

Tema 6: Normalización

Contenido:

- 6 Normalización
 - 6.1. Introducción
 - 6.2. Redundancias y anomalías
 - 6.2.1. Redundancia de datos
 - 6.2.2. Anomalías de actualización
 - 6.2.3. Obtención de datos incorrectos
 - 6.3 Dependencias Funcionales
 - 6.3.1 Introducción y definiciones
 - 6.3.2 Cierre de un conjunto de dependencias funcionales
 - 6.3.3 Cierre de un conjunto de atributos
 - 6.3.4 Recubrimiento mínimo de un conjunto de dependencias funcionales
 - 6.4. Primera, segunda y tercera forma normal
 - 6.5. Forma normal de Boyce-Codd
 - 6.5.1. Preservación de dependencias en la descomposición
 - 6.5.2 Descomposición y reuniones no aditivas o sin pérdida
 - 6.5.3. Preservación de dependencias en la descomposición y reuniones no aditivas
 - 6.6. Otras formas normales
 - 6.7. Resumen
- Bibliografía

6.1 Introducción

Tal y como hemos visto para modelar un problema que pretendemos gestionar con un SGBD relacional debemos:

- Definir el diagrama E-R que represente el problema de la manera más fiel posible
- Pasar el diagrama E-R al esquema relacional. El resultado será un conjunto de tabla + un conjunto de restricciones. Las restricciones principales son:
 - o Claves primarias
 - o Restricciones de unicidad (para las claves candidatas no primarias)
 - o Claves ajenas
 - o Otras: check, domain, etc.

Puede parecer que con esto hemos terminado nuestra tarea, pero no es así; falta un punto fundamental: la *normalización* del esquema de bases de datos. Esta normalización es necesaria porque de otra forma se producirían anomalías que llevarán a una base de datos incoherente. Las anomalías se pueden controlar por programación, pero nuestro objetivo es que con las restricciones que indicamos al crear la tabla (en particular con las claves primarias, restricciones de unicidad y claves ajenas) el sistema se encargue de forma automática de evitar ciertas anomalías.

6.2 Redundancias y anomalías

Redundancia de datos

Un objetivo del diseño de bases de datos relacionales es agrupar atributos en relaciones de forma que se reduzca la redundancia de datos y así el espacio de almacenamiento necesario.

Ejemplo 1.

Los siguientes dos esquemas

Empleados(Id_empleado, NombreP, DirecciónP, Puesto, Salario, Centro)

Centros(NombreC, DirecciónC, Teléfono)

contienen la misma información que el siguiente:

Empleados_Centros(Id_empleado, NombreP, DirecciónP, Puesto, Salario, NombreC, DirecciónC, Teléfono)

| <i>Empleados</i> | | | | | |
|---------------------------|-----------------------|--------------------------|----------------------|-----------------------|----------------------|
| <u>Id empleado</u> | <u>NombreE</u> | <u>DirecciónE</u> | <u>Puesto</u> | <u>Salario</u> | <u>Centro</u> |
| 123A | Ana Almansa | c/ Argentales | Profesor | 20.000 | Informática |
| 456B | Bernardo Botín | c/ Barcelona | Administrativo | 15.000 | Matemáticas |
| 789C | Carlos Crespo | c/ Cruz | Catedrático | 30.000 | CC. Empresariales |
| 012D | David Díaz | c/ Daroca | Ayudante | 10.000 | Informática |

| <i>Centros</i> | | |
|-----------------------|--------------------------|------------------------|
| <u>NombreC</u> | <u>DirecciónC</u> | <u>Teléfono</u> |
| Informática | Complutense | 123 |
| Matemáticas | Complutense | 456 |
| CC. Empresariales | Somosaguas | 789 |

| <i>Empleados_Centros</i> | | | | | | | |
|---------------------------|-----------------------|--------------------------|----------------------|-----------------------|----------------------|--------------------------|------------------------|
| <u>Id empleado</u> | <u>NombreP</u> | <u>DirecciónP</u> | <u>Puesto</u> | <u>Salario</u> | <u>Centro</u> | <u>DirecciónC</u> | <u>Teléfono</u> |
| 123A | Ana Almansa | c/ Argentales | Profesor | 20.000 | Informática | Complutense | 123 |
| 456B | Bernardo Botín | c/ Barcelona | Administrativo | 15.000 | Matemáticas | Complutense | 456 |
| 789C | Carlos Crespo | c/ Cruz | Catedrático | 30.000 | CC. Empresariales | Somosaguas | 789 |
| 012D | David Díaz | c/ Daroca | Ayudante | 10.000 | Informática | Complutense | 123 |

La relación Empleados_Centros presenta redundancia de datos porque se repite para cada empleado la información asociada al centro. Las relaciones con datos redundantes presentan diferentes anomalías de actualización: son las anomalías de inserción, borrado y modificación.

