¿ Y después de la 3FN… ? FNBC, 4FN y 5FN

FORMA NORMAL DE BOYCE-CODD (FNBC)

Tras la aplicación de la 3FN se observó que se encontraban algunas anomalías que no eran abordadas. Son casos de tablas que, estando en 3FN, mantienen una dependencia de un atributo secundario con parte de la clave, es decir, parte de la clave depende funcionalmente de un atributo secundario.

Gráficamente sería de la siguiente forma:

T (A, B, C)

A | --> C|

B <--| -------|

Por ello, se pensó una definición más global que abordase estas anomalías. La nueva definición se debe a Boyce y a Codd.

" Una tabla T está en FNBC si y sólo si las únicas DF elementales son aquellas en las que la clave principal (y claves secundarias) determinan un atributo"

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

Si la clave está formada por un único atributo y se encuentra en 3FN la tabla se encuentra en FNBC.

Veamos como ejemplo la siguiente tabla en la que almacenamos los quesos típicos de cada región y el país al que pertenece la región:

QUESOS (Queso, pais, región).

Cabrales,España, Asturias

Beyusco, España, Asturias

Livarot, Francia, Normandía

Grana, Italia, Parma

Encontramos las siguientes DF:

Queso.pais --> región

región --> país

Esta relación se encuentra en 3FN ya que ningún atributo secundario es conocido a través de otro atributo secundario. Sin embargo no se encuentra en FNBC, ya que región (atributo secundario), me permite conocer parte de la clave (país), lo que origina redundancias perjudiciales.

La información en la tabla está fuertemente relacionada. Si mantenemos la dependencia región --> país en la misma tabla resulta que existen muchas tuplas que se corresponden

con el mismo país, tanta como quesos haya en las regiones del país, lo que genera los problemas conocidos de actualización (MIB), consultas, y mala gestión de la memoria. Por otra parte, si eliminamos la tupla Grana, Italia,Parma, desaparece la relación entre Parma e Italia. Por todo ello, la tabla no se encuentra optimizada.

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

1.- Sea la DF X --> Y donce X e Y son disjuntos, X es un atributo no primario e Y forma parte de la clave.

2.- Se obtienen las proyecciones:

2.1- T1 = P (tabla, X,Y)

2.2- T2 = P (tabla, U - Y) siendo U todos los atributos que componen la tabla.

Por tanto, para que una tabla que está en 3FN y no cumple la FNBC se encuentre en FNBC, se realizan las siguientes proyecciones:

1.- Se crea una tabla con la parte de la clave que depende funcionalmente del atributo secundario (país) y el atributo secundario (región). La clave será el atributo secundario.

REGIONES=P(QUESOS,<u>región,</u> país)

2.- Se crea otra tabla con la parte de la clave que es independiente (queso) y todos los atributos no primarios. La clave de esta tabla será, generalmente, la concatenación de la parte de la clave independiente y el atributo secundario que determina parte de la clave (región), aunque se decide siempre en función de las DF existentes.

T2=P(QUESOS, <u>queso, región</u>) Serán clave ambos atributos si región no DF de queso, es decir si dado un queso no puedo determinar una única región.

Así, a partir del atributo región se puede obtener la relación original.

Ej. CALLEJERO (<u>Dirección, Ciudad</u>, Cod_postal)

C/Pez, 2	Móstoles	28823
C/Luz, 5	Móstoles	28823
C/Pez, 2	Madrid	28007
C/Mar, 4	Madrid	28019

Con las siguientes DF:

Cod_postal --> Ciudad

Dirección.ciudad --> Cod_postal.

No se encuentra en FNBC, ya que existe a partir de una atributo secundario podemos conocer parte de la clave. La convertimos a FNBC.

