Optimización de Descomposiciones de Esquemas Normalizados en el Modelo Relacional

Marcelo A. Falappa

Instituto de Investigación en Ciencia y Tecnología Informática (IICyTI)
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
Av. Alem 1253 - (B8000CPB) Bahía Blanca - Argentina
PHONE/FAX: (+54)(291)459-5136
E-MAIL: mfalappa@cs.uns.edu.ar

Palabras Clave: Bases de Datos, Modelo Relacional, Formas Normales, Descomposiciones de Esquemas, 3FN, FNBC.

Resumen

En este trabajo se presenta una optimización a los algoritmos de descomposición para obtener tercera forma normal y forma normal de Boyce-Codd [Date, 2001, Lewis et al., 2002, Maier, 1983, Silberschatz et al., 1999, Ullman, 1989]. Esta optimización consiste en unir aquellos diferentes subesquemas obtenidos por los algoritmos de descomposición tradicionales, obteniendo menos subesquemas, y por ende, probablemente menos reiteración de atributos. Esta optimización tiene la característica de preservar las propiedades de preservación de dependencias (en 3FN) y join sin pérdida (lossless join) a un bajo costo computacional ya que debe tomar los diferentes subesquemas compuestos por unos pocos atributos.

1. Introducción

En este artículo presentaremos una optimización a los algoritmos de normalización conocidos. Asumiremos que el lector está familiarizado con el modelo relacional y el diseño de esquemas de bases de datos. Para ello, adoptaremos ciertas convenciones en el uso de símbolos. En general, R representará un esquema de relación y $r, r_1, \ldots, r_n$ relaciones definidas sobre dicho esquema. Las primeras letras mayúsculas representarán atributos simples: $A, B, C, \ldots$ mientras que las últimas letras mayúsculas representarán conjuntos de atributos: $U, V, \ldots, Z$. La concatenación se usa como abreviatura de la unión: XY representa $X \cup Y$ mientras que XA representa $X \cup \{A\}$. De manera similar, $X - A$ representará de manera alternativa la diferencia $X \setminus \{A\}$.

El artículo está organizado como sigue: en la Sección 2 se presentan las dependencias funcionales y en la Sección 3 los cubrimientos mínimos. En la Sección 4 se definen descomposiciones y sus propiedades así como las distintas formas normales. Finalmente, en la Sección 5 se presentan las optimizaciones propuestas así como también una serie de ejemplos.

2. Dependencias Funcionales

En el *modelo relacional* [Maier, 1983, Silberschatz et al., 1999, Ullman, 1989] se define un conjunto de atributos que constituyen el esquema de relación o base de datos. En este modelo, una base de datos puede verse como una tabla o conjunto de tablas donde cada columna representa un atributo de un conjunto de entidades y cada fila una tupla, correspondiente a una entidad en el modelo entidad-relación. Los valores de los atributos están condicionados a situaciones del mundo real que se está modelando [Ullman, 1989]:

- **Restricciones Semánticas**: estas restricciones dependen del significado de los atributos. Por ejemplo, si un atributo representa la calificación de un examen, su valor estará restringido a valores entre 0 y 10; si un atributo representa la edad de un empleado activo, su valor estará restringido a valores enteros entre 16 y 65.
- **Restricciones Funcionales**: estas restricciones dependen de la igualdad o desigualdad de valores. Por ejemplo, si ciertas tuplas coinciden en determinados atributos, entonces deben coincidir en otros. Por ejemplo, si en una tabla tenemos almacenados las materias que cursan alumnos de una carrera, podremos tener que un mismo alumno cursa distintas materias, por ejemplo:

Cod-Alumno	*Nombre-Alumno*	*Cod-Materia*	*Nombre-Materia*
29883	Julio Naveira	5954	Bases de Datos
29883	Julio Naveira	5949	Sistemas Operativos
31652	Vanina Lubiz	5954	Bases de Datos
31652	Vanina Lubiz	5523	Algoritmos y Complejidad

Cuadro 1: Un ejemplo de una tabla.

