DISEÑO DE BASES DE DATOS RELACIONALES

El objetivo del diseño de las bases de datos relacionales es la generación de un conjunto de esquemas relacionales que nos permita almacenar la información sin redundancias innecesarias, pero que también nos permita recuperar fácilmente esa información.

Bibliografía:

Fundamentos de Sistemas de Bases de Datos. Ramez Elmasri, Shamkant Navathe, 5ta edición.

Fundamentos de Bases de Datos. Silberschatz, Korth, Sudarshan, 4ta edición.

SUPERCLAVE

Esquema de relación R = {A1, A2, ... ,An}

Superclave: conjunto de atributos S ⊆ R con la propiedad de que no habrá un par de tuplas t1 y t2 tal que t1[S] = t2[S].

Una clave K es una superclave con la propiedad adicional de que la eliminación de cualquier atributo de K provocará que K deje de ser una superclave.

La diferencia entre una clave y una superclave es que la clave tiene que ser mínima, es decir, si tenemos una clave K = {A1, A2, … ,Ak} de R, entonces K - {Ai} no es una clave de R para ningún Ai, 1 <= i <= k.

Por ejemplo {Dni} es una clave de EMPLEADO, mientras que {Dni}, {Dni, NombreE}, {Dni, NombreE, FechaNac} y cualquier otro conjunto de atributos que incluya Dni, son superclaves.

CLAVE CANDIDATA

Si un esquema de relación tiene más de una clave.
Cada una de ellas se denomina clave candidata.
Una de ellas se elige arbitrariamente como clave principal, mientras que el resto son claves secundarias.
Todo esquema de relación debe contar con una clave principal. {Dnj} es la única clave candidata de EMPLEADO, por lo que también será la clave principal.

Atributos Primos y No Primos

Un atributo del esquema de relación R recibe el nombre de **atributo primo** de R si es miembro de alguna de las claves candidatas de R.

Un **atributo es no primo** si no es miembro de ninguna clave candidata.

DIFICULTADES EN EL DISEÑO DE BASES DE DATOS RELACIONALES

Propiedades indeseables :

- Repetición de la información
- Imposibilidad de la representación de determinada información.
- Imposibilidad de recuperar información válida.

Supóngase que información de préstamos bancarios se guarda en una sola relación empréstito, que se define mediante el esquema de relación

Esquema-empréstito = (nombre-sucursal, ciudad-sucursal, activo, nombre-cliente, número-préstamo, importe)

¿Qué pasaría si hay que agregar un préstamo de un cliente de una sucursal existente?.

DIFICULTADES EN EL DISEÑO DE BASES DE DATOS RELACIONALES

¿Qué pasaría si hay que agregar un préstamo de un cliente de una sucursal existente?.

Se repite la ciudad y el activo de la misma sucursal.

No se optimiza el espacio y hace mas compleja la actualización.

DIFICULTADES EN EL DISEÑO DE BASES DE DATOS RELACIONALES

Cada sucursal bancaria tiene un valor único del activo, por lo que dado el nombre de una sucursal se puede identificar de manera única el valor del activo, esta restricción y haciendo una introducción básica a las dependencias funcionales se formaliza la restricción diciendo:

Se cumple la dependencia funcional

nombre-sucursal → activo

Cada sucursal puede conceder muchos préstamos por lo que, dado el nombre de una sucursal, no se puede determinar de manera única el número de un préstamo, de la misma manera que en el caso anterior se formaliza esta restricción diciendo:

No se cumple la dependencia funcional

nombre-sucursal → número-préstamo

DIFICULTADES EN EL DISEÑO DE BASES DE DATOS RELACIONALES

Otro problema del diseño Esquema-empréstito es que no se puede representar de manera directa la información relativa a cada sucursal (nombre-sucursal, ciudad-sucursal, activo) a menos que haya como mínimo un préstamo en esa sucursal. Esto se debe a que las tuplas de la relación empréstito exigen los valores de número_préstamo, importe y nombre-cliente.

Esquema-empréstito = **{ Nombre-Sucursal,Ciudad-Sucursal,Activo,Nombre-Cliente,Número-Préstamo,Importe }**

Una solución es introducir valores nulos. Los valores nulos resultan difíciles de manejar y si no se desea tratar con los valores nulos se puede crear la información sobre cada sucursal sólo después de que se formule la primera solicitud de préstamo en esa sucursal. Pero habrá que eliminar esa información cuando se hayan pagado todos los préstamos.

DEPENDENCIAS FUNCIONALES

Las dependencias funcionales desempeñan un papel fundamental en la diferenciación entre los buenos diseños de bases de datos y los malos. **Una dependencia funcional es un tipo de restricción que constituye una generalización del concepto de clave**.

Se definió el concepto de superclave de la manera siguiente. Sea R el esquema de una relación. El subconjunto K de R es una superclave de R si, en cualquier relación r(R), para todos los pares de tuplas t1 y t2 de r tales que t1 ≠ t2, t1 [K] ≠ t2 [K]. Es decir, ningún par de tuplas de una relación r(R) puede tener el mismo valor para el conjunto de atributos K.

DEPENDENCIAS FUNCIONALES

El concepto de dependencia funcional generaliza la noción de superclave. Considérese el esquema de una relación R y sean $\alpha \subseteq R$ y $\beta \subseteq R$. La dependencia funcional $\alpha \rightarrow \beta$ se cumple para el esquema R si, en cualquier relación r (R), para todos los pares de tuplas t1 y t2 de r tales que t1 [α] = t2 [α], también ocurre que t1 [β] = t2 [β].

Empleando la notación para la dependencia funcional,
se dice que K es una superclave de R si $K \rightarrow R$.
Es decir, K es una superclave si, siempre que t1 [K] = t2 [K], también se produce que t1 [R] = t2 [R] (es decir, t1 = t2).

DEPENDENCIAS FUNCIONALES

Considérese el esquema de una relación R y sean $\alpha \subseteq R$ y $\beta \subseteq R$.
La dependencia funcional $\alpha \rightarrow \beta$ se cumple para el esquema R si, en cualquier relación r (R), para todos los pares de tuplas t1 y t2 de r tales que t1 [α] = t2 [α], también ocurre que t1 [β] = t2 [β].

α	β
a	1
b	2
c	3
a	1
a	1
b	2
b	2

DEPENDENCIAS FUNCIONALES

Empleando la notación para la dependencia funcional,
se dice que K es una superclave de R si K → R
Es decir, K es una superclave si, siempre que t1 [K] = t2 [K], también se produce que t1 [R] = t2 [R] (es decir, t1 = t2).

K	Atta	Attb
a	1	M
b	2	N
c	3	L
d	4	W
e	5	X
f	6	Y
g	7	Z

DEPENDENCIAS FUNCIONALES

Considérese el esquema
Esquema-préstamo = (número-préstamo, nombre-sucursal, nombre-cliente, importe)
El conjunto de dependencias funcionales que se espera se cumplan en este esquema de relación son

préstamo → importe
número-préstamo → nombre-sucursal

Sin embargo, no se espera que se cumpla la dependencia funcional

número-préstamo → nombre-cliente

¿Qué significa que no se cumple la dependencia número-préstamo → nombre-cliente ?

DEPENDENCIAS FUNCIONALES

Las dependencias funcionales se utilizarán:

Para probar las relaciones r y ver si son legales según un conjunto dado de dependencias funcionales F.

Si una relación r es legal según el conjunto F de dependencias funcionales, se dice que r satisface F.

Si se desea restringir a las relaciones del esquema R que satisfagan el conjunto F de dependencias funcionales, se dice que F se cumple en R.

DEPENDENCIAS FUNCIONALES

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_2	c_1	d_2
a_2	b_2	c_2	d_2
a_2	b_3	c_2	d_3
a_3	b_3	c_2	d_4

Obsérvese que se satisface A → C. Hay dos tuplas que tienen un valor para A de a1. Estas tuplas tienen el mismo valor para C, por ejemplo, c1. De manera parecida, las dos tuplas con un valor para A de a2 tienen el mismo valor para C, c2. No hay más pares de tuplas diferentes que tengan el mismo valor para A. Sin embargo, la dependencia funcional C → A no se satisface. Para verlo, considérense las tuplas t1 = (a2, b3, c2, d3) y t2 = (a3, b3, c2, d4). Estas dos tuplas tienen los mismos valores para C, c2, pero tienen valores diferentes para A, a2 y a3, respectivamente. Por tanto, se ha hallado un par de tuplas t1 y t2 tales que t1 [C] = t2 [C], pero t1 [A] ≠ t2 [A].

r satisface muchas otras dependencias funcionales, incluida, por ejemplo, la dependencia funcional AB → D. Obsérvese que se utiliza AB como abreviatura de {A,B}, para adecuarnos al uso estándar. Obsérvese que no hay ningún par de tuplas diferentes t1 y t2 tales que t1 [AB] = t2 [AB]. Por tanto, si t1 [AB] = t2 [AB], debe ser que t1 = t2 y, por tanto, t1 [D] = t2 [D]. Así, r satisface AB → D.