Anomalías de actualización

- Anomalías de inserción. Se produce en dos casos. En primer lugar, cuando se inserta una nueva fila sin respetar las dependencias funcionales. En el ejemplo anterior puede ocurrir si se añade una fila de un empleado adscrito a Informática y con un teléfono distinto de 123. En segundo lugar, la imposibilidad de añadir nuevos datos para el consecuente de la dependencia funcional sin que exista un antecedente para ella. En el ejemplo anterior no se puede dar de alta un centro a menos que exista un empleado destinado en él. Sería necesario dejar valores nulos en la clave (Id_empleado).
- Anomalías de modificación. Se produce cuando se modifican las columnas con datos redundantes de sólo un subconjunto de las filas con el mismo dato. En el ejemplo puede ocurrir cuando se modifica el teléfono de Informática sólo en la primera fila.
- Anomalías de eliminación. Se produce cuando se eliminan todas las filas en las que aparecen los datos redundantes por lo que se pierden los datos de la dependencia funcional. Si se elimina la segunda fila porque el empleado se da de baja, se pierden también los datos del centro.

Las anomalías de actualización aparecen también en los modelos de red y jerárquico, y se resuelven con campos virtuales y tipos de registros virtuales implementados con punteros. Los modelos orientados a objetos evitan el problema mediante la referencia en lugar de la copia.

Obtención de datos incorrectos

Dividir una relación de forma inadecuada puede producir resultados incorrectos al aplicar la operación Join natural (I) para recuperar la información original.

Ejemplo 2.

Al dividir la tabla Empleados_Centros anterior de la siguiente forma:

| <i>Ubicaciones_Empleados</i> | |
|------------------------------|-------------------|
| NombreE | DirecciónC |
| Ana Almansa | Complutense |
| Bernardo Botín | Complutense |
| Carlos Crespo | Somosaguas |
| David Díaz | Complutense |

| <i>Datos_Empleados</i> | | | | | | |
|------------------------|-------------------|----------------|----------------|-------------------|-------------------|-----------------|
| Id empleado | DirecciónE | Puesto | Salario | Centro | DirecciónC | Teléfono |
| 123A | c/ Argentales | Profesor | 20.000 | Informática | Complutense | 123 |
| 456B | c/ Barcelona | Administrativo | 15.000 | Matemáticas | Complutense | 456 |
| 789C | c/ Cruz | Catedrático | 30.000 | CC. Empresariales | Somosaguas | 789 |
| 012D | c/ Daroca | Ayudante | 10.000 | Informática | Complutense | 123 |

| <i>Empleados_Centros = Ubicaciones_Empleados • Datos_Empleados</i> | | | | | | | |
|--------------------------------------------------------------------|----------------|-------------------|----------------|----------------|-------------------|-------------------|-----------------|
| Id empleado | NombreE | DirecciónE | Puesto | Salario | Centro | DirecciónC | Teléfono |
| 123A | David Díaz | c/ Argentales | Profesor | 20.000 | Informática | Complutense | 123 |
| 123A | Bernardo Botín | c/ Argentales | Profesor | 20.000 | Informática | Complutense | 123 |
| 123A | Ana Almansa | c/ Argentales | Profesor | 20.000 | Informática | Complutense | 123 |
| 456B | David Díaz | c/ Barcelona | Administrativo | 15.000 | Matemáticas | Complutense | 456 |
| 456B | Bernardo Botín | c/ Barcelona | Administrativo | 15.000 | Matemáticas | Complutense | 456 |
| 456B | Ana Almansa | c/ Barcelona | Administrativo | 15.000 | Matemáticas | Complutense | 456 |
| 789C | Carlos Crespo | c/ Cruz | Catedrático | 30.000 | CC. Empresariales | Somosaguas | 789 |
| 012D | David Díaz | c/ Daroca | Ayudante | 10.000 | Informática | Complutense | 123 |
| 012D | Bernardo Botín | c/ Daroca | Ayudante | 10.000 | Informática | Complutense | 123 |
| 012D | Ana Almansa | c/ Daroca | Ayudante | 10.000 | Informática | Complutense | 123 |

El atributo que relaciona las tablas originales es DirecciónC, que no es clave de ninguna de las relaciones. Como conclusión se deben dividir las tablas de forma que estén relacionadas por atributos clave. Es una conclusión informal que se formaliza al estudiar más adelante la propiedad de reunión no aditiva. La propiedad de reunión no aditiva asegura recuperar exactamente la información original.

