Anales PANEL'81/12 JAIIO
Sociedad Argentina de informática
e Investigación Operativa. Buenos Aires, 1981

AUTOMATIZACION DEL PROCESO DE NORMALIZACION DE UNA BASE DE DATOS RELACIONAL

J. L. Becerril
R. Casajuana
F. Valer
Centro de Investigación UAM-IBM
Paseo de la Castellana 4, Madrid-1, España

J. Guizán
Facultad de Informática
Universidad Politécnica de Madrid, España

RESUMEN

Este trabajo describe un sistema para simplificar (y en algunos casos automatizar) el proceso de normalización de una base de datos relacional. Este sistema, ademas de obtener las posibles normalizaciones de la base de datos en 2NF, 3NF y 4NF, calcula el volumen de la base de datos para las distintas normalizaciones.

En la primera parte de la comunicación se introduce la teoría de normalización de Codd ampliada con la cuarta forma normal de Fagin. En el resto se presenta el tratamiento que reciben las dependencias funcionales y multivaluadas para obtener la base de datos en 2NF, 3NF y 4NF. En el apéndice final se incluye un ejemplo realizado con el sistema propuesto.

INTRODUCCION

Desde 1970, en que E.F. Codd, con su trabajo sobre relaciones n-arias en un contexto de base de datos, plantea los fundamentos de la teoría de bases de datos relacionales (BDR), se han realizado enormes esfuerzos dentro de este campo, tanto a nivel formal como al de diseño e implementación de prototipos.

El concepto de normalización (1) fué introducido por Codd en 1970 rigorizándolo posteriormente en 1971. El motivo de la aparición de este concepto fué la observación de que para idénticos datos, diferentes organizaciones entre los mismos en forma de relaciones presentaban muy distintas propiedades en cuanto a actualización, inserción o eliminación de información. Se trataba pues de adoptar organizaciones de los datos que permitieran que las operaciones sobre las relaciones se realizaran con la máxima simplicidad y manteniendo la integridad de la BDR.

El trabajo que se presenta describe un sistema para simplificar (y en algunos casos automatizar) el proceso de transformación de una base de datos en cada una de las cuatro formas normales.

La lógica general es la siguiente:

Una vez que el diseñador de la base de datos ha introducido, utilizando una sintaxis predefinida, las dependencias funcionales o multivaluadas y los datos de tipo estadístico (tamaño medio de las tuplas, número medio de apariciones de distinto valor por dominio, etc.) el sistema evalúa la base de dependencias funcionales que definen la BDR siguiendo el algoritmo de Beeri(5). En función de esta base de dependencias funcionales se normaliza la BD en 2NF. La obtención de la BD en 3NF se realiza siguiendo un algoritmo debido a Bernstein(6). Posteriormente, a partir de las 3NF, y teniendo en cuenta las dependencias multivaluadas, se normaliza la base en 4NF.

NORMALIZACION

El concepto de normalización fué introducido por Codd en (2) rigorizándolo posteriormente en (4). El motivo de la aparición de este concepto fué la observación de que para idénticos datos, diferentes agrupaciones en relaciones presentaban muy distintas propiedades en cuanto a actualización, inserción, etc.

La teoría que vamos a describir se basa en una serie de formas normales (llamadas primera, segunda, tercera, de Boyce-Codd y cuarta formas normales) que proporcionan sucesivas mejoras en cuanto a las propiedades de modificación de una base de datos (3), (4), (5) y (7).

PRIMERA FORMA NORMAL. Se dice que la relación R es una primera forma normal sii ninguna de sus tuplas tiene elementos que a su vez sean conjuntos. Una relación es no normalizada si no es una primera forma normal.

Dependencia funcional (DF). Sean dos colecciones de atributos A y B de R; se dice que B es funcionalmente dependiente de A, y se escribe RA -> RB, sii en cada instante a cada valor de A en R le corresponde a lo sumo uno de B. En todos los casos en que no haya posible confusión, y con el fin de simplificar la notación, en lugar de RA -> RB escribiremos A -> B.

Superclave. Se dice que K es una superclave si es un conjunto de atributos de una relación del que dependen funcionalmente los restantes.