DEPENDENCIAS FUNCIONALES

TRIVIALES

Se dice que algunas dependencias funcionales son triviales porque las satisfacen todas las relaciones. Por ejemplo, $A \rightarrow A$ la satisfacen todas las relaciones que impliquen al atributo A. La lectura literal de la definición de dependencia funcional deja ver que, para todas las tuplas t1 y t2 tales que t1 [A] = t2 [A], se cumple que t1 [A] = t2 [A]. De manera parecida, $AB \rightarrow A$ la satisfacen todas las relaciones que impliquen al atributo A. En general, una dependencia funcional de la forma $\alpha \rightarrow \beta$ es trivial si $\beta \subseteq \alpha$.

Cierre de un conjunto de dependencias funcionales

No es suficiente considerar el conjunto dado de dependencias funcionales. También hay que considerar todas las dependencias funcionales que se cumplen. Dado un conjunto F de dependencias funcionales, se puede probar que se cumplen otras dependencias funcionales. Se dice que hay dependencias funcionales que están «implicadas lógicamente» por F. De manera más formal, dado un esquema relacional R, una dependencia funcional f de R está implicada lógicamente por un conjunto de dependencias funcionales F de R si cada ejemplar de la relación r(R) que satisface F satisface también f.

Cierre de un conjunto de dependencias funcionales

Supóngase que se tiene un esquema de relación R = (A, B, C, G, H, I) y el conjunto de dependencias funcionales

$A \rightarrow B$
$A \rightarrow C$
$CG \rightarrow H$
$CG \rightarrow I$
$B \rightarrow H$

La dependencia funcional

$A \rightarrow H$

está implicada lógicamente. Es decir, se puede demostrar que, siempre que el conjunto dado de dependencias funcionales se cumple en una relación, en la relación también se debe cumplir

$A \rightarrow H$. Supóngase que t1 y t2 son tuplas tales que t1 [A] = t2 [A] Como se tiene que $A \rightarrow B$, se deduce de la definición de dependencia funcional que t1 [B] = t2 [B] Entonces, como se tiene que $B \rightarrow H$, se deduce de la definición de dependencia funcional que t1 [H] = t2 [H] Por tanto, se ha demostrado que, siempre que t1 y t2 sean tuplas tales que t1 [A] = t2 [A], debe ocurrir que t1 [H] = t2 [H]. Pero ésa es exactamente la definición de $A \rightarrow H$.

Axiomas de Armstrong

Sea F un conjunto de dependencias funcionales. El cierre de F, denotado por F+ , es el conjunto de todas las dependencias funcionales implicadas lógicamente en F. Dado F, se puede calcular F+ directamente a partir de la definición formal de dependencia funcional. Si F fuera de gran tamaño, este proceso sería prolongado y difícil.

Los axiomas, o reglas de inferencia, proporcionan una técnica más sencilla para el razonamiento sobre las dependencias funcionales. Aplicando las tres reglas siguientes repetidamente, se puede hallar todo F+ , dado F. Este conjunto de reglas se denomina axiomas de Armstrong en honor de la persona que las propuso por primera vez.

Reglas de Armstrong

• **Regla de la reflexividad**. Si α es un conjunto de atributos y β ⊆ α, entonces se cumple que α → β.

α = { A,B,C,D,E } y β = { A,B,C } entonces α → β.
α = { A,B,C,D,E } y β = { A,B,C,D,E } entonces α → β. DF TRIVIAL

• **Regla de la aumentatividad**. Si se cumple que α → β y γ es un conjunto de atributos, entonces se cumple que γα → γβ.

α = { A,B,C,D,E } y β = { A,B,C } entonces α → β.
γ = { X,Y } y α = { A,B,C,D,E,X,Y } y β = { A,B,C,X,Y } entonces α → β.

• **Regla de la transitividad**. Si se cumple que α → β y también se cumple que β → γ, entonces se cumple que α → γ.

α = { A,B,C,D,E } y β = { A,B,C } entonces α → β.
γ = { A,B } entonces β → γ entonces α → γ

Reglas de Armstrong

Aunque los axiomas de Armstrong son completos, resulta difícil utilizarlos directamente para el cálculo de F+ . Para simplificar más las cosas se relacionan unas reglas adicionales.

- Regla de la unión.
  Si se cumple que $\alpha \rightarrow \beta$ y que $\alpha \rightarrow \gamma$, entonces se cumple que $\alpha \rightarrow \beta\gamma$.

- Regla de la descomposición.
  Si se cumple que $\alpha \rightarrow \beta\gamma$, entonces se cumple que $\alpha \rightarrow \beta$ y que $\alpha \rightarrow \gamma$.

- Regla de la pseudotransitividad.
  Si se cumple que $\alpha \rightarrow \beta$ y que $\gamma\beta \rightarrow \delta$, entonces se cumple que $\alpha\gamma \rightarrow \delta$.

Axiomas de Armstrong

Apliquemos las reglas al esquema R = (A, B, C, G, H, I) y el conjunto F de dependencias funcionales {A→ B, A→ C, CG → H, CG → I, B → H}. A continuación se relacionan varios miembros de F+ :

• A → H. Dado que se cumplen A → B y B → H, se aplica la regla de transitividad. Obsérvese que resultaba mucho más sencillo emplear los axiomas de Armstrong para demostrar que se cumple que A → H que deducirlo directamente a partir de las definiciones, como se ha hecho anteriormente en este apartado.

• CG → HI . Dado que CG → H y CG → I , la regla de unión implica que CG → HI .

• AG → I. Dado que A → C y CG → I, la regla de pseudotransitividad implica que se cumple que AG → I.

Se utiliza la regla de aumentatividad en A → C para inferir que AG → CG. Aplicando la regla de transitividad a esta dependencia y CG → I, se infiere que AG → I

Procedimiento para calcular F + .

F+ = F

repeat for each (dependencia funcional f de F+)

aplicar las reglas de reflexividad y de aumentatividad a f

añadir las dependencias funcionales resultantes a F+

for each (pareja de dependencias funcionales f1 y f2 de F+)

if (f1 y f2 pueden combinarse mediante la transitividad)

añadir la dependencia funcional resultante a F+

until (F+ no cambie más)

Cierre de un conjunto de atributos

Para comprobar si un conjunto **α** es una superclave hay que diseñar un algoritmo para el cálculo del conjunto de atributos determinados funcionalmente por **α**. Una manera de hacerlo es calcular F+, tomar todas las dependencias funcionales con **α** como término de la izquierda y tomar la unión de los términos de la derecha de todas esas dependencias. Sin embargo, hacer esto puede resultar costoso, ya que F+ puede ser de gran tamaño. El algoritmo eficiente para calcular el conjunto de atributos determinados funcionalmente por **α** resulta útil para comprobar si **α** es una superclave. Sea **α** un conjunto de atributos. Al conjunto de todos los atributos determinados funcionalmente por **α** bajo un conjunto F de dependencias funcionales se le denomina cierre de **α bajo F ,** se denota mediante **α+**.

Algoritmo para el cálculo de α+, el cierre de α bajo F.

```
α+ Calcular ( Atributos α , DFs F )
{
    resultadoAnterior = {} ;
    resultado = α;
    while (resultadoAnterior != resultado)
        {
            resultadoAnterior = resultado ;
            for each (dependencia funcional β → γ in F)
                {
                    if (β ⊆ resultado)
                        resultado = resultado ∪ γ;
                }
        }
    α+ = resultado ;
}
```

La entrada es un conjunto F de dependencias funcionales y el conjunto α de atributos.
La salida se almacena en la variable resultado

Ilustración del algoritmo para el cálculo de α+

Para ilustrar el modo en que trabaja el algoritmo se utilizará para calcular (AG)+ con las dependencias funcionales A → B, A → C, CG → H, CG → I. Se comienza con resultado = AG. La primera vez que se ejecuta el bucle while para comprobar cada dependencia funcional se halla que

- A → B hace que se incluya B en resultado.
  Para ver este hecho, se observa que A → B se halla en F y A ⊆ resultado (que es AG), por lo que resultado := resultado ∪ B.
- A → C hace que resultado se transforme en ABCG.
- CG → H hace que resultado se transforme en ABCGH.
- CG → I hace que resultado se transforme en ABCGHI.