6.3 Dependencias funcionales

6.3.1 Introducción y definiciones

Para representar las restricciones de integridad que debe cumplir un sistema utilizaremos las *dependencias funcionales*. Este concepto será básico para entender la definición de las primera, segunda y tercera formas normales, así como la Forma Normal de Boyce Codd.

Una dependencia funcional (DF) es una propiedad semántica de un esquema de relación que presentan las tuplas válidas de la relación que determina para cada valor de un conjunto de atributos X el valor de otro conjunto de atributos Y . Es decir, dada una tupla t_1 de la relación con un valor para X y otro para Y , si aparece otra tupla t_2 con el mismo valor para X , entonces esta tupla debe tener el mismo valor en Y que t_1 .

Ejemplo 3.

En la siguiente relación se combinan los datos de los empleados, como su código de identificación y nombre, y de los centros a los que están adscritos, como la dirección y el teléfono.

| <i>Empleados_Centros</i> | | | | | | | |
|--------------------------|----------------|-------------------|---------------|----------------|---------------|-------------------|------------------|
| Id empleado | NombreE | DirecciónE | Puesto | Salario | Centro | DirecciónC | TeléfonoC |
| 123A | Ana Almansa | c/ Argentaes | Profesor | 20.000 | Informática | c/ Complutense | 123 |
| 012D | David Díaz | c/ Daroca | Ayudante | 10.000 | Informática | c/ Complutense | 123 |
| 789C | Carlos Crespo | c/ Cruz | Catedrático | 30.000 | Empresariales | c/ Coruña | 789 |

En este ejemplo se muestra gráficamente que el valor del conjunto de campos DirecciónC y TeléfonoC depende del valor del campo Centro. En concreto, a un centro en particular le corresponden unívocamente una dirección y un teléfono. Es decir, cada vez que aparezca una fila con el valor Informática para Centro, siempre le corresponderá los mismos valores para los campos DirecciónC y TeléfonoC.

Se dice entonces que tanto DirecciónC como TeléfonoC son dependientes funcionalmente de Centro. Por cada fila con un mismo valor de Centro se repiten los valores DirecciónC y TeléfonoC, lo que implica una redundancia de valores no deseable que se estudiará más adelante en la normalización de relaciones.

La validez de una relación con respecto a las DF se interpreta desde el significado que el diseñador asocia a la relación. Por tanto, una DF no se puede inferir de una relación, sino que se debe definir explícitamente sobre los atributos de la relación conociendo perfectamente su semántica. Una DF define los estados consistentes de una relación en función de las dependencias entre los valores de los atributos.

A continuación se proporciona una definición más formal de las dependencias funcionales.

Definición 1. Dependencias funcionales

Sea $R = \{A_1, \dots, A_n\}$ el esquema universal de la base de datos relacional, es decir, el conjunto de todos los atributos que pueden definirla y r una instancia del esquema R .

Una dependencia funcional $X \rightarrow Y$ (los valores de X determinan unívocamente (o funcionalmente) los valores de Y) entre dos conjuntos de atributos X e Y , tales que $X, Y \subseteq R$ especifica la siguiente restricción: $\forall t_1, t_2 \in r$ tales que $t_1[X] = t_2[X]$, entonces $t_1[Y] = t_2[Y]$. X se denomina antecedente e Y consecuente.

En otras palabras, quiere decir que los componentes Y de cada tupla de r están determinados unívocamente por los valores de X .

Observaciones:

- $X \rightarrow Y$ no implica necesariamente $Y \rightarrow X$.

Ejemplo 4.

$\{NIF\} \rightarrow \{Nombre\}$.

Sin embargo, no es cierto $\{Nombre\} \rightarrow \{NIF\}$ puesto que se pueden repetir nombres para diferentes personas. No se debe confiar en general en la dependencia funcional $\{NIF\} \rightarrow \{Nombre\}$ porque en la práctica también hay NIF repetidos. Por ello, en las bases de datos generalmente se usa un identificador propio que identifica unívocamente cada tupla asociada a una persona.