1.- CALLEJ_CIUDAD= P (CALLEJERO, <u>cod_postal</u>, ciudad)

2.- CALLEJ_DIRECC= P (CALLEJERO, <u>Dirección, cod_postal)</u>

DEPENDENCIA MULTIVALUADA (DMV)

La DMV se utiliza para tratar la 4FN. Se define como: "Sean A, B y C tres subconjuntos distintos de atributos de una tabla T, se dice que A tiene una DMV con B, que A multidetermina a B o que B depende multivaluadamentes de A y se escribe como A--->-> B ó A DMV B, si para cada valor de A existen un conjunto de valores de B asociados, y esta asociación es independiente del resto de atributos C. (Es decir, que los valores de C no influyen en esa asociación).

Como es necesario que en una DMV entre 2 atributos el resto de los campos sean independientes, es necesaria la existencia de al menos 3 atributos para que se produzca una dependencia multivaluada.

Por ej. sea la tabla PROF_ASI_TEX que representa a los profesores, asignaturas que imparten y los textos de bibliografía de las asignaturas. Se parte de los condicionantes siguientes:

a) Un profesor imparte varias asignaturas y una asignatura puede ser impartida por varios profesores.

b) Una asignatura tiene varios textos y un texto se puede utilizar en varias asignaturas, independientemente del profesor que imparta la asignatura.

PROF_ASI_TEX (Profesor, asignatura, texto) Al no existir DF entre los atributos la clave está formada por la concatenación de los tres.

Existen por tanto las DMV siguientes:

Asignatura -->--> Profesor (independientemente del texto)

Asignatura -->-> Texto (independientemente del profesor)

pudiendo respresentarse como Asignatura -->-> Profesor | Texto.

Sin embargo, no existe Profesor -->-> texto, puesto que no es independiente de Asignatura.

CUARTA FORMA NORMAL (4FN)

La 4FN se aplica para eliminar las DMV de las tablas, puesto que implican redundancias perjudiciales de información, y por tanto problemas de actualización e integridad de los datos.

La 4FN la enunció Fagin tras el teorema que demostró y que dice: "Una tabla T con atributos A, B y C se puede descomponer sin pérdidas en 2 proyecciones T1(A, B) y T2(A, C) si y sólo si la DMV A -->-> B | C se cumple en T"

Así, se dice que una tabla está en 4FN si cumple 2 condiciones:

1.- Está en FNBC

2.- Las únicas DMV existentes son las DF de la clave con los atributos secundarios.

Entre losa tributos que forman la clave no puede existir más de una relación 1:N.

El algoritmo de descomposición de una tabla que no cumple la 4FN es el siguiente:

a) Sea una DMV X -->-> Y donde X e Y son disjuntos.

b) Se realizan 2 proyecciones.

b1) T1 = P(tabla, X,Y)

b2) T2 = P(tabla, U-Y) siendo U todos los atributos que componen la tabla.

Una tabla que no está en 4FN, se pasa a 4FN realizando, por tanto las siguientes proyecciones:

1.- En una tabla se mantiene una DMV

2.- En otra tabla se deja la otra DMV, de modo que los atributos que dependen multivaluadamente quedan cada uno en una tabla.

Según el ej.

1.- ASI_PROF= P(PROF_ASI_TEX, asignatura,profesor)

2.- ASI_TEX= P(PROF_ASI_TEX, asignatura, texto).

Ej. Dada la tabla ESTUDIANTE (Num_expte, Asignatura, Deporte) que guarda la información de las asignaturas que estudía el alumno y los deportes que practica, (siendo estos independientes de la asignatura que curse).

Encontramos las DMV: Num_expte -->-> Asignaturas

Num_expte -->-> Deportes

No se encuentra en 4FN. La transformamos:

1.- AL_ASIG= P(ESTUDIANTE, Num_expte, Asignatura)

2.- AL-DEP = P(ESTUDIANTE, Num_expte, Deporte)

DEPENDENCIA DE JOIN (DJ)

La dependencia de join es una dependencia entre tablas. Hay casos en los que las tablas que se encuentran en 4FN no están aún totalmente optimizadas, bien porque son pequeñas y tienen mucha redundancia, al realizar determinadas operaciones de consultas frecuentes,o bien por ser muy grandes con entidades diferentes relacionadas con una cardinalidad 1:1 y resultan intratables en las transacciones que las manejan.

En esos casos conviene realizar proyecciones para que las tablas resultantes sean más tratables.