Clave. Se dice que una superclave es una clave si es una superclave mínima, es decir, si no existe otra superclave que esté contenida en ella.

Atributos primarios. Se dice que un atributo de una relación es primario sii aparece en la clave elegida.

Dependencia funcional total. Sean D y E dos subcolecciones distintas de atributos de R, siendo E funcionalmente dependiente de D. Se dice que E presenta, respecto a D, una dependencia funcional total en R sii E no es funcionalmente dependiente de ningún subconjunto de D distinto del total.

SEGUNDA FORMA NORMAL. Se dice que R es una segunda forma normal sii:

1) Es una primera forma normal.
2) Todo atributo que no sea primario es totalmente dependiente de cada clave de R.

Dependencia transitiva. Consideremos tres colecciones distintas de atributos A, B, C en una relación R cuyo grado es mayor o igual a tres. Supongamos que se verifican las tres condiciones independientes del tiempo siguientes: RA -> RB RB $\not\rightarrow$ RA RB -> RC. Si además se verifican RA -> RC y RC $\not\rightarrow$ RA, se dice que C es transitivamente dependiente de A bajo R.

TERCERA FORMA NORMAL. Diremos que una relación R es una tercera forma normal sii:
1) Es una segunda forma normal.
2) Cada atributo que no sea primario, no es transitivamente dependiente de cada clave de R.

FORMA NORMAL DE BOYCE-CODD. R es BCNF sii es 3NF y para todo par de conjuntos de atributos X,Y ($X,Y \neq \phi$, $X \cap Y = \phi$), si X -> Y entonces X es una superclave de R. Con otras palabras, si algun atributo depende de X, todos los demás también.

Proyección de un atributo. Consideremos tres colecciones disjuntas de atributos X,Y,Z y la relacion R(X,Y,Z). Se define la proyección de Y sobre el punto x,z a:
$Y(x,z) = \{ y \in Y;\ (x,y,z) \in R \}$.

Dependencias Multivaluadas. Se dice que un conjunto de atributos (Y) presenta una dependencia multivaluada (DMV) de otro conjunto de atributos (X) en la relación R(X,Y,Z), y se denota $X \twoheadrightarrow Y$, sii $Y(x,z) = Y(x,z')$ para cada x,z,z' tales que $Y(x,z) \neq \phi$, $Y(x,z') \neq \phi$.

Dependencias Multivaluadas Triviales. Sea R(X,Y); las dependencias multivaluadas triviales son del tipo $X \twoheadrightarrow \phi$, o $X \twoheadrightarrow Y$.

Propiedades de las Dependencias Multivaluadas

- Toda dependencia funcional es multivaluada.
- Si Y presenta una dependencia multivaluada respecto a X en R(X,Y,Z) entonces R es el join de sus proyecciones R'(X,Y) y R''(X,Z).

CUARTA FORMA NORMAL. Se dice que R es una 4NF sii para toda dependencia multivaluada no trivial $X \twoheadrightarrow Y$, entonces $X \rightarrow A$ para todos los atributos A de R.

AUTOMATIZACION DEL PROCESO DE NORMALIZACION

A continuación citaremos algunos resultados referentes a la teoria de la normalización que son de interés en el proceso de su automatización:

- El diseño de una BD depende del conocimiento que se tenga sobre las dependencias funcionales y multivaluadas existentes entre los atributos de los datos. Salvo en el caso de una BD estática, las dependencias no pueden conocerse automáticamente, sino que el diseñador de la BD debe definirlas a partir de la semántica de la información.

- El diseño de la BD en cualquier nivel de normalización no es único y por lo tanto se plantea la posibilidad de escoger una forma normal óptima. Se han realizado numerosos estudios sobre cómo llegar a esta optimización, que no citaremos aquí por quedar fuera del alcance de este trabajo.

- Existe siempre un esquema relacional en 3NF y en 4NF.

- No siempre existe un esquema en BCNF (5).