- Una dependencia funcional determina una relación uno a varios entre dos conjuntos de atributos:

$X \xrightarrow{1:N} Y$, es decir:

Para un valor de X sólo puede haber un valor de Y , pero para un valor de Y habrá en general varios de X . Por lo tanto, una dependencia funcional se puede observar como una restricción de cardinalidad entre conjuntos de atributos de una *misma* relación.

Ejemplo 5.

| <i>Empleados_Centros</i> | | | | | | | |
|--------------------------|----------------|-------------------|---------------|----------------|---------------|-------------------|------------------|
| Id empleado | NombreE | DirecciónE | Puesto | Salario | Centro | DirecciónC | TeléfonoC |
| 123A | Ana Almansa | c/ Argentales | Profesor | 20.000 | Informática | c/ Complutense | 123 |
| 012D | David Díaz | c/ Daroca | Ayudante | 10.000 | Informática | c/ Complutense | 123 |
| 789C | Carlos Crespo | c/ Cruz | Catedrático | 30.000 | Empresariales | c/ Coruña | 789 |

X ↑ Y

Corolario 1.

- Una restricción de cardinalidad de uno a varios entre dos esquemas de relación R_1 y R_2 con superclaves $X \subseteq R_1$ e $Y \subseteq R_2$ se especifica con la dependencia funcional $X \rightarrow Y$ en una nueva relación R_3 .

Ejemplo 6.

| <i>Empleados_Centros</i> | | | | | | | |
|--------------------------|----------------|-------------------|---------------|----------------|---------------|-------------------|------------------|
| Id empleado | NombreE | DirecciónE | Puesto | Salario | Centro | DirecciónC | TeléfonoC |
| 123A | Ana Almansa | c/ Argentales | Profesor | 20.000 | Informática | c/ Complutense | 123 |
| 012D | David Díaz | c/ Daroca | Ayudante | 10.000 | Informática | c/ Complutense | 123 |
| 789C | Carlos Crespo | c/ Cruz | Catedrático | 30.000 | Empresariales | c/ Coruña | 789 |

X ↑ Y

- Una restricción de cardinalidad de uno a uno entre dos esquemas de relación R_1 y R_2 con superclaves $X \subseteq R_1$ e $Y \subseteq R_2$ se especifica con las dependencias funcionales $X \rightarrow Y$ e $Y \rightarrow X$ en un nuevo esquema de relación R_3 . No obstante, las relaciones uno a uno se implementan generalmente como un atributo de la relación.
- Una superclave se puede definir en términos de dependencias funcionales. S es superclave del esquema de relación R si $S \rightarrow R$ y $S \subseteq R$. Es decir, si $\forall t_1, t_2 \in r$ tales que $t_1[S] = t_2[S]$, entonces $t_1[R] = t_2[R]$, lo cual implica $t_1 = t_2$ porque deben coincidir en todos sus atributos, por lo que se está hablando de la misma tupla.

Ejemplo 7.

Las siguientes son dependencias funcionales de la tabla anterior:

$\{Id_empleado\} \rightarrow \{NombreE, DirecciónE, Puesto, Salario, Centro, DirecciónC, TeléfonoC\}$
 $\{Centro\} \rightarrow \{DirecciónC, TeléfonoC\}$
 $\{DirecciónC\} \rightarrow \{Centro, TeléfonoC\}$
 $\{TeléfonoC\} \rightarrow \{Centro, DirecciónC\}$

A partir de la definición de DF se puede definir el concepto de satisfacción de DF.

Definición 2. Satisfacción de dependencias funcionales

Una relación r con esquema R satisface una dependencia funcional $X \rightarrow Y$, con $X, Y \subseteq R$, si todas las tuplas de r satisfacen $\forall t_1, t_2 \in r$ tales que $t_1[X] = t_2[X]$, entonces $t_1[Y] = t_2[Y]$.

La comprobación de la satisfacción de dependencias funcionales es necesaria en casos como la migración de datos o la actualización de sistemas heredados.

Bajo esta definición se pueden proponer algoritmos sencillos para comprobar la satisfacción de un conjunto de dependencias funcionales de una relación r , o para comprobar la validez de la inserción de una tupla.

Algoritmos para la comprobación de integridad definida por un conjunto de dependencias funcionales**Algoritmo 1.**

a) Algoritmo para comprobar la integridad de una relación con respecto a un conjunto de dependencias funcionales.