Es claro que en este ejemplo, si dos tuplas coinciden en el código de alumno, deben coincidir en el nombre del mismo. Análogamente, si dos tuplas coinciden en el código de materia, deben coincidir en el nombre de la misma. Este tipo de restricciones puede establecerse mediante dependencias funcionales.

Las *dependencias funcionales* (df's) son ampliamente utilizadas en el diseño de bases de datos como restricciones de integridad. Sea $R(A_1, \ldots, A_n)$ un esquema de relación sobre los atributos $A_1, \ldots, A_n$ y sean X e Y subconjuntos de $\{A_1, \ldots, A_n\}$. Se dice que X determina funcionalmente a Y, notado como $X \rightarrow Y$, si en cualquier relación r sobre el esquema R no existen dos tuplas que coinciden en todos los atributos de X pero no

coinciden en todos los atributos de Y. En el ejemplo anterior, tenemos las dependencias funcionales *Cod-Alumno* $\rightarrow$ *Nom-Alumno* y *Cod-Materia* $\rightarrow$ *Nom-Materia*.

Generalmente, un esquema de relación $R(A_1, \ldots, A_n)$ representa un conjunto de entidades con atributos $A_1, \ldots, A_n$ y una relación r definida sobre R puede verse como un conjunto de tuplas que representan entidades. Si un esquema R satisface $X \rightarrow Y$ entonces dos entidades que coinciden en X deben coincidir en Y.

Existen una serie de axiomas que permiten deducir nuevas dependencias funcionales a partir de la definidas en el problema modelado. Estos axiomas se conocen como *Axiomas de Armstrong* [Ullman, 1989]:

- Reflexividad: si $Y \subseteq X$ entonces $X \rightarrow Y$.
- Aumento: si $X \rightarrow Y$ entonces $XZ \rightarrow YZ$.
- Transitividad: si $X \rightarrow Y$ y $Y \rightarrow Z$ entonces $X \rightarrow Z$.

Por ejemplo, dada la dependencia funcional $AB \rightarrow C$ pueden deducirse, entre otras, las siguientes dependencias funcionales: $AB \rightarrow B$, $ABCD \rightarrow CD$, $ABC \rightarrow ABC$, $ABCDE \rightarrow E$. Algunas de estas deducciones puede implicar el uso de más de uno de los axiomas.

Los axiomas de Armstrong son sensatos (sound) y completos, esto es, permiten derivar solamente dependencias funcionales válidas, y todas las dependencias funcionales válidas pueden derivarse a partir de ellos. En general, dado un conjunto de dependencias funcionales F se asocia un conjunto F^+ que representa la clausura de F, esto es, todas aquellas dependencias funcionales que pueden deducirse a partir de F usando los axiomas de Armstrong. Luego, se define la *derivación* de dependencias funcionales de la siguiente manera: $X \rightarrow Y \in F^+$ si y solo si $F \vdash X \rightarrow Y$.

Cuando se habla de un conjunto de entidades, asumimos que existe un conjunto de atributos que constituye una *llave*, esto es, un conjunto de atributos que determina unívocamente a determinada entidad en el conjunto de entidades. Esto puede extenderse para relaciones con dependencias funcionales. Sea R un esquema de relación con atributos $A_1, \ldots, A_n$ y dependencias funcionales F. Si $X \subseteq \{A_1, \ldots, A_n\}$ entonces X es *llave* de R si $X \rightarrow A_1 \ldots A_n$ puede deducirse de F y, para todo $Y \subset X$, no vale que $Y \rightarrow A_1 \ldots A_n$. Esto es, X es llave si determina todos los atributos de R y es mínimo (en cantidad de atributos). Si X es llave de R entonces para todo Z tal que $X \subseteq Z \subseteq R$ se dice que Z es *superllave* de R.

3. Cubrimientos Mínimos

Dado un conjunto de dependencias funcionales F, se dice que G es un *cubrimiento* si $F^+ = G^+$, esto es, las clausuras de ambos conjuntos es la misma. Esto es, para ver si dos conjuntos de dependencias funcionales F y G son *equivalentes* (notado como $F \equiv G$) se debe verificar que sus clausuras sean iguales.