El algoritmo termina cuando ya no se añaden atributos nuevos a resultado.

Ilustración del algoritmo para el cálculo de α+

El primer paso es correcto, ya que $\alpha \rightarrow \alpha$ se cumple siempre (por la regla de reflexividad). Se asegura que, para cualquier subconjunto β de resultado, $\alpha \rightarrow \beta$. Dado que se inicia el bucle while con $\alpha \rightarrow$ resultado como cierto, sólo se puede añadir γ a resultado si $\beta \subseteq$ resultado y $\beta \rightarrow \gamma$. Pero, entonces, resultado $\rightarrow \beta$ por la regla de reflexividad, por lo que $\alpha \rightarrow \beta$ por transitividad. Otra aplicación de la transitividad demuestra que $\alpha \rightarrow \gamma$ (utilizando $\alpha \rightarrow \beta$ y $\beta \rightarrow \gamma$). La regla de la unión implica que $\alpha \rightarrow$ resultado $\cup \gamma$, por lo que α determina funcionalmente cualquier resultado nuevo generado en el bucle while. Por tanto, cualquier atributo devuelto por el algoritmo se halla en $\alpha+$. Resulta sencillo ver que el algoritmo halla todo $\alpha+$. Si hay un atributo de $\alpha+$ que no se halle todavía en resultado, debe haber una dependencia funcional $\beta \rightarrow \gamma$ para la que $\beta \subseteq$ resultado y, como mínimo, un atributo de γ no se halla en resultado. Resulta que, en el peor de los casos, este algoritmo puede tardar un tiempo proporcional al cuadrado del tamaño de F. Hay un algoritmo más rápido (aunque ligeramente más complejo) que se ejecuta en un tiempo proporcional al tamaño de F.

Hay varios usos para el algoritmo de cierre de atributos:

• Comprobar si α es una superclave, se calcula α+ y se comprueba si α+ contiene todos los atributos de R.

• Se puede comprobar si se cumple la dependencia funcional α → β (o, en otras palabras, si se halla en F+), comprobando si β ⊆ α+ . Es decir, se calcula α+ empleando el cierre de los atributos y luego se comprueba si contiene a β.

• Ofrece una manera alternativa de calcular F+ : para cada γ ⊆ R se halla el cierre γ+.

para cada S ⊆ γ+ , se genera una dependencia funcional γ → S.

DESCOMPOSICIÓN

Esquema-préstamo = (nombre-sucursal, ciudad-sucursal, activo, nombre-cliente, número-préstamo, importe)

Proyecciones de *préstamo*

sucursal-cliente = Π (***préstamo***)
nombre-sucursal, ciudad-sucursal, activo, nombre-cliente

cliente-préstamo = Π (***préstamo***)
nombre-cliente, número-préstamo, importe

Relaciones resultantes sucursal-cliente y cliente-préstamo

Esquema-sucursal-cliente = (nombre-sucursal, ciudad-sucursal, activo, nombre-cliente)

Esquema-cliente-préstamo = (nombre-cliente, número-préstamo, importe)

Reconstruir la relación préstamo

Hay casos en los que hace falta reconstruir la relación préstamo. Se desea hallar todas las sucursales que tienen préstamos con importes inferiores a 1000 $. Ninguna relación de la base de datos alternativa contiene esos datos. Hay que reconstruir la relación préstamo.

Parece que se puede hacer escribiendo

sucursal-cliente |X| cliente-préstamo

nombre-sucursal	*ciudad-sucursal*	*activo*	*nombre-cliente*
Centro	Arganzuela	9.000.000	Santos
Moralzarzal	La Granja	2.100.000	Gómez
Navacerrada	Aluche	1.700.000	López
Centro	Arganzuela	9.000.000	Sotoca
Becerril	Aluche	400.000	Santos
Collado Mediano	Aluche	8.000.000	Abril
Navas de la Asunción	Alcalá de Henares	300.000	Valdivieso
Segovia	Cerceda	3.700.000	López
Centro	Arganzuela	9.000.000	González
Navacerrada	Aluche	1.700.000	Rodríguez
Galapagar	Arganzuela	7.100.000	Amo

FIGURA 7.9. La relación *sucursal-cliente*.

nombre-cliente	*número-préstamo*	*importe*
Santos	P-17	1.000
Gómez	P-23	2.000
López	P-15	1.500
Sotoca	P-14	1.500
Santos	P-93	500
Abril	P-11	900
Valdivieso	P-29	1.200
López	P-16	1.300
González	P-18	2.000
Rodríguez	P-25	2.500
Amo	P-10	2.200

FIGURA 7.10. La relación *cliente-préstamo*.

sucursal-cliente |X| cliente-préstamo

nombre-sucursal	*ciudad-sucursal*	*activo*	*nombre-cliente*	*número-préstamo*	*importe*
Centro	Arganzuela	9.000.000	Santos	P-17	1.000
Centro	Arganzuela	9.000.000	Santos	P-93	500
Moralzarzal	La Granja	2.100.000	Gómez	P-23	2.000
Navacerrada	Aluche	1.700.000	López	P-15	1.500
Navacerrada	Aluche	1.700.000	López	P-16	1.300
Centro	Arganzuela	9.000.000	Sotoca	P-14	1.500
Becerril	Aluche	400.000	Santos	P-17	1.000
Becerril	Aluche	400.000	Santos	P-93	500
Collado Mediano	Aluche	8.000.000	Abril	P-11	900
Navas de la Asunción	Alcalá de Henares	300.000	Valdivieso	P-29	1.200
Segovia	Cerceda	3.700.000	López	P-15	1.500
Segovia	Cerceda	3.700.000	López	P-16	1.300
Centro	Arganzuela	9.000.000	González	P-18	2.000
Navacerrada	Aluche	1.700.000	Rodríguez	P-25	2.500
Galapagar	Arganzuela	7.100.000	Amo	P-10	2.200

FIGURA 7.11. La relación *sucursal-cliente* ⋈ *cliente-préstamo*.

nombre-sucursal	*ciudad-sucursal*	*activo*	*nombre-cliente*	*número-préstamo*	*importe*
Centro	Arganzuela	9.000.000	Santos	P-17	1.000
Moralzarzal	La Granja	2.100.000	Gómez	P-23	2.000
Navacerrada	Aluche	1.700.000	López	P-15	1.500
Centro	Arganzuela	9.000.000	Sotoca	P-14	1.500
Becerril	Aluche	400.000	Santos	P-93	500
Collado Mediano	Aluche	8.000.000	Abril	P-11	900
Navas de la Asunción	Alcalá de Henares	300.000	Valdivieso	P-29	1.200
Segovia	Cerceda	3.700.000	López	P-16	1.300
Centro	Arganzuela	9.000.000	González	P-18	2.000
Navacerrada	Aluche	1.700.000	Rodríguez	P-25	2.500
Galapagar	Arganzuela	7.100.000	Amo	P-10	2.200

FIGURA 7.1. Relación *empréstito* de ejemplo.

Considérese la consulta «Hallar todas las sucursales que han realizado un préstamo por un importe inferior a 1000$». Se observa que las únicas sucursales con créditos con importe inferior a 1000$ son Becerril y Collado Mediano. Sin embargo, al aplicar la expresión

$$\Pi_{\text{nombre-sucursal}}(\sigma_{\text{importe} < 1000}(\text{sucursal-cliente} \;|X|\; \text{cliente-préstamo}))$$

obtenemos los nombres de tres sucursales: Becerril, Collado Mediano y Centro.

sucursal-cliente |X| cliente-préstamo

nombre-sucursal	*ciudad-sucursal*	*activo*	*nombre-cliente*	*número-préstamo*	*importe*
Centro	Arganzuela	9.000.000	Santos	P-17	1.000
Centro	Arganzuela	9.000.000	Santos	P-93	500
Moralzarzal	La Granja	2.100.000	Gómez	P-23	2.000
Navacerrada	Aluche	1.700.000	López	P-15	1.500
Navacerrada	Aluche	1.700.000	López	P-16	1.300
Centro	Arganzuela	9.000.000	Sotoca	P-14	1.500
Becerril	Aluche	400.000	Santos	P-17	1.000
Becerril	Aluche	400.000	Santos	P-93	500
Collado Mediano	Aluche	8.000.000	Abril	P-11	900
Navas de la Asunción	Alcalá de Henares	300.000	Valdivieso	P-29	1.200
Segovia	Cerceda	3.700.000	López	P-15	1.500
Segovia	Cerceda	3.700.000	López	P-16	1.300
Centro	Arganzuela	9.000.000	González	P-18	2.000
Navacerrada	Aluche	1.700.000	Rodríguez	P-25	2.500
Galapagar	Arganzuela	7.100.000	Amo	P-10	2.200

FIGURA 7.11. La relación *sucursal-cliente* ⋈ *cliente-préstamo*.