Dentro de un proceso para automatizar el proceso de normalización anteriormente descrito se pueden distinguir dos

tipos de problemas bien diferenciados. El primero de ellos se refiere al problema teórico de partir de un conjunto de dependencias funcionales y/o multivaluadas y obtener una base de datos en 2NF, 3NF, BCNF o 4NF. El segundo contempla el aspecto práctico relativo a la elección de una posible normalización en función de datos de tipo estadístico (tamaño medio de las tuplas, número medio de apariciones de distinto valor por columnas, cardinalidad y grado de las relaciones, etc.) y características de la BD. En esta comunicación nos vamos a restringir al primer problema, pudiendo verse en (8) una más completa exposición del tema.

Una vez el diseñador de la BD ha suministrado el conjunto de dependencias funcionales y multivaluadas, el sistema realiza las siguientes operaciones:

- Obtención de una base de dependencias funcionales.
- Obtención de la BD en 2NF.
- Obtención de la BD en 3NF.
- Obtención de la BD en 4NF.

ALGORITMO DE EXTRACCION DE DEPENDENCIAS FUNCIONALES.

Amstrong (9) demuestra que el conjunto mínimo de propiedades de las DF, por cuya aplicación se obtienen las DF derivadas son:

- Reflexividad: Si $Y \subseteq X$ entonces $X \rightarrow Y$. Esta propiedad es obvia pues en cada instante cada valor de Y tiene asociado a lo sumo uno en X (el mismo valor).
- Aumentación: Si $X \rightarrow Y$ y $W \subseteq Z$, entonces $X \cup Z \rightarrow Y \cup W$. Propiedad evidente por la anterior y por definición de dependencia funcional.
- Seudotransitividad: Si $X \rightarrow Y$ y $Y \cup Z \rightarrow W$ entonces $X \cup Z \rightarrow W$. Esta regla de sustitución es evidente a partir de la definición de DF.

* Al conjunto de DF que se obtiene por sucesiva aplicación de las reglas anteriores a un conjunto F de DF se le denomina cierre de F^+.

* Dado un conjunto F de DF se dice que un conjunto H es una cobertura de F si tiene el mismo cierre que F.

* Una cobertura se dice que es no-redundante si no contiene ningún subconjunto que a su vez sea cobertura.

* Una DF $f \in F$ se dice redundante en F si pertenece al cierre de $F - \{f\}$.

* Una cobertura se dice redundante si y solo si contiene DF redundantes.

El problema que nos planteamos en esta sección es: dado un conjunto de DF obtener una cobertura no redundante. Aunque las tres reglas son el conjunto mínimo completo de propiedades de las DF, es conveniente, desde un punto de vista práctico, introducir dos reglas adicionales que son consecuencia inmediata de las anteriores.

- Union: Si X -> Y y X -> Z entonces X -> YZ. En efecto, aplicando la aumentación a X -> Z para Y se tiene XY -> YZ, aplicando a esta regla y a la X -> Y la seudotransitividad se tiene X -> YZ.
- Descomposición: Si X -> Y y Z $\subseteq$ Y entonces X -> Z. Por la reflexividad Y -> Z, como X -> Y aplicando la seudotransitividad queda X -> Z.

De estas últimas dos reglas se obtiene inmediatamente que la dependencia funcional X -> A B ...K, es equivalente al conjunto de dependencias funcionales X -> A, X -> B,... X -> K. Así pues, para probar que una dependencia funcional X -> A B ...K es derivable de un conjunto de DF es suficiente con probar que lo son las X -> A, X -> B,..., X -> K.

Arboles de derivacion. Sea F un conjunto dado de DF donde cada dependencia tiene solo un atributo en su parte derecha (elección factible en función del parrafo anterior). Una derivación de una dependencia funcional f desde F es una secuencia $f_1,\ldots,f_n$ tal que f_n =f y para cada i entre 1 y n se verifica alguna de las siguientes propiedades:

- La DF f_i pertenece a F.
- La DF f_i es el resultado de aplicar alguna de las propiedades de las dependencias funcionales en el conjunto $F - \{f_i\}$.

Con el fin de tratar a nivel práctico el problema de extracción de una cobertura no redundante, vamos a introducir un modelo denominado arbol de derivación, en el que es posible realizar todas las operaciones definidas sobre árboles a la vez que cumplen las propiedades referentes a las DF.

