomalías del diseño de BDR que son las anomalías de inserción, actualización y eliminación.