Descomposición con Pérdida

Debido a esta pérdida de información se dice que la descomposición de Esquema-préstamo en Esquema-sucursal-cliente y Esquema-cliente-préstamo es una descomposición con pérdida, o una descomposición de reunión con pérdida. Una descomposición que no es una descomposición con pérdida es una descomposición de reunión sin pérdida. Queda claro con este ejemplo que una descomposición de reunión con pérdida supone, en general, un mal diseño de base de datos.

Para tener una descomposición de reunión sin pérdida hay que imponer restricciones en el conjunto de las relaciones posibles. Si se realiza la descomposición de Esquema-préstamo en Esquema-sucursal y Esquema-cliente-préstamo es sin pérdida porque se cumple la dependencia funcional nombre-sucursal → ciudad-sucursal , activo en Esquema-sucursal.

¿ Porqué la descomposición es una descomposición con pérdida ?

Hay un atributo en común (nombre-cliente) entre Esquema-cliente-sucursal y Esquema-cliente-préstamo:

Esquema-sucursal-cliente ∩ ***Esquema-cliente-préstamo = {nombre-cliente}***

Considérese otro diseño alternativo en el que se descompone Esquema-préstamo en los dos esquemas siguientes:

Esquema-sucursal = (nombre-sucursal, ciudad-sucursal, activo)

nombre-sucursal → ciudad-sucursal , activo

Esquema-cliente-préstamo = (nombre-sucursal, nombre-cliente, número-préstamo, importe)

Hay un atributo en común entre estos dos esquemas:

Esquema-sucursal ∩ ***Esquema-cliente-préstamo = {nombre-sucursal}***

¿ Porqué la descomposición es una descomposición sin pérdida ?

La diferencia entre este ejemplo y el anterior es que la ciudad-sucursal y el activo de una sucursal es el mismo, independientemente del cliente al que se haga referencia, mientras que la sucursal prestamista sí depende del cliente al que se haga referencia.
Para un valor dado de **nombre-sucursal**, hay exactamente un valor de **activo** y un valor de **ciudad-sucursal**, mientras que no se puede hacer esa afirmación para **nombre-cliente**.
Es decir, se cumple la dependencia funcional:
nombre-sucursal → activo, ciudad-sucursal

Y

nombre-cliente no determina funcionalmente a número-préstamo

PROPIEDADES DESEABLES DE LA DESCOMPOSICIÓN

Se puede utilizar un conjunto dado de dependencias funcionales para diseñar una base de datos relacional en la que no se halle presente la mayor parte de las propiedades no deseables. Por ejemplo:

- Repetición de la información.
- Imposibilidad de la representación de determinada información.

Cuando se diseñan estos sistemas puede hacerse necesaria la descomposición de una relación en varias relaciones de menor tamaño.

PROPIEDADES DESEABLES DE LA DESCOMPOSICIÓN
Ejemplo

Esquema-préstamo = (nombre-sucursal, ciudad-sucursal, activo, nombre-cliente, número-préstamo, importe)

El conjunto F de dependencias funcionales que se exige que se cumplan en Esquema-préstamo es

nombre-sucursal → ciudad-sucursal, activo
número-préstamo → importe, nombre-sucursal
Número-préstamo → nombre-cliente

Se descompone Esquema-préstamo en las tres relaciones siguientes:

Esquema-sucursal = (nombre-sucursal, ciudad-sucursal, activo)
Esquema-préstamo=(número-préstamo, nombre-sucursal, importe)
Esquema-prestatario = (número-préstamo, nombre-cliente)

Esta descomposición tiene varias propiedades deseables.

Descomposición de reunión sin pérdida

Criterio para determinar si una descomposición es una descomposición con pérdida.

Sea R un esquema de relación, y sea F un conjunto de dependencias funcionales de R.

R1 y R2 son descomposición de R.

Esta descomposición es una descomposición de reunión sin pérdida de R
Si al menos una de las siguientes dependencias se halla en F+ :

- $R1 \cap R2 \rightarrow R1$
- $R1 \cap R2 \rightarrow R2$

En otras palabras, si $R1 \cap R2$ forma una superclave de R1 o de R2, la descomposición de R es una descomposición de reunión sin pérdida.

Descomposición de reunión sin pérdida

Secuencia de pasos que generan la descomposición

Se descompone Esquema-préstamo en dos esquemas:

Esquema-sucursal = (nombre-sucursal, ciudad-sucursal, activo)
Esquema-cliente-préstamo=(nombre-sucursal,nombre-cliente,número-préstamo, importe)

Dado que **nombre-sucursal → ciudad-sucursal, activo**
la regla de la aumentatividad para las dependencias funcionales implica que
nombre-sucursal → nombre-sucursal, ciudad-sucursal, activo

Como **Esquema-sucursal ∩ Esquema-cliente-préstamo = {nombre-sucursal},** se concluye que la descomposición inicial es una descomposición de reunión sin pérdida.

A continuación se descompone Esquema-cliente-préstamo en
Esquema-préstamo = (número-préstamo, nombre-sucursal, importe)
Esquema-prestatario = (nombre-cliente, número-préstamo)

Este paso da lugar a una descomposición de reunión sin pérdida ya que número-préstamo es un atributo común y número-préstamo → importe, nombre-sucursal.

Conservación de las dependencias

Hay otro objetivo en el diseño de las bases de datos relacionales: la conservación de las dependencias. Cuando se lleva a cabo una actualización de la base de datos el sistema debe poder comprobar que la actualización no crea ninguna relación ilegal, es decir, una relación que no satisface todas las dependencias funcionales dadas.

La definición de restricción utiliza todas las dependencias de F+ , no sólo las de F. Por ejemplo, supóngase que se tiene F = { A → B, B → C } y que se tiene una descomposición en AC y AB. La restricción de F a AC es, entonces, A → C, ya que A → C se halla en F+ , aunque no se halle en F.

Conservación de las dependencias

La idea es comprobar cada dependencia funcional $\alpha \rightarrow \beta$ de F empleando una forma modificada del cierre de los atributos para ver si la descomposición la conserva. Se aplica el siguiente procedimiento a cada $\alpha \rightarrow \beta$ de F.

resultado = α
while (cambios en resultado) do
    for each Ri de la descomposición
        t = (resultado $\cap$ Ri) + $\cap$ Ri
        resultado = resultado $\cup$ t

El cierre de los atributos está tomado con respecto a las dependencias funcionales de F. Si resultado contiene todos los atributos de β, se conserva la dependencia funcional $\alpha \rightarrow \beta$.
La descomposición conserva las dependencias si y sólo si se conservan todas las dependencias de F.

Mediante las dependencias funcionales se pueden definir varias formas normales que representan «buenos» diseños de bases de datos.

Las tres primeras **formas normales**: 1FN, 2FN y 3FN fueron propuestas por Codd (1972) como una secuencia para alcanzar el estado deseable de relaciones en 3FN

Primera forma normal

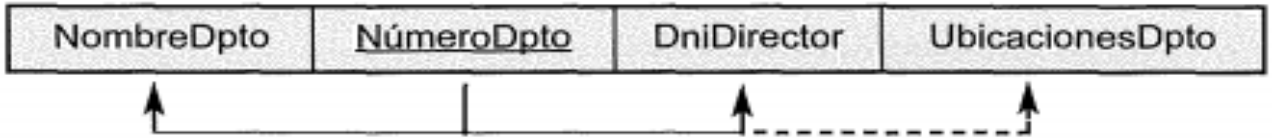
La primera forma normal (1FN) está considerada como una parte de la definición formal de una relación en el modelo relacional básico; históricamente, fue definida para prohibir los atributos multivalor, los atributos compuestos y sus combinaciones. Afirma que el dominio de un atributo sólo debe incluir valores atómicos (simples, indivisibles) y que el valor de cualquier atributo en una tupla debe ser un valor simple del dominio de ese atributo. Por tanto, 1FN prohíbe tener un conjunto de valores. Los únicos valores de atributo permitidos por 1FN son los atómicos (o indivisibles).

