Normalización

1. Introducción

Nuestro departamento de informática ha recibido el encargo de diseñar una base de datos para llevar el control de las piezas, proveedores y proyectos que realiza nuestra empresa. Concretamente se necesita mantener los siguientes datos:

De los proveedores: nombre, ciudad, país, domicilio.(Consideraremos que conociendo la ciudad ya sabemos a qué país corresponde).
De los proyectos: nombre del proyecto, ciudad donde se realiza, duración estimada.
De las piezas: nombre de las piezas utilizadas, color.
Finalmente se necesita poder conocer qué proveedores han sido asignados a qué proyectos, así como las piezas que ha suministrado cada proveedor y en que cantidad.

A nuestro analista se le ha ocurrido la idea de diseñar la base de datos según inspiración divina (algo que ocurre con más frecuencia de lo debiera).

El diseño que hemos recibido está compuesto de estas dos relaciones:

SP(S#,P#,nompro,nompie,ciudad,pais,color,domicilio,cantidad)
SJP(S#,J#,P#,nomproy,duración,ciudad)

En la práctica: Antes de normalizar deberíamos verificar que las tablas que nos han dado cumplen los requisitos que se nos piden, es decir, representan correctamente lo que se quiere implementar, y si no es así, rediseñar lo que nos dan según nuestro criterio. Aquí es donde nos ayudará la experiencia, considerando que una relación constituye una buena representación del problema si:

1. Incluye toda la información que hay que representar.
2. Se cumple la Mínima redundancia:
   - Ahorro de espacio.
   - Evitar incoherencias.
3. Resulta Eficiente.

Si lo hacemos bien seguramente nos ahorraremos mucho trabajo al pasar a la fase de normalización. Sin embargo, como lo que nos interesa es trabajar la normalización empezaremos a analizar las formas normales de las relaciones proporcionadas, directamente, sin plantearnos si su diseño es correcto o no.

1.1. La Primera Forma Normal (1FN).

Teoría

R está en 1FN si y solo si todos los dominios simples subyacentes contienen sólo valores atómicos.

Lo que quiere decir: "Todos los campos deben ser atómicos". En las relaciones que nos han proporcionado se cumple. Quizás después de todo nuestro analista no sea tan malo, claro que con los SGBD actuales los dominios son simples por definición. Esta propiedad en la práctica se da siempre.

1.2. La Segunda Forma Normal (2FN).

Teoría

R está en 2FN si y sólo si está en 1FN y todos los atributos no clave dependen funcionalmente de manera completa de la clave primaria.

(NOTA: cuando se dice un "*atributo no clave*" estamos indicando tanto que este atributo no forma parte de una clave candidata o no es en sí mismo clave candidata, concepto equivalente a *no primo* o *no principal*)

En la práctica:

Si la clave es compuesta (más de un atributo), surge el problema. Tenemos que comprobar que no haya dependencias entre los atributos que componen la clave primaria y el resto de atributos que no son clave.

En nuestro caso, podemos tener problemas. De todos modos lo primero será identificar las claves primarias de cada relación (si no lo habíamos hecho ya). En este caso tenemos:

En SP tomamos la clave (S#,P#)
En SJP la clave es (J#,P#,S#)

Como vemos, todas las claves son compuestas. Empecemos con la primera relación, SP.

SP(S#,P#,nompro,nompie,ciudad,pais,color,domicilio,cantidad)

Dibujamos el diagrama de las dependencias funcionales respecto a la clave primaria existente (de momento omitimos cualquier otra dependencia que no sea la de clave primaria para no confundirnos, tenemos que tener claro que estamos comprobando la 2FN).