Se define el conjunto de árboles de derivación basados en F por medio de las reglas:

- Si A es un atributo entonces un nodo etiquetado con A es un árbol de derivación basado en F.
- Si T es un árbol de derivación con un nodo hoja etiquetado con A y la dependencia funcional B ... M -> A pertenece a F, entonces el árbol construido a partir de T añadiéndole B,...,M como hijos de A es también un árbol de derivación.
- Un árbol etiquetado es un árbol de derivación solo si es posible obtenerlo por aplicación de un número finito de veces de las dos reglas anteriores.

En este punto ya estamos en condiciones de desarrollar un algoritmo para que, dado un conjunto de dependencias funcionales F y una nueva DF f, podamos determinar si f pertenece a la cierre de F (5).

Consideremos un conjunto de dependencias funcionales F, definidas sobre un conjunto de atributos A,B,...,M. Supongamos que todas las dependencias funcionales que tienen la misma parte izquierda están agrupadas en una sola DF (este hecho no es restrictivo, pues se puede conseguir sin más que aplicar la regla de unión). Representemos el conjunto F por una cadena de pares donde cada par representa una DF y denominemos PD a su parte derecha y PI a la izquierda. Cada parte es un conjunto de atributos que representaremos por números enteros del conjunto 1,2,...,m . La longitud de la representación de F la denotaremos por # F. Sea f : X -> S donde X y S son subconjuntos de {A,B,...,M}. Como cada atributo de f aparece en al menos una DF de F, podemos asumir que # f $\leq$ # F.

El método para probar si f $\in$ F^+ es calcular el conjunto de atributos que son funcionalmente dependientes de X. Sea DEP un conjunto que contiene a esos atributos. Asignemos inicialmente el valor X a DEP, puesto que por reflexividad X es funcionalmente dependiente de X. Para buscar nuevos atributos que se puedan añadir a DEP, seleccionamos una DF, f', que esté en F y cuya parte izquierda esté contenida en DEP pero no su parte derecha. Por seudotransitividad la parte derecha de f' es funcionalmente dependiente de X y puede añadirse a DEP. Podemos continuar seleccionando DF de esta manera, añadiendo su parte derecha a DEP, hasta que ya no se pueda añadir ningún nuevo atributo.

Conceptualmente, DEP contiene un conjunto de atributos cada uno de los cuales es la raiz de un árbol de derivación basado en F y cuyas hojas son las X. Cada vez que encontramos una DF, f', cuya parte izquierda está contenida en DEP, la segunda regla de construcción de los árboles de derivación establece que cada atributo de la parte derecha de f' puede ser la raiz de un arbol de derivación cuyas hojas son X. Sabemos que f$\in$ F^+ si y solo si S esta en DEP.

Una aplicación importante de este algoritmo es la eliminación de DF redundantes en F. Para determinar si una f es redundante en F, aplicamos el algoritmo anterior para probar si f$\in$ $(F - \{f\})^+$. Así pues, un subconjunto de F que sea cobertura no redundante puede ser calculado con el siguiente procedimiento: Hacer G = F, un bucle para cada f F en el que si f$\in$ $(G - \{f\})^+$ se haga G = G -{f}. El valor final de G dependerá en general del orden en que se han seleccionado las DF de F. Además G será por construcción un subconjunto de F, aunque este no sea un requerimiento de cobertura no redundante.

ALGORITMO DE OBTENCION DE LA BASE DE DATOS EN 2NF.

Una vez se ha obtenido la base de DF y eliminado las redundantes, se evaluan las claves de las futuras relaciones en 2NF. El algoritmo para la evaluación de las claves, inicializa un conjunto H al vacio y un contador k a 0. Al finalizar el proceso H contendrá las claves y el contador el número de claves distintas. Este algoritmo consta de un bucle principal en las DF de la base. Para cada DF(i-sima) se comprueba si su parte izquierda PI_i esta contenida en H; si es así no se realiza ninguna operación, en caso contrario se hace un bucle secundario en las restantes DF, para cada DF(j-sima) se comprueba si la intersección de PI_i y PI_j es vacia en caso afirmativo no se hace operación alguna si no es así se reasigna a PI_i la unión de PI_i y PI_j. Al final de este bucle secundario se asigna a k el valor k+1 y a H_k el valor PI_i final, reasignandose a H la unión de H y H_k.