Primera forma normal

Figura 10.8. Normalización en 1FN. (a) Un esquema de relación que no está en 1FN. (b) Ejemplo de un estado de relación DEPARTAMENTO. (c) Versión 1FN de la misma relación con redundancia.

(a)

DEPARTAMENTO



(b)

DEPARTAMENTO

NombreDpto	NúmeroDpto	DniDirector	UbicacionesDpto
Investigación	5	333445555	{Valencia, Sevilla, Madrid}
Administración	4	987654321	{Gijón}
Sede central	1	888665555	{Madrid}

(c)

DEPARTAMENTO

NombreDpto	NúmeroDpto	DniDirector	UbicaciónDpto
Investigación	5	333445555	Valencia
Investigación	5	333445555	Sevilla
Investigación	5	333445555	Madrid
Administración	4	987654321	Gijón
Sede central	1	888665555	Madrid

Segunda forma normal

La segunda forma normal (2FN) está basada en el concepto de dependencia funcional total. Una dependencia funcional X → Y es total si la eliminación de cualquier atributo A de X implica que la dependencia deje de ser válida, es decir, para cualquier atributo A ∈ X, (X - {A}) no determina funcionalmente a Y. Una dependencia funcional X → Y es parcial si al eliminarse algún atributo A ∈ X de X la dependencia sigue siendo válida, es decir, para algún A ∈ X, (X - {A}) → Y.

{Dní, NumProyecto} → Horas

Es una dependencia completa

ni Dni → Horas

ni NumProyecto → Horas

son válidas

Sin embargo, la dependencia {Dni, NumProyecto} → NombreE es parcial porque se cumple Dni → NombreE.

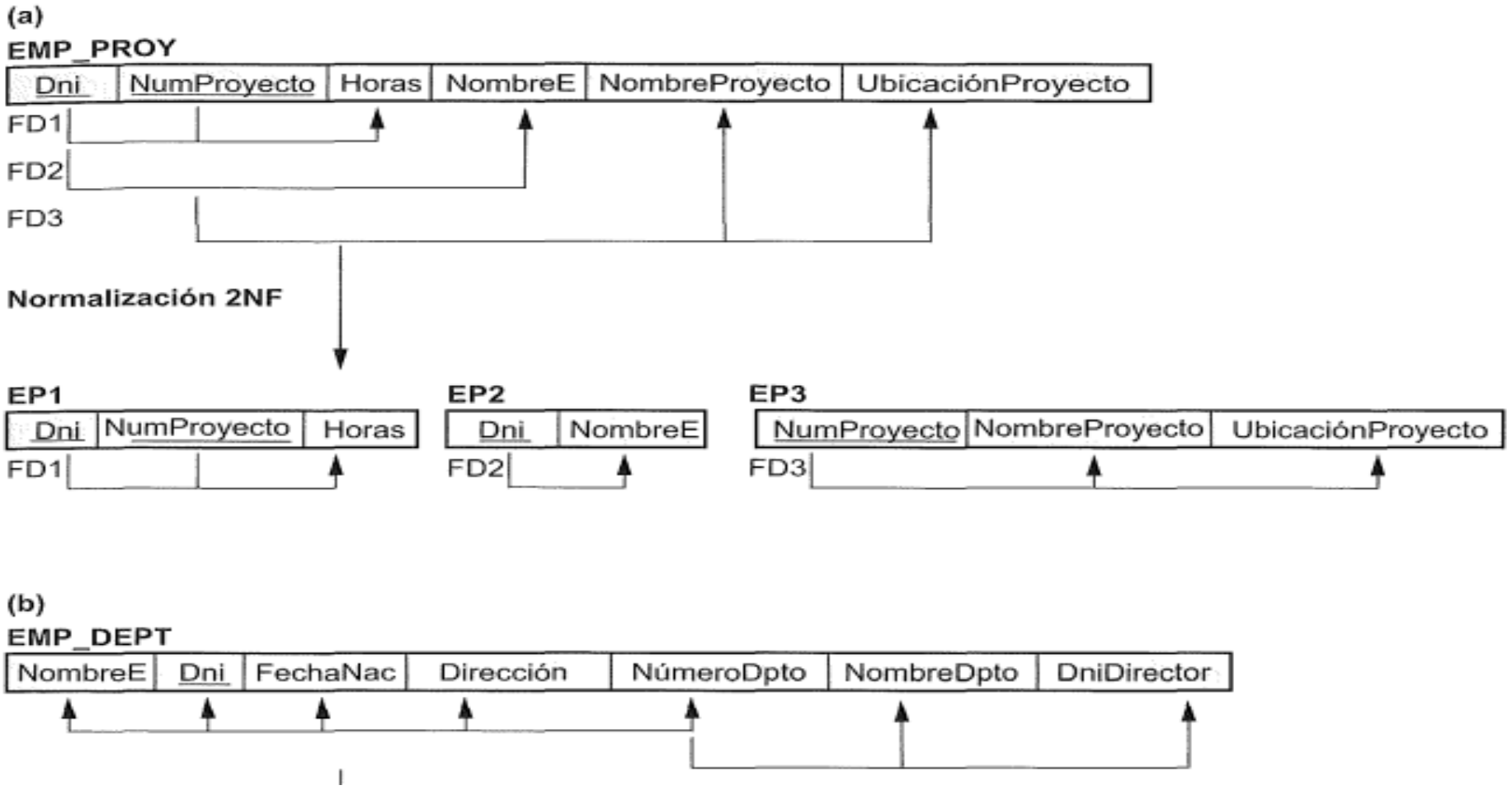
Segunda forma normal

Definición. Un esquema de relación R está en 2FN si todo atributo no primo A en R es una dependencia completa y funcionalmente dependiente de la clave principal de R.

Un **atributo es no primo** si no es miembro de ninguna clave candidata.

Segunda forma normal

La comprobación para 2FN implica la verificación de las dependencias funcionales cuyos atributos del lado izquierdo (determinante) forman parte de la clave principal. Si ésta contiene un único atributo, no es necesario aplicar la verificación.



TERCERA FORMA NORMAL

La tercera forma normal (3FN) se basa en el concepto de dependencia transitiva. Una dependencia funcional X → Y en un esquema de relación R es una dependencia transitiva si existe un subconjunto de atributos Z que no es clave candidata ni un subconjunto de ninguna clave de R, y se cumple tanto X → Z como Z → Y, esto implica que X → Y, corresponde al axioma de transitividad de Amstrong.

Definición. Un esquema de relación R está en tercera forma normal (3FN) si, siempre que una dependencia funcional no trivial X → A se cumple en R, ya sea que

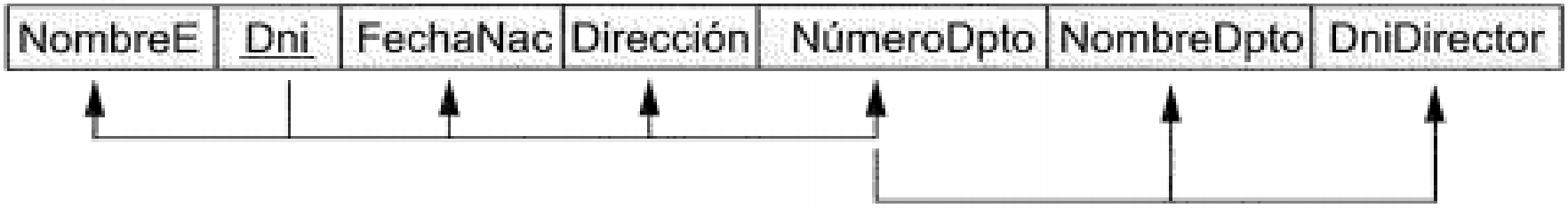
X es una superclave de R *o* A es un atributo primo de R.

Definición. Un esquema de relación R está en tercera forma normal (3FN) si esta en segunda forma normal y no hay dependencia funcional transitiva de ninguna clave, si X es clave de R entonces X → A se cumple en R no de forma transitiva.