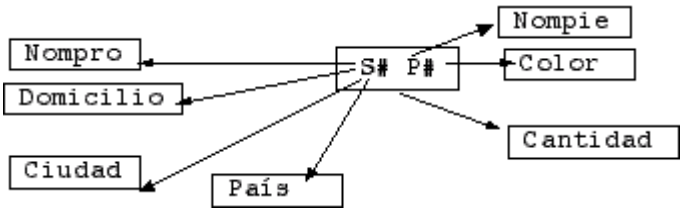

Fig.1. Diagrama de dependencias funcionales

Problemas de no cumplir 2FN:

Los inconvenientes que presenta una relación así son principalmente:

- En la INSERCIÓN no puede reflejarse el hecho de que un proveedor vive en una determinada ciudad hasta que no suministre una pieza.

- En el BORRADO, si tenemos que borrar los datos de un proveedor deberemos borrar todos los datos de las piezas que suministra, con lo que perderemos información sobre las piezas.

- Finalmente los atributos que dependen funcionalmente del proveedor aparecerán tantas veces como piezas haya suministrado. Del mismo modo, todos los atributos que dependen de la pieza aparecerán tantas veces como proveedores haya que la vendan. Esto se traduce en un coste muy alto en el caso de que realicemos una ACTUALIZACIÓN ya que deberemos mantener la consistencia para todas las tuplas, así si cambia el nombre del proveedor S1 se debe propagar este cambio para todas las piezas que haya vendido, si cambiara el color de la pieza P4, se debería cambiar allí donde aparezca (que será para todos los proveedores que la suministran).

Y en este caso tenemos un auténtico desastre, y es que sólo hay un atributo no clave que dependa funcionalmente de manera completa de la clave, la cantidad. ¿Cómo lo arreglamos?

La Solución:

Teoría

Teorema.

Sea R una relación con atributos (A,B,C,D) y con clave primaria (A,B) tal que A --> D, entonces la relación R puede descomponerse como:

R --> R1(A,D) y R2(A,B,C)

En la práctica:

Sacamos de la relación los atributos no clave que dependan funcionalmente de un componente de la clave y los metemos en una nueva relación, esta relación tendrá como clave primaria el componente de la clave del que depende el resto.

En nuestro caso (tabla SP) veremos que tenemos que aplicar el método dos veces. Pero vayamos poco a poco. En primer lugar tenemos que la clave primaria es (S#,P#) y que existen atributos que dependen funcionalmente de S#. Tomamos S# y todos esos atributos y los metemos en una nueva relación, a la que, para ser sorprendentemente originales, le pondremos el nombre S. Tendremos por tanto la siguiente relación:

S(S#,nompro,domicilio,ciudad,pais)

Y la relación original quedaría como:

SP(S#,P#,color,cantidad,nompie)

Sin embargo, esta última relación sigue sin estar en 2FN, mirando el diagrama observamos que color y nompie dependen funcionalmente de P#, un componente de la clave (S#,P#). Volvemos a aplicar el teorema para SP, en este caso para con el atributo P# y todos los atributos que dependen funcionalmente de él (color y nompie).

El resultado final es:

S(S#,nompro,domicilio,ciudad,pais)
P(P#,color,nompie)
SP(S#,P#,cantidad)

Ahora nos queda la tabla SPJ, directamente el resultado final es:

SPJ(S#,J#,P#)
J(J#,nomproy,duración,ciudad)

Ejercicio: Desarrolla todo el proceso no descrito para la tabla SJP.

1.3. La Tercera Forma Normal (3FN).

Teoría

R está en 3FN si y sólo si está en 2FN y todos los atributos no clave dependen de manera no transitiva de la clave primaria.

En la práctica:

Tenemos que conseguir que NO haya dependencias funcionales de la forma ClavePrimaria --> B -->...--> C. NO hay que permitir que se pueda llegar desde la clave primaria hasta un atributo C siguiendo las dependencias funcionales entre atributos intermedios. Esta son las relaciones que tenemos hasta ahora:

S(S#,nompro,domicilio,ciudad,pais)
P(P#,color,nompie)
SP(S#,P#,cantidad)
SPJ(S#,J#,P#)
J(J#,nomproy,duración,ciudad)

Lo que hacemos es buscar dependencias funcionales entre los atributos que no son clave primaria de una misma relación. Sólo tenemos esa característica en la tabla S, escribimos el diagrama de dependencias funcionales sólo para los atributos implicados:

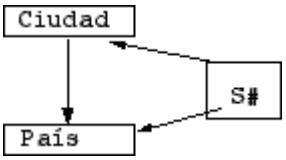

Fig.2. Dependencias transitivas existentes.