Al final se obtiene una colección de H_1, H_2, ...,H_k que serán las claves de las 2NF.

En función de los H_j j=1,2, ...,k se construyen k relaciones R_j en las que aparecen los H_j y los dominios B_s que dependan funcionalmente de H_j. En el supuesto de que estos B_s no dependan funcionalmente de ningún subconjunto de H_j distinto del total esta R_j está en 2NF. En el caso de que algún B_s dependa de un subconjunto SH_j de la clave H_j se elimina este B_s de R_j. Al eliminarlo se contempla la posibilidad de que en otra R_ℓ aparezca SH_j en cuyo caso se agrega alli el B_s, volviendo a considerarse la relación así obtenida para el estudio anterior. Si no existe ninguna relación de las creadas en las que aparezca SH_j se define una nueva relación cuyos dominios son SH_j y B_s.

Al final se comprueba si existe alguna relación R_j que sea unión de otras, eliminandose en este caso R_j.

ALGORITMO DE OBTENCION DE LA BASE DE DATOS EN 3NF.

Uno de los métodos mas simples de obtener relaciones a partir de un conjunto dado de dependencias funcionales, es agrupar en una misma relación todos los atributos que son funcionalmente dependientes de un mismo conjunto de atributos. Esto sugiere el siguiente procedimiento: Primero, dividir el conjunto de dependencias funcionales en grupos de tal forma que todas las DF en cada grupo tengan idénticas partes izquierdas. Segundo, para cada grupo construir una relación que conste de todos los atributos que aparecen en ese grupo. De esta forma, la parte izquierda de cada DF en el grupo es una clave de la relación correspondiente. Este método tiene, sin embargo, algunos inconvenientes:

- Las relaciones obtenidas no están en 3NF. Esto es debido a redundancias en el conjunto de DF.

- Las partes izquierdas de cada DF de un grupo, no son necesariamente claves de la relación formada a partir de esas DF; aunque siempre serán superclaves de la relación.

- Este método da origen en muchos casos a más relaciones de las necesarias.

- Al no considerarse las reglas de composición de las DF las DF redundantes que entren a formar parte de una relación, introducirán en ella atributos extra y contribuirán con ello a conexiones no normalizadas entre atributos.

Asi pues, una primera medida para aliviar el problema de la normalización, es el obtener una cobertura no redundante del conjunto de DF. Sin embargo, esta medida no es suficiente para evitar los problemas que las DF pueden causar en una relación; nos referimos al problema de los atributos innecesarios dentro de una dependencia funcional. Se dice que un atributo es innecesario si su eliminación no altera la clausura del conjunto de DF.

La eliminación de atributos innecesarios en las DF, ayuda a evitar dependencias parciales y superclaves que no son claves, en una relación R.

Una propiedad utilizada en el algoritmo para obtener la tercera forma normal es: Si dos relaciones tienen claves, que son funcionalmente dependientes la una de la otra (o sea, son equivalentes) entonces las dos relaciones pueden unirse. Según se vio antes la condición que debe cumplir una relación para que esté en 3NF es que ningún atributo no primario sea transitivamente dependiente de cada clave de la relación. Para probar que ningún atributo no primario es transitivamente dependiente de cualquier clave de una relación R, enunciamos la siguiente propiedad: Un atributo A se dice que satisface la propiedad P en la relación R, si verifica la siguiente proposiciónÑ Sea H una cobertura no redundante de un conjunto F de DF. Si la DF K -> A está en H y K -> A se utiliza para formar la relación R, entonces para cualquier atributo primario B de R, la DF K -> B puede ser derivada sin usar para ello la DF K -> A, o sea puede ser derivada de H-(K-> A). Se demuestra que una violación de la propiedad P puede inducir a una violación de la 3NF(5). Para obtener una relación R en 3NF, basta con eliminar de cada relación R todo atributo no primario que sea transitivamente dependiente de alguna clave de R, cambiándolo a otra o creando una nueva relación, con lo que el esquema de relaciones resultantes englobaría a las mismas DF.

Una vez expuestos todos los conceptos anteriores, pasamos a exponer el algoritmo en el que se ha basado la obtención de la 3NF (6).