Entrada: relación r con un conjunto enumerable de tuplas $t_i \in r, i = 1 \dots n$ y conjunto enumerable D de dependencias funcionales $d_i = X_i \rightarrow Y_i, d_i \in D, i = 1 \dots m$.

```
for i:=1 to n-1
  for j:=i+1 to n
    for k:=1 to m
      if  $t_i[X_k] = t_j[X_k]$  and  $t_i[Y_k] \neq t_j[Y_k]$  then
        Valores inconsistentes de  $t_i$  y  $t_j$  debido a DF  $d_k$ 
```

Algoritmo 2.

b) Algoritmo para comprobar la integridad de la inserción de una tupla en una relación con respecto a un conjunto de dependencias funcionales.

Entrada: relación r con un conjunto enumerable de tuplas $t_i \in r, i = 1 \dots n$, una tupla t para insertar en r y un conjunto enumerable D de dependencias funcionales $d_i = X_i \rightarrow Y_i, d_i \in D, i = 1 \dots m$.

```
for i:=1 to n
  for j:=1 to m
    if  $t[X_j] = t_i[X_j]$  and  $t[Y_j] \neq t_i[Y_j]$  then
      Valores inconsistentes de  $t_i$  y  $t$  debido a la DF  $d_j$ 
```

Las dependencias funcionales representan restricciones de integridad que el sistema de gestión de bases de datos debe asegurar. Así que, dado un cierto conjunto D de dependencias funcionales, es deseable encontrar otro conjunto E que sea lo menor posible que D de manera que cada $d \in D$ se deduzca de E , con el objetivo de que el coste de mantener la integridad definida en D se reduzca con E . Éste es un objetivo de gran interés práctico al que se dedica el resto del apartado.

Una de las maneras de reducir el coste del aseguramiento de la consistencia mediante dependencias funcionales es eliminar las que no aportan nada semánticamente, es decir, son dependencias funcionales que cumple cualquier tupla.

Definición 3. Dependencias funcionales triviales

Una dependencia funcional $X \rightarrow Y$ es trivial si y sólo si $Y \subseteq X$.

Esto sólo dice que si dos tuplas coinciden en una serie de atributos, entonces coinciden (obviamente) en un subconjunto de esos mismos atributos. Se denomina trivial porque no aporta ninguna restricción al esquema de relación.

En general interesará encontrar el conjunto mínimo de dependencias funcionales que sea semánticamente equivalente (asegure el mismo nivel de integridad) a un conjunto dado de dependencias funcionales aportadas por el diseñador de la base de datos.

6.3.2 Cierre de un conjunto de dependencias funcionales

Definición 4. Cierre de un conjunto de dependencias funcionales

El cierre de un conjunto de dependencias funcionales S , denotado S^+ , es el conjunto de todas las dependencias definidas intensionalmente por S .

En otras palabras, es el conjunto de todas las dependencias funcionales que se pueden deducir de S . Este concepto es importante para poder determinar la equivalencia semántica de dos conjuntos de dependencias y poder elegir el menor de forma que la comprobación de su satisfacción sea más rápida. Por otra parte, permite razonar sobre la descomposición de relaciones que se estudia en el tema Normalización.

Ejemplo 8.

Es fácil ver que $\{Centro\} \rightarrow \{DirecciónC, Teléfono\}$ implica $\{Centro\} \rightarrow \{DirecciónC\}$ y $\{Centro\} \rightarrow \{Teléfono\}$. Si a un centro le corresponden una dirección y un teléfono determinados, en particular también es cierto que a ese centro le corresponde una dirección, y que a ese centro le corresponde un teléfono.

Notación: Si X e Y son conjuntos de atributos, $XY = X \cup Y$

Para calcular el cierre de un conjunto de dependencias funcionales se dispone de un conjunto de axiomas de producción denominados Axiomas de Armstrong en honor a la persona que los propuso.

1. Reflexividad: Si $Y \subseteq X$, entonces $X \rightarrow Y$.

2. Aumentatividad: Si $X \rightarrow Y$, entonces $XZ \rightarrow YZ$.

3. Transitividad: Si $X \rightarrow Y$ e $Y \rightarrow Z$, entonces $X \rightarrow Z$.

Estos axiomas son correctos en cuanto que derivan información consistente con la definición de dependencia funcional. Además son completos porque permiten deducir todas las consecuencias de un conjunto de dependencias funcionales, es decir, su cierre.

