Dependencias Funcionales y Normalización de bases de datos relacionales

Rubén Cañón, Yeison Tafur, Ricardo Peñaloza.
Universidad Distrital Francisco José de Caldas;

***Abstract*—This article pretends to show the process of normalize the relation, R(T, L) in a database modeling, in order to minimize the data redundancy and update anomalies** (Hussain, Shamail, & Awais, 2003)**. It presents the design of the algorithm used to solve this problem implementing algorithms to calculate functional dependencies closure, minimal cover and relation´s candidate keys. In addition, the algorithm shows if the relation match with first normal form, third normal form or Boyce-Coddnormal form.**

Palabras clave: Dependencia funcional, Cierre de dependencias funcionales, recubrimiento mínimo, formas normales, llave candidata.

I. INTRODUCTION

En diseño de bases de datos relacionales es importante reducir al máximo la redundancia de datos y evitar anomalías de actualización en el esquema relacional (Hussain et al., 2003) . Para lograr esto se lleva a cabo un proceso de normalización de la relación que está apoyado en algoritmos y técnicas que permiten acercarse a este objetivo.

En este artículo se presentará el proceso de diseño e implementación de una herramienta que permite verificar si una relación cumple con las formas normales (Segunda forma normal, tercera forma normal, forma normal de Boyce-Codd) a través de la implementación de algoritmos para el cálculo de cierre de dependencias funcionales, recubrimiento mínimo y cálculo de llaves candidatas.

Adicionalmente, se presenta los resultados obtenidos de las pruebas realizadas para validar los requerimientos funcionales y no funcionales de la herramienta desarrollada.

II. DEFINICIÓN DEL PROBLEMA

A partir de una relación R(T, L), donde T son los atributos y L son la lista de dependencias funcionales de la relación R. Se requiere eliminar los problemas de redundancia de la relación y verificar si la relación ingresada cumple con la 2FN. 3FN y FNBC. Se deben realizar las operaciones necesarias para descomponer las dependencias funcionales sin pérdida de información buscando reducir el tiempo de ejecución.

La aplicación desarrollada debe estar en la capacidad de recibir la relación R(T, L) tanto en formato JSON como a través de interfaz gráfica.

III. MARCO DE REFERENCIA

A. Normalización

Es encontrar una descomposición adecuada de la "relación universal" de la base de datos que nos permite cumplir con los criterios de eficacia, ausencia de redundancia, evolución, comprensión y flexibilidad. También se puede ver como una serie de reglas que ayudan a desarrollar un esquema que minimice los problemas de lógica.

B. Dependencias Funcionales.

Dados dosatributos X y Y de una relación R, se dice que Y esfuncionalmente dependiente de X si para cada valor de X existeun valor de Y, y sólo uno, asociado con él. Se denota como X→Y. Las DF se determinan al estudiar las propiedades detodos los atributos de la relación y deducir cómo estánrelacionados los atributos entre sí.

Axiomas de Armstrong

Reflexividad: ∀ X⊆T, X → X
Aumentatividad: Si X → Y, ∀ X' tal que X⊆X', entonces X' → Y.
Proyectividad: Si X → Y, entonces X → Y', ∀ Y'⊆Y.**Aditividad**: Si X → Y y Z → W, entonces X∪Z → Y∪W.**Transitividad**: Si X → Y e Y → Z, entonces X → Z

C. Cierre de dependencias funcionales

Para calcular el cierre de un descriptor respecto a un conjunto de DF se utiliza el algoritmo de Ullman que calcula una secuencia recursiva de descriptores:
X(0), X(1), X(2),... X(K) = X(K+1) = XL+
Pasos para el cálculo de XL+

1. X(0) = X.
2. X(i) = X (i-1) U B (tal que (A→B) ^ A ∈ X (i+1)).

3. X (n-1) = X(n) El interés del cálculo del cierre de un descriptor estriba en que dado una DF = X→ Y, y se cumple que si Y ⊆ XL+ ; se puede decir que la dependencia funcional X→Y pertenece a dicho conjunto. Por lo tanto, dado el problema planteado con R(T, L) y L={(X1,Y1),(X2,Y2), …, (Xn, Yn)}. Podemos validar si una dependencia funcional A = (XA→YA) se puede o no derivar, al calcular el cierre de XA (XA +) y comprobar si YA ⊆ XA + .