TERCERA FORMA NORMAL

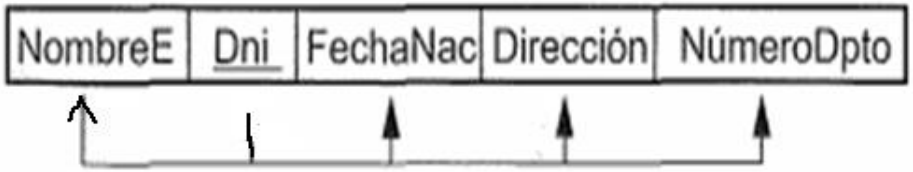
(a)

EMP_DEPT

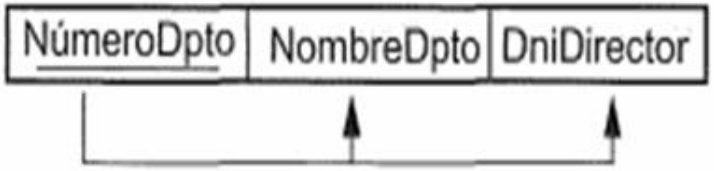


Normalización 3NF

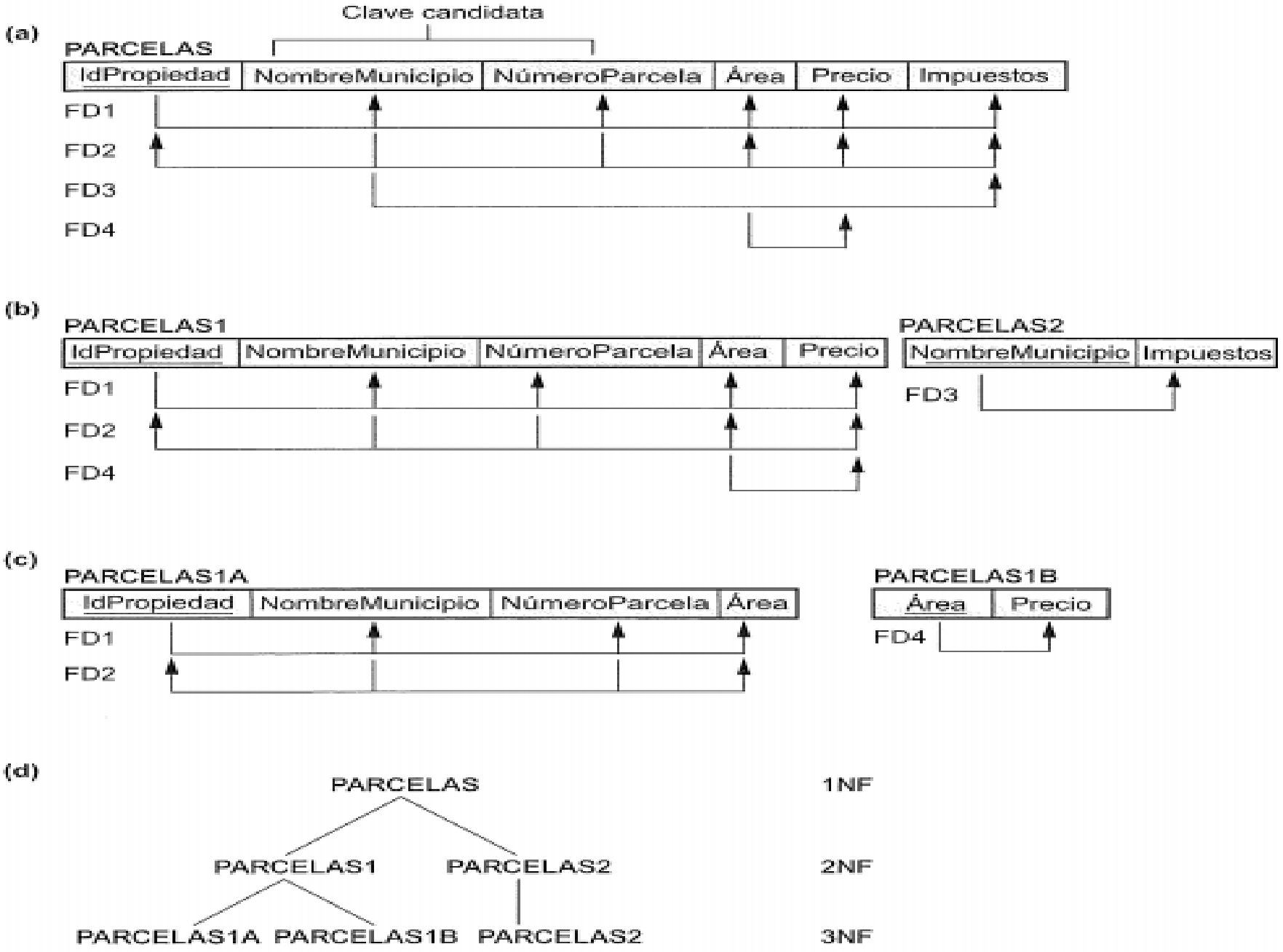
ED1



ED2



Intuitivamente podemos ver que cualquier dependencia funcional en la que el lado izquierdo es parte de la clave principal (subconjunto propio), o cualquier dependencia funcional en la que el lado izquierdo es un atributo no clave, implica una DF problemática. La normalización 2FN y 3FN elimina estas DFs descomponiendo la relación original en nuevas relaciones.


(a)
Clave candidata
PARCELAS
IdPropiedad
NombreMunicipio
NúmeroParcela
Área
Precio
Impuestos
FD1
FD2
FD3
FD4
(b)
PARCELAS1
IdPropiedad
NombreMunicipio
NúmeroParcela
Área
Precio
FD1
FD2
FD4
PARCELAS2
NombreMunicipio
Impuestos
FD3
(c)
PARCELAS1A
IdPropiedad
NombreMunicipio
NúmeroParcela
Área
FD1
FD2
PARCELAS1B
Área
Precio
FD4
(d)
PARCELAS
1NF
PARCELAS1
PARCELAS2
2NF
PARCELAS1A
PARCELAS1B
PARCELAS2
3NF

Forma normal	Prueba	Normalización
Primera (1 FN)	La relación no debe tener atributos multivalor.	Generar nuevas relaciones para cada atributo multivalor .
Segunda (2FN)	Para relaciones en las que la clave principal contiene varios atributos, un atributo no clave debe ser funcionalmente dependiente en forma completa de la clave principal.	Descomponer y configurar una nueva relación por cada clave parcial con su(s) atributo(s) dependiente(s). Asegurarse de mantener una relación con la clave principal original y cualquier atributo que sea completa y funcionalmente dependiente de ella.
Tercera (3FN)	La relación no debe tener un atributo no clave que esté funcionalmente determinado por otro atributo no clave (o por un conjunto de atributos no clave). No debe ser una dependencia transitiva de un atributo no clave de la clave principal.	Descomponer y configurar una relación que incluya el(los) atributo(s) no clave que determine(n) funcionalmente otro(s) atributo(s) no clave.