Sin embargo, el cómputo de la clausura de un conjunto de dependencias funcionales es un proceso costoso y, a los fines prácticos, a veces excesivo ya que la clausura contiene

muchas dependencias triviales. Por ejemplo, dado $F = \{A \rightarrow B\}$ sobre $R(AB)$ su clausura es:

$$F^+ = \{A \rightarrow A, B \rightarrow B, A \rightarrow B, AB \rightarrow A, AB \rightarrow B, AB \rightarrow AB\}$$

Uno puede ver que la clausura contiene muchas dependencias irrelevantes y es por eso que presentaremos un algoritmo que permite computar cubrimientos sin necesidad de computar la clausura de conjuntos de dependencias funcionales [Falappa, 2003].

En lugar de definir la clausura de un conjunto de dependencias funcionales definiremos la *clausura de un conjunto de atributos*. Esto es, dado un conjunto de atributos X se definirá su clausura con respecto a un conjunto de dependencias funcionales F mediante el siguiente algoritmo:

Algoritmo ClausurarAtributos
Datos de Entrada: un conjunto de atributos X y un conjunto de df's F
Datos de Salida: un conjunto de atributos clausurados X_F^+
Comienzo del Algoritmo
    $X_F^+ := X$
    Para cada dependencia funcional $U \rightarrow V$ en F **hacer**:
        Si $U \subseteq X_F^+$
        Entonces
            $X_F^+ := X_F^+ \cup V$
Fin del Algoritmo

Luego, las siguientes condiciones son equivalentes:

1. $X \rightarrow Y \in F^+$.
2. $F \vdash X \rightarrow Y$.
3. $Y \subseteq X_F^+$.

Para no sobrecargar la notación, notaremos X^+ en lugar de X_F^+. A partir de esto, definiremos un *cubrimiento cerrado en atributos*. Esto es, dado un conjunto de dependencias funcionales F, su cubrimiento cerrado en atributos es el conjunto G tal que:

$$G = \{X \rightarrow X^+ : X \rightarrow Y \in F\}$$

Por ejemplo, dado el conjunto de df's $F = \{A \rightarrow B, B \rightarrow C\}$ su cubrimiento cerrado en atributos es:

$$G = \{A \rightarrow ABC, B \rightarrow BC\}$$

A partir de esto, puede obtenerse un cubrimiento mínimo (en canditad de dependencias funcionales) eliminando dependencias redundantes. Una df $X \rightarrow Y$ es *redundante* en F si $F \setminus \{X \rightarrow Y\} \vdash X \rightarrow Y$. Esto es, $X \rightarrow Y$ es redundante en F si quitando dicha dependencia de F, la misma puede deducirse a partir de las restantes dependencias funcionales.

Una df $X \to Y$ contiene *atributos extraños a izquierda* en F si $X = AZ, X \neq Z$, y $F \setminus \{X \to Y\} \cup \{Z \to Y\} \equiv F$. Una df $X \to Y$ está *reducida a izquierda* en F si no contiene atributos extraños a izquierda. Alternativamente, una df $X \to Y$ contiene *atributos extraños a derecha* en F si $Y = BW, Y \neq W$, y $F \setminus \{X \to Y\} \cup \{X \to W\} \equiv F$. Una df $X \to Y$ está *reducida a derecha* en F si no contiene atributos extraños a derecha.

Luego, podemos obtener un *cubrimiento mínimo reducido* de un conjunto de dependencias funcionales mediante el siguiente algoritmo.

Algoritmo CubrimientoMínimoReducido
Datos de Entrada: un conjunto de df's F
Datos de Salida: un cubrimiento mínimo reducido G
Comienzo del Algoritmo
Sea G_1 el cubrimiento cerrado en atributos de F.
G_2 se obtiene eliminando dependencias redundantes de G_1.
G_3 se obtiene reduciendo a izquierda cada df en G_2.
G se obtiene reduciendo a derecha cada df en G_3.
Fin del Algoritmo

Puede demostrarse que este algoritmo es correcto en el sentido de que $G \equiv F$ y, a su vez, la cantidad de dependencias funcionales de G es la mínima posible.