Demostración de la corrección:

1. De la definición de dependencia funcional trivial.
2. Supongamos una relación r que satisfaga $X \rightarrow Y$, y dos tuplas de r , t_1 y t_2 , que coinciden en XZ pero no coinciden en YZ . Sólo es posible que difieran en X e Y , pero esto contradice $X \rightarrow Y$. Por lo tanto, de la definición de dependencia funcional, se debe cumplir $XZ \rightarrow YZ$.
3. Por la definición de dependencia funcional: de $X \rightarrow Y$ se tiene que $t_1[X] = t_2[X]$ implica $t_1[Y] = t_2[Y]$, y de $Y \rightarrow Z$ se tiene que $t_1[Y] = t_2[Y]$ implica $t_1[Z] = t_2[Z]$. Por lo tanto, si se tiene que si $t_1[X] = t_2[X]$ implica $t_1[Z] = t_2[Z]$, entonces, por la definición de dependencia funcional, se cumple $X \rightarrow Z$.

Hay otras reglas de inferencia que se deducen de los axiomas de Armstrong y que permiten calcular más rápidamente el cierre de un conjunto de dependencias funcionales.

4. Autodeterminación: $X \rightarrow X$.

5. Unión: Si $X \rightarrow Y$ y $X \rightarrow Z$, entonces $X \rightarrow YZ$.

6. Descomposición: Si $X \rightarrow YZ$, entonces $X \rightarrow Y$ y $X \rightarrow Z$.

7. Composición: Si $X \rightarrow Y$ y $Z \rightarrow W$, entonces $XZ \rightarrow YW$.

8. Pseudotransitividad: Si $X \rightarrow Y$ e $YZ \rightarrow W$, entonces $XZ \rightarrow W$.

La autodeterminación se sigue directamente de la reflexividad. La unión y la descomposición son duales.

Ejemplo 9.

Dado el conjunto S de dependencias funcionales:

$$\{A\} \rightarrow \{B, C\}$$

$$\{C, D\} \rightarrow \{E, F\}$$

Se puede demostrar que $\{A, D\} \rightarrow \{F\}$ está en S^+ :

$$\{A\} \rightarrow \{B, C\}, \text{ dada.}$$

- $\{A\} \rightarrow \{C\}$, descomposición.
- $\{A, D\} \rightarrow \{C, D\}$, aumentatividad.
- $\{C, D\} \rightarrow \{E, F\}$, dada.
- $\{A, D\} \rightarrow \{E, F\}$, transitividad de las dos anteriores.
- $\{A, D\} \rightarrow \{F\}$, descomposición.

Se puede desarrollar un algoritmo que calcule el cierre del conjunto de dependencias funcionales a partir de sólo las tres primeras reglas de inferencia aplicándolas repetidamente hasta que no se produzcan más dependencias funcionales (se alcance el punto fijo). Este algoritmo es seguro con respecto a la completitud de los axiomas. La demostración de completitud necesita la noción de cierre de un conjunto de atributos. Sin embargo, también es un algoritmo muy ineficiente por la cantidad de dependencias funcionales que se generan.

Ejemplo 10.

Dado el conjunto de dependencias funcionales:

$$S = \{X \rightarrow \{B_1\}, \dots, X \rightarrow \{B_n\}\}$$

El cierre de S incluye todas las dependencias funcionales $X \rightarrow Y_i$ tales que $Y_i \subseteq \{B_1, \dots, B_n\}$, es decir, $2^n - 1$, demasiado grande aunque S sea pequeño.

6.3.3 Cierre de un conjunto de atributos

En la práctica no es necesario en general calcular todo el cierre de un conjunto de dependencias. Es más interesante calcular el conjunto de las dependencias que tienen en su parte izquierda un conjunto especificado de atributos.

El cálculo del cierre de un conjunto de atributos permite:

1. Comprobar si una dependencia funcional se deduce de un conjunto de dependencias funcionales sin necesidad de calcular su cierre. Se puede determinar si su comprobación es redundante para la integridad de los datos.
2. Comprobar si un conjunto de atributos es superclave. Asegura que el conjunto de atributos elegido por el diseñador es adecuado para determinar unívocamente cada tupla de una relación. Permite determinar superclaves que se pueden usar como índice sin repetidos (algoritmo de indexación más eficiente) para el acceso a los datos mediante consultas.
3. Calcular un conjunto mínimo de dependencias funcionales. Útil para mantener la comprobación de integridad menos costosa.