Existe por tanto una dependencia transitiva, ya que si seguimos la flecha que parte de S# hasta ciudad podemos seguir hasta País.

Problema:

Las dependencias transitivas producen una serie de anomalías que afectan al rendimiento de la base de datos, éstas son:

- No podemos realizar la INSERCIóN de una ciudad y el país que le corresponde hasta que exista un proveedor en dicha ciudad.

- El BORRADO de la única ocurrencia de una ciudad provoca que borremos, por un lado la ciudad a la que pertenece S1 y por otro el país al que pertenece la ciudad. Si lo analizamos con cuidado estamos borrando dos informaciones semánticamente diferentes, ya que el hecho de que un proveedor deje de existir no significa que la ciudad en la que reside deje también de pertenecer a un país.

- También tenemos redundancias, en este caso el país se repite tantas veces como proveedores haya residiendo en ciudades pertenecientes a él. Como siempre las redundancias penalizan las operaciones de ACTUALIZACIóN.

La Solución:

Teoría:

Teorema de Heath.

Sea R una relación con atributos (A,B,C) y con clave primaria (A) tal que B --> C, entonces la relación R puede descomponerse como:

R --> R1(A,B) y R2(B,C)

En la práctica:
Tal y como dice el teorema, si tenemos ClavePrimaria --> B --> C, creamos una nueva relación R1(B,C) y dejamos en la relación anterior R(ClavePrimaria,B). En nuestro caso tenemos S# --> ciudad --> país. Con lo que finalmente podemos dividir la relación en las siguientes tablas:

S(S#,nompro,domicilio,ciudad)
CP(ciudad,país) (relación ciudad-país).
o bien en...
S(S#,nompro,domicilio,ciudad)
PS(S#,país).

Ambas descomposiciones están en 3FN, pero ¿cuál es la mejor opción?.

En el primer caso podemos introducir información sobre las ciudades sin tener que conocer información alguna sobre el proveedor. Ceuta es una ciudad española independientemente de que el proveedor S1 resida en ella o no. En el segundo caso no podemos introducir información del país sin conocer antes el proveedor, es más, si cambia un proveedor la actualización será más dificil. En general las operaciones son más complejas según la última descomposición.

El motivo de todo esto tiene un nombre y es INDEPENDENCIA. En el primer caso ambas relaciones son indipendientes la actualización en una no afecta en otra y viceversa. En general esta propiedad es más que desable por eso si tenermos varias opciones nos decantaremos por aquellas descomposiciones que sean independientes, sin embargo hay que hacer constar que muchas veces nos encontraremos con relaciones que no pueden ser descompuestas en otras que no sean dependientes. Existe un teorema que nos servirá para comprobar si dos relaciones, R1 y R2, resultantes de haber descompuesto R son independientes.

Teoría:

Teorema de Rissanen.

La descomposición en R1 y R2 de R es independiente si y sólo si:

1. **Toda dependencia funcional en R se puede deducir lógicamente de las dependencias funcionales en R1 y R2.**

2. **Los atributos comunes de R1 y R2 componen una clave candidata de al menos una de las relaciones.**

En la práctica: Ajustándonos a los teoremas vistos, nuestra base de datos tiene ahora el siguiente aspecto:

S(S#,nompro,domicilio,ciudad)
CP(ciudad,país) (relación ciudad-país).
P(P#,color,nompie)
SP(S#,P#,cantidad)
SPJ(S#,J#,P#)
J(J#,nomproy,duración,ciudad)

Ejercicio:
Verificar que efectivamente las relaciones obtenidas están en 3FN y especificar porqué.
Desarrolla algunas relaciones que NO estén en 3FN, y justifica porqué.