Por ejemplo, consideremos el siguiente conjunto de df's F:

$$\{ABD \to E, AB \to L, B \to K, C \to J, J \to ABC, CJ \to I\}$$

A partir de F, se calcula su cubrimiento clausurado en atributos:

$$\{ABD \to ABDEKL, AB \to ABKL, B \to BK, C \to ABCKLIJ, J \to ABCKLIJ, CJ \to ABCKLIJ\}$$

Luego, se eliminan las dependencias redundantes:

$$\{ABD \to ABDEKL, AB \to ABKL, B \to BK, C \to ABCKLIJ, J \to ABCKLIJ\}$$

Luego, se eliminan atributos extraños a izquierda aunque en este caso particular no existen dichos atributos. Finalmente, se eliminan atributos extraños a derecha:

$$\{ABD \to E, AB \to L, B \to K, C \to ABIJ, J \to C\}$$

Veremos más adelante que para alcanzar determinadas formas normales es necesario *abrir a derecha* las dependencias funcionales. En este caso, la df $C \to ABIJ$ puede descomponerse en 4 df's: $C \to A$, $C \to B$, $C \to I$, $C \to J$.

4. Descomposiciones y Formas Normales

4.1. Descomposiciones

Una descomposición de un esquema de relación $R(A_1 \dots A_n)$ es una colección de conjuntos $\rho = (R_1, R_2, \dots, R_m)$ tal que $R = R_1 \cup R_2 \cup \dots \cup R_m$. Los R_i's no necesitan ser

disjuntos y, en general, estas descomposiciones buscan evitar redundancias, y anomalías de inserción y borrado en una bases de datos. Si consideramos la relación de la Tabla 1 vemos que se podría descomponer el esquema para no reiterar el nombre de cada alumno y el nombre de cada materia. Una posible descomposición sería:

Cod-Alumno	*Nombre-Alumno*
29883	Julio Naveira
31652	Vanina Lubiz

Cod-Materia	*Nombre-Materia*
5954	Bases de Datos
5949	Sistemas Operativos
5523	Algoritmos y Complejidad

Cod-Alumno	*Cod-Materia*
29883	5954
29883	5949
31652	5954
31652	5523

Cuadro 2: Descomposición en 3 relaciones.

Si una relación r sobre un esquema R se descompone en $R_1, R_2, \ldots, R_n$, deberían satisfacerse dos propiedades:

- **Join sin Pérdida (lossless join)**: la relación r es exactamente igual al join natural [Maier, 1983, Silberschatz et al., 1999] de sus proyecciones en los distintos R_i's.
- **Preservación de Dependencias**: la unión de las dependencias que se proyectan sobre los distintos R_i's permiten deducir las dependencias planteadas originalmente a partir de la semántica del problema modelado.

Estas dos propiedades son independientes entre si y es deseable que se satisfagan. La primera para asegurar que se preserva la calidad de la información almacenada, y la segunda para seguir respetando las restricciones de integridad planteadas en el momento de diseño.

4.2. Formas Normales

Las formas normales son restricciones definidas sobre un esquema de bases de datos que presumiblemente, previenen redundancias y ciertas anomalías en las operaciones de actualización [Lewis et al., 2002]. Existen básicamente las siguientes formas normales: *Primera Forma Normal* (1FN), *Segunda Forma Normal* (2FN), *Tercera Forma Normal* (3FN), *Forma Normal de Boyce-Codd* (FNBC).

La 1FN exige que los dominios de los atributos de una relación sean atómicos (esto es, no sean listas, conjuntos o estructuras de tamaño variable) con el objetivo de alcanzar una representación tabular y lenguajes de consulta más simples. Este es el requerimiento básico del que se parte cuando se utiliza el modelo relacional.