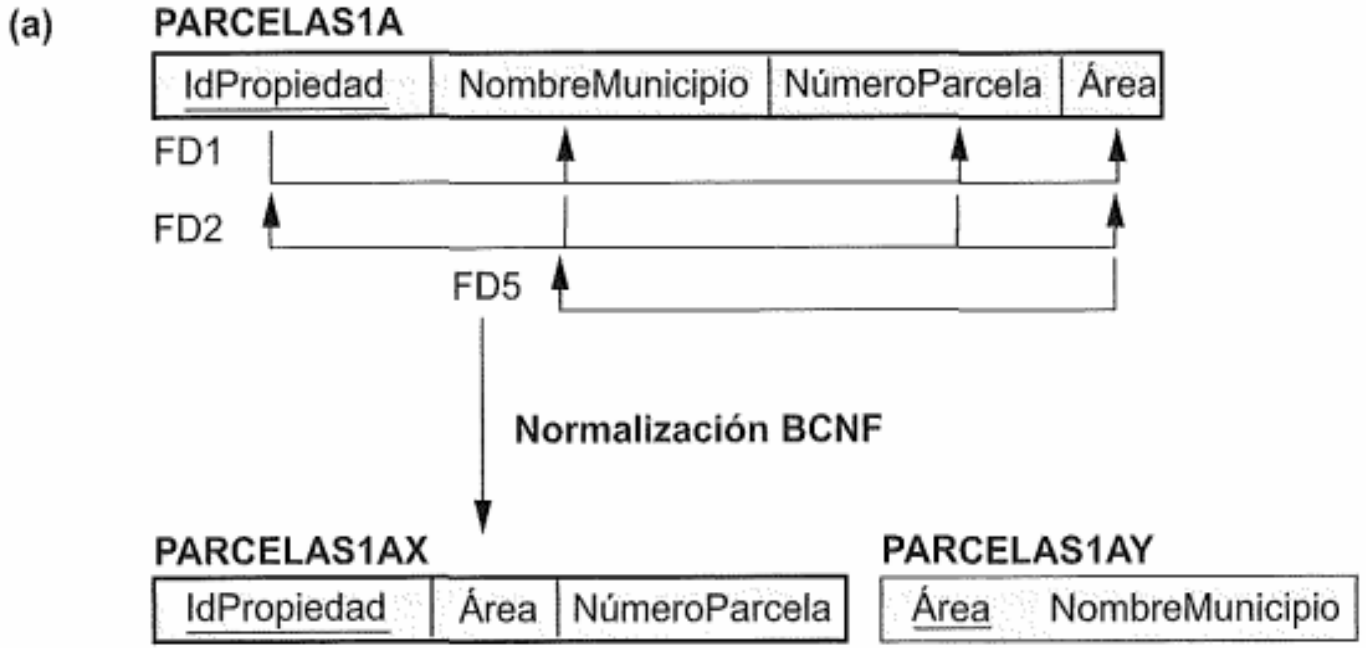
Forma normal de Boyce-Codd

La BCNF (Forma normal de Boyce-Codd) se propuso como una forma más simple de la 3FN, aunque es más estricta que ésta. Es decir, toda relación que esté en BCNF lo está también en 3FN; sin embargo, una relación 3FN no está necesariamente en BCNF.

Definición. Un esquema de relación R está en BCNF si siempre que una dependencia funcional no trivial X → A se cumple en R, entonces X es una superclave de R.

La definición formal de BCNF difiere ligeramente de la de 3FN. La única diferencia entre ellas es la condición (b) de 3FN, la cual permite que A sea primo, lo que no se consiente en BCNF.

Figura 10.12. Forma normal de Boyce-Codd. (a) Normalización BCNF de PARCELAS1A en la que se pierde la dependencia funcional FD2 en la descomposición. (b) Una relación esquemática con FDs; está en 3FN pero no en BCNF.



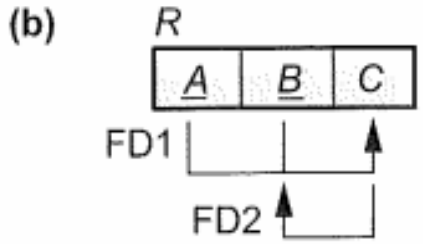


Figura 10.13. Una relación ENSEÑAR que está en 3FN pero no en BCNF.

Estudiante	Curso	Profesor
Campos	Bases de datos	Marcos
Pérez	Bases de datos	María
Pérez	Sistemas operativos	Amanda
Pérez	Teoría	Sergio
Ochoa	Bases de datos	Marcos
Ochoa	Sistemas operativos	Aurora
Morera	Bases de datos	Eduardo
Celaya	Bases de datos	María
Campos	Sistemas operativos	Amanda

Clave {Estudiante, Curso}

FD1: {Estudiante, Curso} → Profesor
FD2: Profesor → Curso

Forma normal de Boyce-Codd

Observe que {Estudiante, Curso} es una clave candidata para esta relación.
FD1: {Estudiante, Curso} → Profesor
FD2: Profesor → Curso

Por consiguiente, esta relación está en 3FN pero no en BCNF.
La descomposición de este esquema de relación en dos esquemas no es muy correcta porque puede dividirse en uno de los tres siguientes pares:

1. R1 = {Estudiante, Profesor} y R2 = {Estudiante, Curso}.
2. R1 = {Curso, Profesor} y R2 = {Curso, Estudiante}.
3. R1 = {Profesor, Curso} y R2 = {Profesor, Estudiante}.

Las tres descomposiciones pierden la dependencia funcional FD1. De ellas, la descomposición deseable es la número 3.
La cual produce dos relaciones en BCNF como éstas:
(Profesor, Curso) y (Profesor, Estudiante)
Observe que si designamos (Profesor, Estudiante) como clave principal de la relación ENSEÑAR, la DF Profesor → Curso provoca una dependencia parcial (no completamente funcional) de Curso en una parte de esta clave. Esta DF podría eliminarse como parte de una segunda normalización generando las dos relaciones de la tercer opción. Esto es un ejemplo de cómo alcanzar el diseño en BCNF a través de rutas de normalización alternativas.

FORMA NORMAL DE BOYCE–CODD

Una de las formas normales mas deseables que se pueden obtener es la forma normal de Boyce-Codd (FNBC).

Un esquema de relación R está en FNBC respecto a un conjunto de dependencias funcionales F si, para todas las dependencias funcionales de F+ de la forma $\alpha \rightarrow \beta$, donde $\alpha \subseteq R$ y $\beta \subseteq R$, se cumple al menos una de las siguientes condiciones:

- $\alpha \rightarrow \beta$ es una dependencia funcional trivial (es decir, $\beta \subseteq \alpha$)
- α es una superclave del esquema R.

Un diseño de base de datos está en FNBC si cada miembro del conjunto de esquemas de relación que constituye el diseño está en FNBC.

Están en FNBC

Considérense el siguiente esquema de relación y sus dependencias funcionales:

• Esquema-cliente = (nombre-cliente, calle-cliente, ciudad-cliente)
nombre-cliente (CC) → calle-cliente, ciudad-cliente

• Esquema-sucursal = (nombre-sucursal, activo, ciudad-sucursal)
nombre-sucursal (CC) → activo, ciudad-sucursal

Los esquemas están en FNBC. Obsérvese las claves candidatas de los dos esquemas.

Las únicas dependencias funcionales no triviales que se cumplen en los dos esquemas tienen a las **(CC)** a la izquierda de la DFs. Las DFs con **(CC)** en la parte izquierda no violan la definición de FNBC.

No está en FNBC

Considérense el siguiente esquema de relación y sus dependencias funcionales:

• Esquema-cliente-préstamo = (nombre-sucursal, nombre-cliente, número-préstamo, importe)
número-préstamo → importe, nombre-sucursal

Obsérvese que número-préstamo no es una superclave de Esquema-cliente-préstamo, ya que puede que haya un par de tuplas que representen a un solo préstamo concedido a dos personas.

número-préstamo no es una clave candidata. Sin embargo, la dependencia funcional número-préstamo → importe, nombre-sucursal es de tipo no trivial. Por lo tanto, Esquema-cliente-préstamo no satisface la definición de FNBC.

Se puede afirmar que Esquema-cliente-préstamo no está en una forma normal adecuada, ya que sufre del problema de repetición de información.

Están en FNBC

Considérese la descomposición de Esquema-cliente-préstamo en dos esquemas:

- Esquema-préstamo = (número-préstamo, nombre-sucursal, importe)
número-préstamo **(CC)**→ importe, nombre-sucursal

- Esquema-prestatario = (nombre-cliente, número-préstamo)
nombre-cliente **(CC)**→ número-préstamo

Las DFs con **(CC)** en la parte izquierda no violan la definición de FNBC.

Comprobación de una relación para ver si satisface FNBC

• Para comprobar si la dependencia no trivial $\alpha \rightarrow \beta$ provoca una violación de FNBC hay que calcular α^+ (el cierre de los atributos de α) y comprobar si el resultado incluye todos los atributos de R, es decir, si α^+ es una superclave de R.

• Para comprobar si el esquema de relación R se halla en FNBC basta con comprobar únicamente las dependencias del conjunto dado F en búsqueda de violaciones de FNBC, en lugar de comprobar todas las dependencias de F^+ .

Algoritmo de descomposición

Se aplicará el algoritmo de descomposición FNBC al esquema mal diseñado

Esquema-empréstito = (nombre-sucursal, ciudad-sucursal, activo, nombre-cliente, número-préstamo, importe)

El conjunto de dependencias funcionales que se exige que se cumplan en Esquema-empréstito es

nombre-sucursal → activo, ciudad-sucursal
número-préstamo → importe nombre-sucursal

Una clave candidata para este esquema es

{número-préstamo, nombre-cliente}.

Se puede aplicar el siguiente procedimiento al ejemplo Esquema-empréstito

1) La dependencia funcional **nombre-sucursal → activo , ciudad-sucursal**
se cumple en Esquema-empréstito, pero nombre-sucursal no es una superclave. Por tanto, Esquema-empréstito no está en FNBC

Se sustituye Esquema-empréstito por

Esquema-sucursal = (nombre-sucursal, ciudad-sucursal, activo)
Esquema-cliente-préstamo = (nombre-sucursal, nombre-cliente, número-préstamo, importe)

DFs originales
nombre-sucursal → activo, ciudad-sucursal
número-préstamo → importe nombre-sucursal

Las únicas dependencias funcionales no triviales que se cumplen en Esquema-sucursal incluyen a nombre-sucursal a la izquierda de la flecha. Como nombre-sucursal es una clave de Esquema-sucursal, la relación Esquema-sucursal está en FNBC.