D. Recubrimiento Mínimo.

PASO 1 (L1): Buscar dependencias elementales:

a. Eliminar la dependencia s tales que X->Y: YcX .
b. Descomponer DF (X→Y) para lograr |Y|=1; X-Yi para todo Yi ϵ Y

Paso 2 (L2): Eliminación de atributos extraños (Bi) a izquierda

X->A es parcial si Z->Y: ZcX y Bi ϵ X
Z= X-Bi
L+ ≡ L- {X-A} U {Z-A}+ = (L1)
L1 ≡ L1+ ∀ |X|>1

Paso 3 (L3): Eliminar dependencias redundantes:

∀ (X-> A) ϵ L2 se determina X+ respecto a L2-{X-A} . Si A ϵX+ entonces (X→A) es redundante en L2, Por lo tanto, debe eliminarse.

E. Llaves

1. Calcular el conjunto de atributos esenciales:

$$\mathbf{Z=T-} \bigcup_{i=1}^{n} Y_i$$

2. Calcular el cierre del descriptor obtenido en el paso anterior: (ZL)+. Si (ZL)+ =T, entonces Z es clave única.

3. Calcular el conjunto de atributos prescindibles o imposibles:

$\mathbf{W=T-} \bigcup_{i=1}^{n} X_i$ y el conjunto de atributos posibles:

$$\mathbf{V = T - \{W \cup Z_L^{+}\}}$$

siendo V un conjunto ordenado.

4. Hacer:

$$\mathbf{M_1 = \{Z \cup A\}, \forall\ A / A \in V\ y\ M_2 = \varnothing .}$$

5. Calcular:

$$\mathbf{(U)_L^{+}, \forall\ U / U \in M_1 .}$$

6. Si (UL) + = T entonces introducimos U en M2 y lo borramos de M1.
7. Si (UL) + ≠ T, entonces sustituimos U en M1 por U u B, ∀ B / B ϵ V, B ϵ UL+, ord(B)V>ord(A)V, (siendo AcU el último atributo añadido en pasos anteriores a dicho descriptor U) y Xi c {UUB} (siendo Xi cualquier implicante de los existentes en L).
∀ P y Q / P ϵ M1 y Q ϵ M2: Si QcP borrar P de M1.
8. Si M1 = Ø, eliminar súper conjuntos de M2. M2 contiene todas las claves. En caso contrario volver al Paso 5.

F. Formas Normales

Primera forma normal (1FN)

G. Una relación está en primera forma normal si todo atributo contiene un valor indivisible, atómico(Abraham Silberschatz; Henry Korth, 2002).

Segunda forma Normal (2FN)

Una relación está en segunda forma normal si, y sólo si:

- Está en 1FN.
- Todo atributo que no pertenezca a la clave debe depender de la clave en su totalidad. Es decir, los registros no deben depender de nada aparte de su clave primaria.

Dependencia transitiva

Se tiene la relación R(A, B,C). Si A →B, B →C y A no depende funcionalmente de B, entonces se dice que C depende transitivamente de A y se puede formar la cadena A → B → C.

Tercera forma normal (3FN)

Una relación está en 3FN si, y sólo si:

- Está en 2FN.
- Todo atributo que no pertenezca a la clave no depende de un atributo o clave.

La 3FN elimina las redundancias ocasionadas por las dependencias transitivas.

Deficiencias de la 3FN

- hay varias claves candidatas.
- Las claves candidatas son compuestas.
- las claves candidatas tienen por lo menos un atributo en común (se traslapan).

Forma normal de Boyce-Codd (FNBC)

Una relación esta en FNBC si y solo si:

- está en 3FN
- Cada dependencia funcional no trivial tiene una clave candidata como determinante.

Simplificando, una tabla está en FNBC si está en 3FN y los únicos determinantes son claves candidatas.

IV. DISEÑO DE LA HERRAMIENTA

La herramienta desarrollada se implementó en lenguaje Python con una interfaz gráfica en Django. Esta herramienta permite cargar la relación R(T,L) desde un archivo Json (Figura 1). Adicionalmente permite ingresar los atributos y las dependencias funcionales ingresándolas en la interfaz gráfica.

La herramienta realiza el cálculo del recubrimiento mínimo y las llaves candidatas de la relación, apoyándose en el cierre de un descriptor sobre el conjunto de dependencias funcionales. A continuación se presenta el proceso de diseño de la aplicación.