La 2FN exige que no existan *dependencias parciales*. Esto es, todas las dependencias funcionales deben estar reducidas a izquierda. Si se define un conjunto de df's F y se obtiene un cubrimiento mínimo reducido se garantiza que se alcance la 2FN.

La 3FN y FNBC son las más interesantes y las que presentaremos con algo más de profundidad. Dado un esquema de relación R, un subconjunto X de R, un atributo A en R y un conjunto de df's F, se dice que A es *transitivamente dependiente* de X si existe un subconjunto Y de R tal que $X \rightarrow Y$, no es el caso en que $Y \rightarrow X$, $Y \rightarrow A$ está en F pero A no está en XY. Una atributo A es *primo* en R con respecto a F si A es parte de alguna llave de R; de lo contrario, se dice que A es *no primo*.

Supongamos que F es un conjunto de df's mínimo reducido donde sus dependencias fueron abiertas a derecha. Decimos que un esquema R con df's F *está en 3FN* si para cada dependencia $X \rightarrow A$ en F se verifica que X es llave o bien A es un atributo primo. Esta forma normal tiene como objetivo eliminar dependencias transitivas de atributos no primos. Por otra parte, decimos que un esquema R con df's F *está en FNBC* si para cada dependencia $X \rightarrow A$ en F se verifica que X es llave. Podemos ver que la FNBC es más estricta que la 3FN ya que elimina todo tipo de dependencias transitivas. En 3FN y FNBC es necesario tener presentes algoritmos para obtener las llaves de un esquema de relación. Dichos procedimientos pueden hallarse en [Falappa, 2003, Ullman, 1989] pero no serán abordados en el presente artículo.

Cuando un esquema no satisface determinada forma normal, se lo descompone tratando de alcanzar la forma más alta posible. Es deseable que en esta descomposición se satisfagan las dos propiedades mencionadas anteriormente: join sin pérdida y preservación de dependencias. Una descomposición $\rho = (R_1, \ldots, R_n)$ se dice que está en una forma normal FN (hablamos genéricamente, FN puede ser 1FN, 2FN, 3FN o FNBC) si cada esquema R_i está en la forma normal FN. Por lo tanto, cuando determinado esquema no satisface una forma normal, se buscará alcanzar la misma aplicando diferentes algoritmos de descomposición.

La 3FN es fácil de alcanzar partiendo de un cubrimiendo F que sea mínimo reducido y cuyas dependencias sean abiertos a derecha. El algoritmo consiste básicamente en tomar una descomposición ρ compuesta por un esquema XA por cada dependencia funcional $X \rightarrow A$ en F. Obviamente, puede haber menos esquemas que dependencias ya que puede darse el caso en que dos dependencias estén contenidas en el mismo esquema.

Suponiendo que partimos del siguiente conjunto de dependencias funcionales (que es un cubrimiento mínimo reducido) definidas sobre $R(ABCDEKLIJ)$:

$$\{ABD \rightarrow E, AB \rightarrow L, B \rightarrow K, C \rightarrow ABIJ, J \rightarrow C\}$$

Si abrimos a derecha las dependencias funcionales obtenemos como descomposición a $\rho_1 = (ABDE, ABL, BK, CA, CB, CI, CJ)$ sobre las cuales se proyectan, respectivamente, las df's $\{ABD \rightarrow E\}$, $\{AB \rightarrow L\}$, $\{B \rightarrow K\}$, $\{C \rightarrow A\}$, $\{C \rightarrow B\}$, $\{C \rightarrow I\}$, $\{C \rightarrow J, J \rightarrow C\}$.

Con este algoritmo, se obtiene una *descomposición en 3FN que claramente preserva dependencias*. Para garantizar que la misma sea join sin pérdida, es necesario verificar que algunos de los esquemas contenga una llave. De no ser ese el caso, es

necesario agregar un esquema con una llave. En el caso anterior, las llaves son CD y DJ, no contenidas en ninguno de los esquemas. Por lo tanto, la descomposición $\rho_2 = (ABDE, ABL, BK, CA, CB, CI, CJ, CD)$ es una descomposición en 3FN que preserva dependencias y es join sin pérdida.