PASO-1: Eliminar atributos innecesarios. Sea F el conjunto de

DF dado. Eliminar atributos innecesarios de la parte izquierda de cada DF en F, obteniendo como resultado el conjunto G.

PASO-2: Encontrar una cobertura (H) no redundante de G.

PASO-3: Partición. Dividir H en grupos, de tal forma que todas las DF pertenecientes a un grupo tengan idénticas partes izquierdas.

PASO-4: Mezclar claves equivalentes. Tomar J igual al conjunto vacio. Para cada par de grupos, H_i y H_j, con partes izquierdas X e Y, respectivamente, mezclar H_i y H_j si existe una biyección en H^+ entre X e Y. Para cada biyección, agregar $X \rightarrow Y$ e $Y \rightarrow X$ a J. Para cada A perteneciente a Y, si $X \rightarrow A$ está en H, entonces borrar esta DF de H. Hacer lo mismo para cada B perteneciente a X, tal que $Y \rightarrow B$ está en H.

PASO-5: Eliminar dependencias transitivas. Encontrar un H' contenido en H tal queÑ $(H' + J)^+ = (H + J)^+$, y ningún subconjunto propio de H' tiene esta propiedad. Agregar cada DF de J a su correspondiente grupo de H'.

PASO-6: Construir relaciones. Para cada grupo, contruir una relación que conste de todos los atributos que aparecen en ese grupo. Cada conjunto de atributos que aparecen en la parte izquierda de cualquier DF en el grupo, es una clave de la relación (el PASO-1 garantiza que ninguno de esos conjuntos contiene atributos extra). El conjunto de relaciones construidas constituyen un esquema para el conjunto de DF dado.

ALGORITMO DE OBTENCION DE LA BASE DE DATOS EN 4NF.

Una vez obtenida la BD en 3NF para el paso a 4NF se consideran las DMV que el diseñador de la BD introdujo. Para obtener la BD en 4NF inicializamos un conjunto R en donde se incluyen todas las 3NF. El algoritmo consta de un bucle general sobre todas las DMV; para la i-sima DMV, se considera un elemento de R que incluya la parte izquierda y derecha de la DMV, sea R_i; si allí la DMV es de tipo trivial no se modifica R, en caso contrario se evalua el número de atributos que dependen funcionalmente de la parte izquierda de la DMV, si ese número coincide con el grado de R_i se sigue con la siguiente DMV, si ese número es no nulo y distinto al grado de R se define una nueva relación R_i' que tiene por dominios la parte izquierda y todos los dominios que dependen funcionalmente de dicha parte izquierda de la DMV, si ese numero es cero se define una R_i' que consta únicamente de la partes izquierda y derecha de la DMV. Por último se modifica R_i eliminándole los atributos que dependen funcionalmente de la parte izquierda de la DMV considerada, agregando R_i' a R y sustituyendo la nueva R_i en R.

APENDICE

Consideremos una empresa textil, que posee datos referentes a sus proveedores, que se citan a continuación:

P#	Identificador del proveedor
NOM-P	Nombre del proveedor
CIUDAD	Ciudad de residencia del proveedor
REGION	Región de residencia del proveedor
C#	Identificador del componente
NOM-C	Nombre del componente
COLOR	Color del componente
CANT	Cantidad total pedida
P-M	Precio por metro
TIPO	Tipo de proveedor
BENEFICIO	Beneficio obtenido
AÑO	Año

Hagamos algunas hipótesis razonables sobre estos datos como pueden ser que todas las componentes independientemente del proveedor tienen un único precio por metro, y que un mismo proveedor puede tener dos tipos distintos (p.ej. sea proveedor de forros y telas).