La DJ indica que una tabla T formada por los atributos A1, A2,...,An tiene una DJ con sus proyecciones T1, T2,...Tn, si la tabla T original se puede obtener por medio del join de sus proyecciones T1*T2*.....*Tn.

Por ej. sea la Tabla T (A, B, C)

a1,b1,c1

a2,b1,c1

a1,b2,c1

a1,b1,c2

Podríamos manejar tablas más pequeñas, con menos columnas, eliminando la redundancia, creando las siguientes proyecciones:

T1= P(T, A , B)	T2=P(T,B,C)
a1,b1	b1,c1
a2,b1	b2,c1
a1,b2	b1,c2

Si mediante un join se intenta recuperar la tabla original, resulta:

T12= T1 * T2 (T1.B=T2.B) => T(A, B, C)

a1, b1, c1
a1, b1, c2
a2, b1, c1
a2, b1, c2 => Tupla intrusa que no está en la original
a1, b2, c1

A las tuplas que aparecen y no existían en la tabla originial se las denomina Tuplas intrusas. Esta anomalía indica que la proyección no está bien realizada. Cualquier otra combinación de 2 proyecciones de la tabla original volverá a generar tuplas intrusas. Sin embargo, la creación de 3 proyecciones, las tres posibles, dan como resultado de realizar su join la tabla original. La proyección que falta es T3 = P(T, A, C)

a1,c1
a2,c1
a1,c2

de forma que al realizar el join con la tabla T12 por los campos comunes, resulta la tabla original, eliminándose las tuplas intrusas:

T12*T3 (T12.A=T3.A y T12.C=T3.C) => T(A, B, C)

a1, b1, c1
a1, b1, c2
a2, b1, c1
a1, b2, c1

Se puede decir entonces que la tabla T mantiene una DJ con sus proyecciones T1,T2 y T3 y se escribe como:

T (A, B, C) DJ (T1 (A,B), T2(B, C), T3(A, C))

En una tabla que se pretende seguir descomponiendo no se trata de realizar combinaciones de sus atributos en las diferentes proyecciones que se puedan generar, ni tampoco probar a descomponer realizando el join con algunos valores, puesto que pueden no aparecer tuplas intrusas inicialmente y en futuro aparecer. (En el ej. anterior eliminando la tupla original a1,b1,c2 habría resultado que las 2 primeras proyecciones T1 y T2 serían correctas cuando realmente no es así).

La norma a seguir aparece en la definición de la 5FN.

QUINTA FORMA NORMAL (5FN)

Para que una tabla se encuentre en 5FN se deben cumplir 2 condiciones:

1.- Se encuentra en 4FN

2.- Toda DJ viene implicada por las claves (principales o secundarias) de la tabla.

Es decir, una tabla estará en 5FN si es posible generar unas proyecciones, y al realizar su join, los atributos comunes que realizan la operación (atributos de join) están formados por claves (primaria o secundarias) de la tabla.

Si las posibles proyecciones de una tabla no pueden llevar implicadas las claves, entonces la tabla no se encuentra en 5FN, y debe por tanto proyectarse para evitar los problemas de las anomalías de actualización.

Por ej. Las operaciones que realizan en los cajeros diversos clientes.

CAJOPE= (Cajero,Operación, Cliente)

001, Ingreso, Cliente 1
001, Reintegro, Cliente 1
002, Reintegro, Cliente 1
001, Ingreso, Cliente 2
001, Ingreso, Cliente 3
001, Reintegro, Cliente 3
002, Reintegro, Cliente 3
003, Reintegro, Cliente 3

Se encuentra en 4FN.