La FNBC es la más restrictiva de las formas normales aunque no siempre puede alcanzarse una descomposición que sea join sin pérdida y además preserve dependencias. El siguiente algoritmo, extraído de [Ullman, 1989], permite obtener una descomposición join sin pérdida partiendo de un cubrimiento mínimo reducido y abriendo a derecha sus dependencias funcionales.

Algoritmo DescomposiciónFNBC-JSP
Datos de Entrada: un esquema R que no está en FNBC con df's F
Datos de Salida: una descomposición ρ en FNBC join sin pérdida
Comienzo del Algoritmo

$Z := R$
$\rho := \{Z\}$
Repetir
Sea $X \rightarrow A$ una df violando la FNBC en Z.
Descomponer Z en $Z - A$ y XA donde XA está en FNBC.
$\rho := \rho \setminus \{Z\} \cup \{Z - A, XA\}$
$Z := Z - A$
Hasta que Z esté en FNBC

Fin del Algoritmo

Este algoritmo garantiza que la descomposición ρ esté en FNBC y es join sin pérdida [Ullman, 1989]. La subrutina de descomposición no la presentamos aquí aunque puede demostrarse que este algoritmo toma un tiempo polinomial.

Aplicándolo sobre el siguiente conjunto de dependencias funcionales definidas sobre $R(ABCDEKLIJ)$:

$$\{ABD \rightarrow E, AB \rightarrow L, B \rightarrow K, C \rightarrow ABIJ, J \rightarrow C\}$$

obtenemos la descomposición $\rho_3 = (BK, CJ, ABL, ABDE, CA, CB, CI, CD)$ sobre las cuales se proyectan, respectivamente, las df's $\{B \rightarrow K\}$, $\{C \rightarrow J, J \rightarrow C\}$, $\{AB \rightarrow L\}$, $\{ABD \rightarrow E\}$, $\{C \rightarrow A\}$, $\{C \rightarrow B\}$, $\{C \rightarrow I\}$, $\{\}$. Como podemos ver, ρ_3 es join sin pérdida y se preservan dependencias. Más aún, ρ_3 es equivalente a ρ_2 por lo que surge la cuestión: ¿que algoritmo aplicar? En general, no existe una respuesta aplicable a todos los casos. A veces podemos optar con alcanzar 3FN, join sin pérdida y preservación de dependencias o FNBC, join sin pérdida pero perder algunas dependencias. El problema es que 3FN y FNBC son equivalentes cuando existe una única llave o todas las llaves son simples. En caso de tener llaves compuestas y solapadas, los resultados suelen ser diferentes [Date, 2001].

5. Optimizaciones

A partir de las descomposiciones generadas simbólicamente, vemos que podemos obtener esquemas más compactos. Por ejemplo, en las descomposiciones anteriores tenemos como subesquemas $C \to A$, $C \to B$, $C \to I$. Imaginemos que C representa el documento de una persona, A sus nombres, B su apellido e I su edad. Es claro que no tiene mucho sentido tener esquemas separados y es mucho más eficiente (en espacio) tener un único esquema $CABI$ con la dependencia $C \to ABI$. La optimización de fusionar los esquemas con el mismo lado izquierdo fue propuesta por Ullman [Ullman, 1989] para aplicarla después de obtener una descomposición en 3FN que sea join sin pérdida y preserve dependencias.

En este artículo proponemos una optimización sustancialmente mejor a la anterior, que puede ser aplicada a los algoritmos de descomposición en 3FN y FNBC. Dicha optimización está sustentada en la siguiente propiedad.

Propiedad: Sean R_1 y R_2 dos subesquemas en FNBC con dependencias funcionales F_1 y F_2 respectivamente y con al menos una llave en común. Entonces $R_1 \cup R_2$ con dependencias funcionales $F_1 \cup F_2$ está en FNBC.