Se puede aplicar el siguiente procedimiento al ejemplo Esquema-empréstito

2) La dependencia funcional
número-préstamo → importe, nombre-sucursal

se cumple en Esquema-cliente-préstamo = (nombre-sucursal, nombre-cliente, número-préstamo, importe) , pero número-préstamo no es una clave de Esquema-cliente-préstamo.

Se sustituye Esquema-cliente-préstamo por

Esquema-préstamo = (número-préstamo, nombre-sucursal, importe)
Esquema-prestatario = (nombre-cliente, número-préstamo)

Esquema-préstamo y Esquema-prestatario están en FNBC. Por tanto, la descomposición de Esquema-empréstito da lugar a tres esquemas de relación Esquema-sucursal, Esquema-préstamo y Esquema-prestatario, cada uno de los cuales está en FNBC

Conservación de las dependencias

No todas las descomposiciones FNBC conservan las dependencias. Por ejemplo, el esquema de relación

Esquema-asesor = (nombre-sucursal, nombre-cliente, nombre-asesor)

que indica que el cliente tiene un «asesor personal» en una sucursal determinada.
El conjunto F de dependencias funcionales es

nombre-asesor → nombre-sucursal
nombre-sucursal, nombre-cliente → nombre-asesor

NO ESTA EN FNBC ya que nombre-asesor no es una superclave.

Esquema-asesor
No está en FNBC, ya que **nombre-asesor no es una superclave**.

La descomposición FNBC:
Esquema -asesor-sucursal = (nombre-asesor, nombre-sucursal)
Esquema -cliente-asesor = (nombre-cliente, nombre-asesor)

Los esquemas descompuestos sólo conservan
nombre-asesor → nombre-sucursal (y las dependencias triviales)
pero el cierre de {nombre-asesor → nombre-sucursal}
no incluye **nombre-cliente, nombre-sucursal → nombre-asesor.**

La violación de esta dependencia no puede detectarse a menos que se calcule la reunión.

El ejemplo demuestra que no se pueden cumplir siempre los tres objetivos del diseño:

1. Reunión sin pérdida
2. FNBC
3. Conservación de las dependencias

Dependencias multivaluadas.

Una dependencia multivaluada es una sentencia que se escribe : X→ →Y y cuyo significado intuitivo es el siguiente : A cada valor de X se le asocia un conjunto de valores de Y independiente del contexto (si X e Y son subconjuntos de R , el contexto es Z=R- (X - Y)) . Quiere decir que la dependencia multivaluada asigna a cada valor de Dominio (X) un valor de P(Dominio (Y)) , y esta asignación no varía con el contexto . Es muy importante no confundir la existencia de una dependencia multivaluada con el hecho de que entre los dominios de X e Y pueda establecerse una correspondencia (1 : n). Tres definiciones equivalentes .

Cada una tiene su mayor o menor utilidad según el punto de vista bajo el que las DFs´Mul sean consideradas .

Dependencias multivaluadas.

Definición usando la notación de dependencia generalizada :

X	Y	Z
x	y	z
x	y’	z’
x	y	z’
x	y’	z

X→ →Y si la existencia de las tuplas **< x y z > < x y´z´ >** implica la existencia de **< x y z´ > <x y´z>** . En notación de dependencia generalizada, a las dos primeras se les llama **tuplas hipótesis** y a las dos últimas **tuplas conclusión**. La consistencia requiere de la aparición de las tuplas conclusión en cualquier instancia válida del esquema en la que aparecen las tuplas hipótesis.

Dependencias multivaluadas.

R = { Materia , Profesor , Bibliografía }

Materia →→ Profesor
Materia →→ Bibliografía

Materia	Profesor	Bibliografía
001	002	004
001	002	005
001	003	004
001	003	005
002	004	006
002	004	007
002	005	006
002	005	007

Cuarta forma normal

Considérese el ejemplo del esquema en la FNBC

R = (número_préstamo, id_cliente, calle_cliente, ciudad_cliente)

id_cliente $\rightarrow\!\!\rightarrow$ número_préstamo

Es una dependencia multivalorada no trivial, y que id_cliente no es superclave de R.

Aunque este esquema se halla en la FNBC, el diseño no es el ideal, ya que hay que repetir la información de número_préstamo para cada cliente. Se verá que se puede utilizar esta dependencia multivalorada para mejorar el diseño de la base de datos, realizando la descomposición del esquema en la cuarta forma normal. Un esquema de relación R está en la cuarta forma normal (4FN) con respecto a un conjunto D de dependencias funcionales multivaloradas si, para todas las dependencias multivaloradas de D+ de la forma $\alpha \rightarrow\!\!\rightarrow \beta$, donde $\alpha \subseteq R$ y $\beta \subseteq R$, se cumple, como mínimo, una de las condiciones siguientes

- $\alpha \rightarrow\!\!\rightarrow \beta$ es una dependencia multivalorada trivial.
- α es superclave del esquema R.

Descomposición en la 4FN

Dado R = (número_préstamo, id_cliente, calle_cliente, ciudad_cliente) se descubre que id_cliente →→ número_préstamo es una dependencia multivalorada no trivial, y que id_cliente no es superclave del esquema.

Se sustituye el esquema original por estos dos esquemas:

id_cliente_préstamo = (número_préstamo, id_cliente)
residencia_cliente = (id_cliente, calle_cliente, ciudad_cliente)

Estos esquemas, que se hallan en la 4FN, elimina la redundancia que se encontró anteriormente.

Cuarta forma normal

El diseño de una base de datos está en la 4FN si cada componente del conjunto de esquemas de relación que constituye el diseño se halla en la 4FN.

Téngase en cuenta que la definición de la 4FN sólo se diferencia de la definición de la FNBC en el empleo de las dependencias multivaloradas en lugar de las dependencias funcionales.

Todos los esquemas en la 4FN están en la FNBC. Para verlo hay que darse cuenta de que, si un esquema R no se halla en la FNBC, hay una dependencia funcional no trivial $\alpha \rightarrow \beta$ que se cumple en R en la que α no es superclave. Como $\alpha \rightarrow \beta$ implica $\alpha \rightarrow\rightarrow \beta$, R no puede estar en la 4FN

Cuarta forma normal

Sea R un esquema de relación y sean R1, R2, . . . , Rn una descomposición de R. Para comprobar si cada esquema de relación Ri de la descomposición se halla en la 4FN hay que averiguar las dependencias multivaloradas que se cumplen en cada Ri . Recuérdese que, para un conjunto F de dependencias funcionales, la restricción Fi de F a Ri son todas las dependencias funcionales de F + que sólo incluyen los atributos de Ri. Considérese ahora un conjunto D de dependencias funcionales multivaloradas. La restricción de D a Ri es el conjunto Di consistente en

1. Todas las dependencias funcionales de D+ que sólo incluyen atributos de Ri

2. Todas las dependencias multivaloradas de la forma $\alpha \rightarrow\rightarrow \beta \cap Ri$, donde $\alpha \subseteq Ri$ y $\alpha \rightarrow\rightarrow \beta$ está en D+

Ejercicio Formas Normales

Ejemplo: Dado el esquema de relación

sucursal_asesor = (id_cliente, id_empleado, nombre_sucursal, tipo)

UD: Cada empleado sólo puede trabajar en una sucursal, sólo puede haber un valor de nombre_sucursal asociado con cada valor de id_empleado: **id_empleado → nombre_sucursal**
Cada Cliente puede ser atendido por un empleado en cada sucursal, pero un empleado puede atender a muchos clientes: **id_cliente, nombre_sucursal → id_empleado,tipo.**

No obstante, no queda más remedio que repetir el nombre de la sucursal cada vez que un empleado participa en la relación sucursal_asesor. Es evidente que sucursal_asesor no se halla en la FNBC, ya que id_empleado no es superclave. De acuerdo con la regla para la descomposición en la FNBC, se obtiene:

(id_cliente, id_empleado, tipo)
(id_empleado, nombre_sucursal)

Este diseño, No puede hacer cumplir la restricción de que cada cliente tiene un asesor en cada sucursal. Esta restricción se expresa mediante la dependencia funcional id_cliente, nombre_sucursal → id_empleado

Muchas Gracias, hasta aquí llegamos hoy.