A. Algoritmo de cierre de un descriptor X sobre DF.

El la Figura 2 se representa el proceso para el cálculo del cierre de dependencias funcionales. Este cierre es necesario para el posterior cálculo del recubrimiento mínimo de la relación R(T,L).

```
{
"relacion": {
        "atributos": [
                        {  "t": "A"},
                        {  "t": "B"},
                        {  "t": "C"},
                        {  "t": "D"},
                        {  "t": "E"},
                        {  "t": "F"},
                        {  "t": "G"}
        ],
        "dFuncionales": [    {
                            "x": "AB",
                            "y": "C"
                        },
                        {
                            "x": "BE",
                            "y": "C"
                        },
                        {
                            "x": "BC",
                            "y": "D"
                        },
                        {
                            "x": "CF",
                            "y": "B"
                        },
                        {
                            "x": "CF",
                            "y": "D"
                        }
                    ]
    }
}
```

Ilustración 1Estructura de relación JSON

B. Implementación del algoritmo del recubrimiento mínimo.

Para lograr eliminar DF triviales, atributos extraños y DF redundantes se debe aplicar el algoritmo de cubrimiento mínimo. En la figura 3 se representa el proceso para hallar el conjunto de dependencias funcionales mínimo en una relación R(T,L).

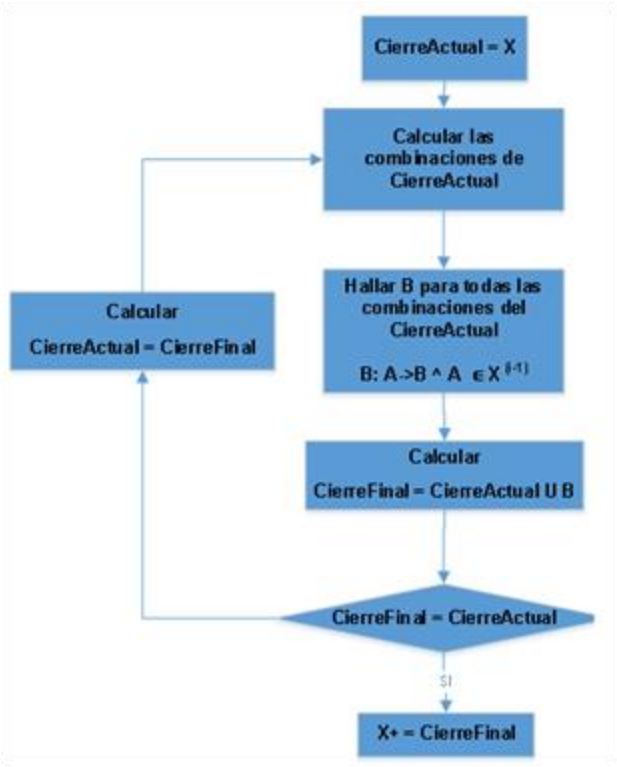

Figura 2 Cierre de DFs

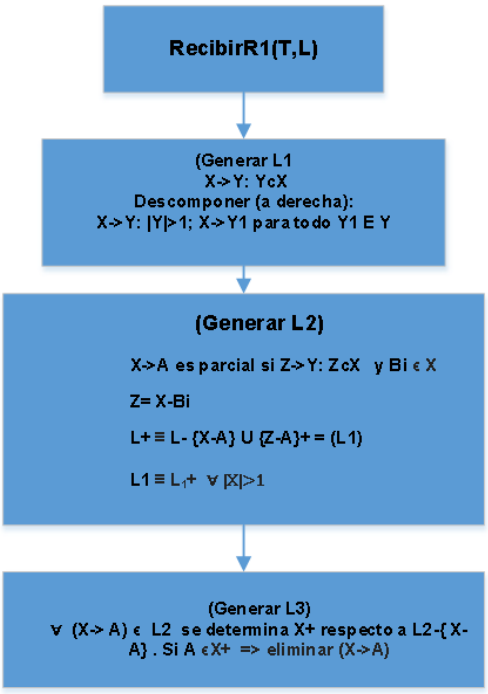

Figura 3.Recubrimiento mínimo.

C. *Implementación del algoritmo de cálculo de llaves.*

Para calcular llaves partimos de un conjunto de dependencias funcionales con recubrimiento mínimo. A continuación, calculamos Z y su cierre, si el cierre de Z es igual a los atributos T, Concluimos que Z es llave única de lo contrario seguimos el proceso representado en la figura 4. Hasta alcanzar un conjunto M2 de llaves candidatas