Demostración: Sean $X_1, \ldots, X_n$ las llaves del esquema R_1 según las df's F_1 y sean $Y_1, \ldots, Y_m$ las llaves del esquema R_2 según las df's F_2. Sabemos que las llaves de un esquema son funcionalmente equivalentes entre si, esto es, $X_1 \to X_2 \to \ldots \to X_n \to X_1$ y $Y_1 \to Y_2 \to \ldots \to Y_m \to Y_1$. Como existe una llave en común entre R_1 y R_2 (esto es, existe $i, 1 \leq i \leq n$ y $j, 1 \leq j \leq m$ tal que $X_i = Y_j$) entonces se verifica que:

$$X_1 \to X_2 \to \ldots X_n \to Y_1 \to Y_2 \to \ldots \to Y_m \to X_1$$

Por lo tanto, las llaves de R_1 serán llaves de R_2 y viceversa. A partir de esto, es trivial demostrar que si $V \to U$ es una df de $F_1 \cup F_2$ entonces V es llave de $R_1 \cup R_2$ por lo cual se satisface la FNBC (**cqd**).

Queda al lector verificar que si (R_1, R_2) es una descomposición join sin pérdida, entonces lo $R_1 \cup R_2$ es un esquema join sin pérdida.

Consideraremos el esquema de relación $R(ABCDEKL)$ con df's:

$$F = \{DE \to C, BC \to D, DL \to E, BL \to A, A \to BC, AD \to K\}$$

Un cubrimiento mínimo reducido para el conjunto F es el siguiente:

$$G = \{DE \to C, BC \to D, DL \to E, BL \to A, A \to BCK\}$$

En este conjunto, las llaves son AL y BL y, por tener dos llaves solapadas, es candidato a que la 3FN no sea equivalente a la FNBC. Abriendo a derecha la df $A \to BCK$ y aplicando el algoritmo de 3FN, join sin pérdida y preservación de dependencias obtenemos la descomposición $\rho_1 = (CDE, BCD, DEL, ABL, AC, AK)$ con los siguientes subesquemas:

- CDE con df's $\{DE \to C\}$ y llave DE.

- BCD con df's $\{BC \rightarrow D\}$ y llave BC.
- DEL con df's $\{DL \rightarrow E\}$ y llave DL.
- ABL con df's $\{A \rightarrow B, BL \rightarrow A\}$ y llaves AL y BL.
- AC con df's $\{A \rightarrow C\}$ y llave A.
- AK con df's $\{A \rightarrow K\}$ y llave A.

En este caso, la optimización consistiría en unir los dos últimos subesquemas, obteniendo $\rho_2 = (CDE, BCD, DEL, ABL, ACK)$ donde el subesquema ACK tiene como df's a $\{A \rightarrow CK\}$ y la llave es A. Tanto ρ_1 como ρ_2 preservan dependencias y son join sin pérdida aunque no están en FNBC ya que el subesquema ABL no la respeta (en la df $A \rightarrow B$, A no es llave).

Si aplicamos el algoritmo de FNBC presentado anteriormente obtenemos la descomposición $\rho_3 = (AK, AC, AD, AB, AEL)$ con los siguientes subesquemas:

- AK con df's $\{A \rightarrow K\}$ y llave A.
- AC con df's $\{A \rightarrow C\}$ y llave A.
- AD con df's $\{A \rightarrow D\}$ y llave A.
- AB con df's $\{A \rightarrow B\}$ y llave A.
- AEL con df's $\{AL \rightarrow E\}$ y llave AL.

Puede verse que aparecen dependencias nuevas que surgen del proceso de descomposición utilizado en el algoritmo de FNBC. Luego, la optimización permitiría unir los primeros cuatro subesquemas, obteniendo $\rho_3 = (AKCDB, AEL)$ donde el subesquema $AKCDB$ tiene como df's a $\{A \rightarrow KCDB\}$ y la llave es A. Tanto ρ_3 como ρ_4 están en FNBC y son join sin pérdida aunque claramente se pierden (muchas) dependencias funcionales. Sin embargo, esta pérdida no es propia de la optimización sino intrínseca del ejemplo modelado.