1.4. La Forma Normal de Boyce-Codd (FNBC).

Teoría

Una relación R está en FNBC si y solo si todo determinante es una clave candidata.

def. Determinante. Un determinante es un atributo del que depende funcionalmente de manera completa algún otro atributo.

En la práctica:

La mayoría de las relaciones en 3FN estarán en FNBC. Para verificar que esto es así buscaremos todos los determinantes existentes en la relación así como todas las claves candidatas, comparamos ambos conjuntos y si encontramos que hay algún determinante que no resulta ser clave candidata podemos afirmar que no estamos en FNBC. Usualmente se considera aceptable tener relaciones que lleguen sólo hasta la FNBC.

El conjunto de relaciones obtenido hasta ahora está en 3FN y también en FNBC, de hecho es lo que suele ocurrir en la mayoría de los casos. Sin embargo, para ver una resolución práctica, consideraremos el caso de la tabla SP inicial con ligeras modificaciones:

Dada la relación SP(S#,P#,nompro,cantidad), donde el nombre del proveedor es único, es decir, no se repite, y además no puede contener valores nulos. Esto significa que el par (nompro,P#) es clave candidata, ya que conociendo su valor podemos determinar el resto de atributos de la relación.

En primer lugar realizamos el diagrama de dependencias correspondiente:

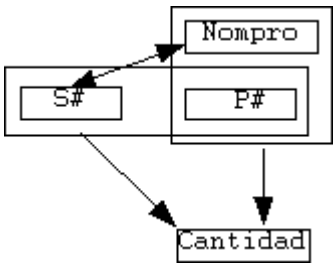

Fig.3. Diagrama de dependencias funcionales para la FNBC.

Comprobamos que la relación está en 1FN, todos los atributos son atómicos. También está en 2FN ya que no hay dependencias funcionalmente completas entre atributos que no sean clave (formen parte de la clave). Y finalmente se verifica que no hay ningún atributo que dependa de forma transitiva de la clave Primaria, luego está en 3FN.

Ahora queda comprobar si está en FNBC, según la definición habrá que verificar si todos los determinantes son clave candidata, vemos que no es así, S# y P# son determinantes y no son clave candidata, luego no está en FNBC.

Problema:

La definición de la 3FN no produce diseños satisfactorios cuando se dan las siguientes condiciones, o lo que es lo mismo, cuando una relación NO ESTE EN FNBC concurrirán las siguientes circunstancias:

- Existen varias claves candidatas.
- Las claves candidatas son compuestas.
- Las claves candidatas se solopan, tienen al menos un atributo en común.

La Solución:

En la práctica:

No hay un teorema sobre la división de la relación, el motivo es que no se puede asegurar que al descomponer una relación en dos para conseguir la FNBC el significado de las relaciones obtenidas se corresponda semánticamente a lo que representa la relación inicial. En otras palabras, podemos equivocarnos al descomponer ya que puede que perdamos parte de la semántica de la relación anterior.

En el caso que nos ocupa podemos perfectamente dividir la relación en dos sin perder significado alguno.

El resultado sería:

S(S#,nombre)
SP(S#,P#,cantidad)

o bien esta otra descomposición:

S(S#,nombre)
SP(nombre,P#,cantidad)

Ambas están en FNBC.

Como posible criterio para seleccionar una u otra, en la práctica podemos escoger aquella distribución que optimice el espacio ocupado. Supongamos que en nuestro caso el atributo nombre tiene hasta 50 caracteres (no variables, fijos) mientras que S# es sólo un entero, parecería más razonable tomar la primera descomposición como solución ya que las tuplas de la tabla SP son más pequeñas que las de la misma tabla en la otra descomposición.

En situaciones normales podíamos darnos por satisfechos con haber llegado hasta esta descomposición.

Ejercicio:

Atrévete a diseñar algunas relaciones que incumplan la Forma Normal de Boyce-Codd pero que sí estén en 3 FN. (Recuerda las características que tiene una relación en FNBC).