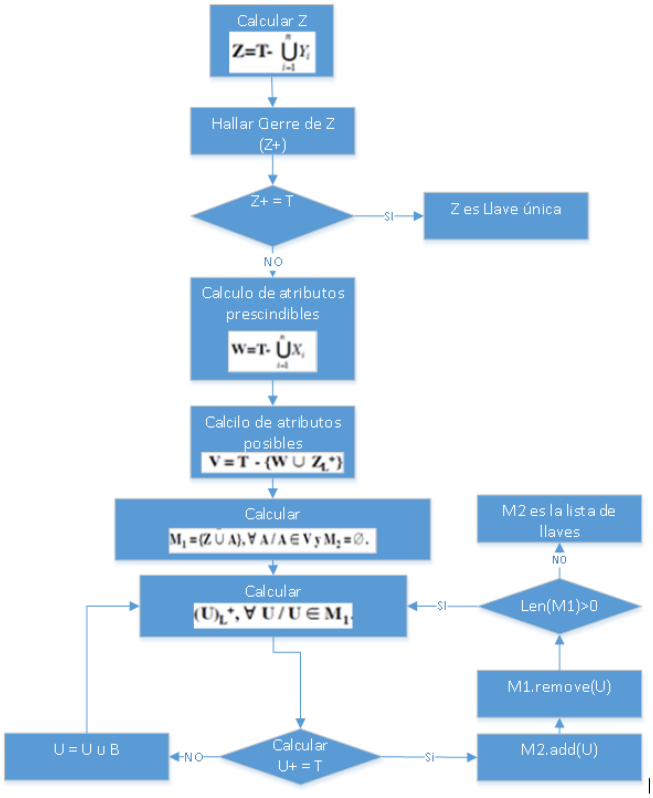

Figura 4. Cálculo de Llaves

D. *Verificación de formas normales*

El proceso de verificación de formas normales se hizo en cascada, es decir, al comprobar que la relación superaba la 2FN era posible continuar el proceso con la 3FN de lo contrario se descartaban las formas normales de mayor nivel. Partiendo del hecho que en el diseño de bases de datos, la primera premisa es contar con elementos atómicos, obviamos el cálculo de la 1FN. A continuación aplicamos las reglas descritas en el apartado F. Una vez son comprobadas las formas normales presentamos en la interfaz grafica los resultados obtenidos.

V. PRUEBAS DE LA HERRAMIENTA

Para la ejecución de la herramienta se plantearon diferentes escenarios básicos, los cuales fueron validados desde la interfaz gráfica para optimizar la ejecución de los algoritmos, los casos de validación iniciales tenidos en cuenta son los siguientes

- Ingresar Alfabeto
- Ingresar Dependencias
- Validar que no se ingresen dependencias triviales de la forma {A→ A}
- Validar que no se puedan repetir implicantes de la forma {A,B,C,A →B,D}
- Validar que no se puedan repetir elementos implicados en la misma dependencia funcional de la forma { A,B,C→ D,E,E}

Luego de realizar los filtros básicos, se puede realizar el proceso de cálculo de llaves, la ejecución realiza implementación de hilos según la cardinalidad de V, es decir para V= {A,B,C} se realiza la ejecución de 3 hilos.

VI. CONCLUSIONES

Al aplicar los algoritmos de recubrimiento mínimo y calculo de llaves candidatas fue posible implementar una herramienta que a partir de una relación R(T,L) fuera capaz de eliminar los problemas de redundancia de dependencias funcionales y atributos extraños en la relación. A partir de este conjunto de dependencias mínimo se reduce el costo computacional para el cálculo de llaves ya que en este punto se cuenta con menos dependencias funcionales y simplificadas. De esta manera se facilita la construcción del conjunto M2 de llaves candidatas, en el caso de no encontrar una llave única.

Es precisamente en este paso donde se pudo aprovechar el uso de hilos de ejecución para lanzar tantos procesos como elementos en la lista V se encontraran, esto permitió que los tiempos de ejecución se redujeran.

VII. REFERENCIAS

[1] Abraham Silberschatz; Henry Korth, S. S. (2002). Fundamentos de bases de datos (Cuarta). McGraw-Hill Inc.

[2]. Hussain, T., Shamail, S., & Awais, M. M. (2003). ELIMINATING PROCESS OF NORMALIZATION IN RELATIONAL, 408–413.

Rubén Cañon: Ingeniero electronico, Universidad distrital Francisco José de Caldas. **Codigo: 20182495005. Cod. Lista 10**

Ricardo Peñaloza: Ingeniero en telematica, Universidad distrital Francisco José de Caldas. **Codigo: 20182495009. Cod. lista 14**

Yeison Tafur: Ingeniero en telematica, Universidad distrital Francisco José de Caldas**. Codigo: 20182495015. Cod. Lista 20**