La tabla original, CAJOPE, tendrá multitud de registros que son costosos de tratar cuando se realizan diversas consultas sobre la misma tabla, por ej., para conocer qué operaciones se han realizado en una cajero tendrá que tratar todas aquellas filas en las que los clientes han realizado operaciones en ese cajero. Evidentemente, si pasamos la tabla a 5FN, los datos se distribuirán en tres tablas, complicándose el programa de alta o modificación, sin embargo, las consultas se distribuyen en tablas diferentes y tratan única y exclusivamente las filas afectadas. Por ej., para conocer las operaciones que ha realizado un cliente, no se repetiría para cada cajero. Además frente a una transacción que se realiza de alta, cuando la aplicación ya está disponible, se ejecutan generalmente cientos de consultas y éstas son las que hay que optimizar. Si no se desean realizar accesos on-line para dichas consultas, no sería necesario pasarlo a 5FN proyectando la tabla original, y se puede evaluar dejarla en 4FN.

Para pasarla a 5FN creamos proyecciones de las claves de manera que exista una DJ de las proyecciones generadas con la tabla CAJOPE, resultando:

CAJOPE(Cajero, Operación, Cliente) DJ (T1(Cajero, Operación), T2(Cajero, Cliente), T3(Operación, Cliente))

La 5FN también se aplica a tablas muy grandes que son intratables desde transacciones y que mantiene varias entidades, todas relacionadas entre sí de la forma 1:1. Estas tablas están en 5FN, pero se aplica la DJ para proyectarlas. Las proyecciones

que se pueden formar deben tener como atributo de join una clave de la tabla (primaria o secundaria), para que cunado se realice la operación de join siempre se cumpla que una tupla se relaciona con una única tupla de otra proyección.
Por ej. datos de los empleados de una empresa: EMPLEADOS
Cod_trabajador, nif, nombre, dirección,...,nº S/S, fecha alta en S.S.,..,Categoría,puesto..

Esta tabla se encuentra en 5FN, pero para optimizar el diseño, nos interesa crear varias proyecciones de manera que se maneje por cada fila un número menor de atributos, sólo los necesarios.

EMPLEADOS(Cod_trabajador,...,puesto....)
DJ ((DATOS_PERSONALES(Cod_trabajador, nombre, dirección,..),
DATOS_SS(Cod_trabajador, nºSS, fecha,...),
DATOS_LABORALES(Cod_trabajador, categoría, puesto,...)).

Se realizarían por tanto 3 proyecciones. El atributo de join será el Cod_trabajador, clave primaria de la original.

Campos Redundantes

Por último hay que tener en cuenta los inconvenientes de la existencia de campos redundantes, es decir, aquellos campos cuyo resultado se basa en los cálculos efectuados con otros atributos o cuya información puede obtenerse a partir de los atributos existentes, ya que conlleva los problemas de actualización (MIB) derivados de la redundancia de información.

Por ej. ARTICULOS (Cod_artículo,....,Precio Unitario, Unidades en almacen, ,Total pesetas en almacen de ese artículo, total acumulado de ventas de ese artículo)
En esta tabla encontramos un campo calculado: Total pesetas en almacen que se obtiene de la operación: Precio_unitario*Unidades en almacen. Este campo deberá ser actualizado cada vez que modifiquemos cualquiera de los campos que intervienen.
Igualmente ocurre con el total acumulado de ventas, que se obtiene a partir de las ventas realizadas reflejadas en las facturas. Estamos guardando información redundante, y mantener la integridad de los datos conlleva operaciones extras, cuando esta información podría obtenerse a partir de los atributos existentes. En este caso, estamos hablando de una redundancia perjudicial, ya que es más costoso, mantener esos campos que recalcularlos.

Sin embargo, y depediendo de la explotación que se vaya a realizar de la información y del número de datos que manejemos, pueden existir casos en los que la redundancia esté justificada. Por ej. CLIENTE(Cod_cliente,..,total compras)
Si se van a realizar consultas frecuentes sobre el importe total de compras de los clientes, y el número de facturas con el que trabajamos es grande, se puede justificar así, la existencia de este campo redundante, a pesar de las operaciones extra que hemos de

realizar en procesos de MIB (Modificación, Inserción y Borrado) para mantenerlo actualizado.

Por tanto, antes de introducir un campo redundante, hemos de sopesar las ventajas e inconvenientes, para un sistema de información concreto, y en caso de introducirlo, hemos de justificarlo por escrito, indicando siempre a qué operaciones de qué tablas afecta.

- Realizado por Nuria Rivera -