Con estas hipótesis, y las características intrínsecas a los restantes datos, las dependencias funcionales y multivaluadas serán:

P# -> NOM-P
P# -> REGION
P# -> CIUDAD
CIUDAD -> REGION
P#.C# -> CANT
C# -> P-M
C# -> NOM-C
C# -> COLOR
P#.C# -> P-M
NOM-P.TIPO.AÑO -> BENEFICIO

NOM-P ->-> BENEFICIO.AÑO
NOM-P ->-> TIPO

Al introducir en el sistema esta colección de dependencias se obtiene:

INFORMACION SOBRE LOS ATRIBUTOS INNECESARIOS

EL ATRIBUTO P# ES INNECESARIO EN LA DEPENDENCIA FUNCIONAL: 9

INFORMACION SOBRE EL CALCULO DE
LAS DEPENDENCIAS FUNCIONALES BASICAS

LA DEPENDENCIA FUNCIONAL NUMEROÑ 2 , NO ES BASICA
LA DEPENDENCIA FUNCIONAL NUMEROÑ 6 , NO ES BASICA
EL RESTO DE LAS DEPENDENCIAS FORMAN UNA COBERTURA

BASE DE DATOS EN SEGUNDA FORMA NORMAL

LA BASE DE DATOS EN SEGUNDA FORMA NORMAL
ESTARIA FORMADA POR LAS SIGUIENTES RELACIONES :

* R01(P#,C#,CANT)
  CLAVE = P#,C#

* R02(CIUDAD,REGION)
  CLAVE = CIUDAD

* R03(NOM-P,TIPO,AÑO,BENEFICIO)
  CLAVE = NOM-P,TIPO,AÑO

* R04(P#,NOM-P,CIUDAD)
  CLAVE = P#

* R05(C#,NOM-C,COLOR,P-M)
  CLAVE = C#

BASE DE DATOS EN TERCERA FORMA NORMAL

LA BASE DE DATOS EN TERCERA FORMA NORMAL
ESTARIA FORMADA POR LAS SIGUIENTES RELACIONES:

* R01(P#,NOM-P,CIUDAD)
  CLAVE= P#

* R02(CIUDAD,REGION)
  CLAVE= CIUDAD

* R03(P#,C#,CANT)
  CLAVE= P#,C#

* R04(C#,NOM-C,COLOR,P-M)
  CLAVE= C#

* R05(NOM-P,TIPO,AÑO,BENEFICIO)
  CLAVE= NOM-P,TIPO,AÑO

BASE DE DATOS EN CUARTA FORMA NORMAL

LA BASE DE DATOS EN CUARTA FORMA NORMAL ESTARIA FORMADA POR LAS SIGUIENTES RELACIONES:

* R01 (P#, NOM-P, CIUDAD)
  CLAVE= P#

* R02 (CIUDAD, REGION)
  CLAVE= CIUDAD

* R03 (P#, C#, CANT)
  CLAVE= P#, C#

* R04 (C#, NOM-C, COLOR, P-M)
  CLAVE= C#

* R05 (NOM-P, AÑO, BENEFICIO)
  CLAVE= NOM-P, AÑO, BENEFICIO

* R06 (NOM-P, TIPO)
  CLAVE= NOM-P, TIPO

REFERENCIAS

(1). J. L. Becerril, R. Casajuana, F. Valer Sistemas de gestion de bases de datos relacionales, Inforprim 1978.

(2). E. F. Codd, A Relational Model of Data for Large Shared Data Banks. Comm. ACM Vol. 13, No 6, Junio 1970. pp. 377-387.

(3). C. J. Date, An Introduction to Data Base Systems. Addison-Wesley 1975.

(4). E. F. Codd, Further Normalization of the Data Base Relational Model. Courant Computer Science Symposia 6, Data Base Systems. Nueva York, 24-25 Mayo, 1971, pp. 33-64, Prentice-Hall, Nueva York.

(5). C. Beeri, P. A . Bernstein, Computational Problems Related to the Design of Normal Form Relational Schemas. ACM Trans. on Database Systems, Vol. 4,No. 1, March 1979.

(6). P. A . Bernstein, Synthesizing Third Normal Form Relations from Functional Dependencies. ACM Trans. on Database Systems, Vol. 1,No. 4, December 1976.

(7). R. Fagin, Multivalued Dependencies and a New Normal Form for Relational Databases. Research Report , IBM Research Laboratory, San Jose, California, Febrero 1977.

(8). H. K. Wong, N. C. Shu, An Approach to Relational Database Schema Design. Research Report RJ2688, IBM Research Laboratory, San Jose, California, Febrero 1980.

(9). W. W. Armstrong, Dependency structures of data base relationships. Information Processing 74,North-Holland Pub. Co., Amsterdam, 1974.