Una mejor alternativa para normalización consiste en *combinar* ambos algoritmos [Lewis et al., 2002, Falappa, 2003]. Se aplica primero el algoritmo de 3FN, join sin pérdida y preservación de dependencias y, puesto que la 3FN es en muchos casos equivalente a la FNBC, se verifican uno a uno los distintos subesquemas, y sobre cada uno de ellos que viola la FNBC se aplica el algoritmo de FNBC tradicional. La mejora aquí radica en el hecho de que los subesquemas tienen generalmente muchos menos atributos que el esquema original. Una vez hecho esto, se procede a la optimización formulada anteriormente.

En el ejemplo anterior, si consideramos $\rho_1 = (CDE, BCD, DEL, ABL, AC, AK)$ vemos que el único subesquema que no respeta la FNBC es ABL. Si se descompone ese subesquema con el algoritmo de FNBC tradicional se obtienen dos nuevos subesquemas, y la descomposición $\rho_5 = (CDE, BCD, DEL, AB, AL, AC, AK)$ con los siguientes subesquemas:

- CDE con df's $\{DE \rightarrow C\}$ y llave DE.
- BCD con df's $\{BC \rightarrow D\}$ y llave BC.
- DEL con df's $\{DL \rightarrow E\}$ y llave DL.
- AB con df's $\{A \rightarrow B\}$ y llave A.
- AL sin df's y llave AL.
- AC con df's $\{A \rightarrow C\}$ y llave A.
- AK con df's $\{A \rightarrow K\}$ y llave A.

Finalmente, el proceso de optimización generaría una nueva y última descomposición $\rho_6 = (CDE, BCD, DEL, ABCK, AL)$ con los siguientes subesquemas:

- CDE con df's $\{DE \rightarrow C\}$ y llave DE.
- BCD con df's $\{BC \rightarrow D\}$ y llave BC.
- DEL con df's $\{DL \rightarrow E\}$ y llave DL.
- $ABCK$ con df's $\{A \rightarrow BCK\}$ y llave A.
- AL sin df's y llave AL.

que representa la más compacta descomposición en FNBC y en la que sólo se pierde la df $BL \rightarrow A$. Será decisión del diseñador de la base de datos optar por la descomposición ρ_1 (3FN, join sin pérdida y preservación de dependencias) o ρ_6 (FNBC, join sin pérdida y pérdida de una dependencia) en función del significado de los atributos y la importancia de las dependencias funcionales.

6. Conclusiones

Hemos abordado el modelo relacional y los algoritmos de normalización para alcanzar 3FN y FNBC. A partir de algoritmos tradicionales y previamente definidos, se presentó una optimización a los mismos que permiten obtener descomposiciones más compactas que respetan la propiedad de join sin pérdida. Esta optimización también puede aplicarse en formas más altas tales como la 4FN que involucra no solamente dependencias funcionales, sino también dependencias multivaluadas. Estas mejoras en las descomposiciones producirá beneficios en el espacio de almacenamiento de las relaciones (tablas de datos) así como también en los mecanismos de verificación de restricciones de integridad.

Referencias

[Date, 2001] Date, C. J. (2001). *Introducción a los Sistemas de Bases de Datos*. Prentice Hall.

[Falappa, 2003] Falappa, M. A. (2003). Bases de datos: Notas de clase. Universidad Nacional del Sur.

[Lewis et al., 2002] Lewis, P. M., Bernstein, A., and Kifer, M. (2002). *Databases and Transaction Processing: an application-oriented approach*. Addison Wesley.

[Maier, 1983] Maier, D. (1983). *The Theory of Relational Databases*. Computer Science Press.

[Silberschatz et al., 1999] Silberschatz, A., Korth, H., and Sudarshan, S. (1999). *Database System Concepts*. McGraw-Hill, third edition.

[Ullman, 1988] Ullman, J. (1988). *Database and Knowledge-Base Systems*, volume I. Computer Science Press.

[Ullman, 1989] Ullman, J. (1989). *Database and Knowledge-Base Systems*, volume II. Computer Science Press.