Definición 5. Cierre de un conjunto de atributos

El cierre de un conjunto de atributos X con respecto a un conjunto de dependencias funcionales S , denotado X_S^+ , es el conjunto de atributos Y tales que $X \rightarrow Y$ se puede deducir de S .

En otras palabras, el cierre de un conjunto de atributos X es el conjunto de atributos Y determinados funcionalmente por X .

Ahora se puede usar el siguiente lema para asegurar el primer punto anterior.

Lema 1:

$X \rightarrow Y$ se deduce de un conjunto de dependencias funcionales $S \Leftrightarrow Y \subseteq X_S^+$.

Demostración:

Sea $Y = \{B_1, \dots, B_n\}$:

\Leftarrow) Supongamos $Y \subseteq X_S^+$. Por la definición de X_S^+ se deduce en particular el conjunto de todas las dependencias funcionales $X \rightarrow \{B_i\}$. Por la regla de la unión para todas las $X \rightarrow \{B_i\}$ ($X \rightarrow \{B_1\}, \dots, X \rightarrow \{B_n\}$) se deduce $X \rightarrow Y$.

\Rightarrow) Supongamos $X \rightarrow Y$. Por la regla de descomposición se deducen todos los $X \rightarrow \{B_i\}$ y, por tanto, $Y \subseteq X_S^+$.

Algoritmo 3.

Algoritmo simple para calcular el cierre de un conjunto de atributos

Entrada: Conjunto de atributos X y un conjunto de dependencias funcionales S .

Salida: X_S^+

Resultado := X

while cambios en Resultado do

 for each $Y \rightarrow Z \in S$ do

 if $Y \subseteq \text{Resultado}$ then Resultado := Resultado $\cup Z$

Se puede demostrar que el algoritmo es correcto y completo. El algoritmo tiene una complejidad cuadrática con la cardinalidad de S . Existen otros algoritmos de complejidad lineal. Estos aspectos se tratan en los ejercicios.

Corolario 2.

Se puede determinar si una dependencia funcional $X \rightarrow Y$ se deduce de un conjunto S de dependencias funcionales si $Y \subseteq X_S^+$. Se puede determinar, por tanto, en tiempo lineal, si una dependencia funcional está en S^+ .

Corolario 3.

Se puede determinar si un conjunto de atributos C es superclave de una relación r bajo un conjunto de dependencias funcionales S si todos los atributos de r pertenecen al cierre de C , es decir, si todos los atributos de la relación están determinados funcionalmente por C . Además, será clave candidata si el conjunto de atributos C es irreducible (no hay ningún conjunto de cardinalidad menor que C tal que determine funcionalmente todos los atributos de r).

Ejercicio.

Proponer un algoritmo para determinar el conjunto de claves candidatas de una relación.

A continuación ya es posible definir lo que es un recubrimiento mínimo de dependencias o conjunto irreducible equivalente, que va a permitir mantener la integridad definida por un conjunto de dependencias funcionales a coste mínimo.

6.3.4 Recubrimientos mínimos de dependencias funcionales

Para definir un recubrimiento mínimo hay que definir dos conceptos: el recubrimiento de un conjunto de dependencias funcionales y la equivalencia entre conjuntos de dependencias funcionales.

Definición 6. Recubrimiento de un conjunto de dependencias funcionales

Dados dos conjuntos de dependencias funcionales S_1 y S_2 , se dice que S_2 es un recubrimiento de S_1 si cada dependencia de S_1 se deduce de S_2 (es decir, se puede demostrar que cada dependencia de S_1 está en el cierre de S_2).

Definición 7. Equivalencia entre conjuntos de dependencias funcionales

Dos conjuntos de dependencias funcionales S_1 y S_2 son equivalentes si $S_1^+ = S_2^+$.

De forma alternativa se define como: Dos conjuntos de dependencias funcionales S_1 y S_2 son equivalentes si S_1 es un recubrimiento de S_2 y S_2 es un recubrimiento de S_1 .

La comprobación de la equivalencia se hace usando el Corolario 2. Así, para cada dependencia funcional de S_1 se comprueba si ésta pertenece al cierre de S_2 . De igual forma se procede con cada dependencia de S_2 comprobando que pertenezca al cierre de S_1 .

Definición 8. Conjunto mínimo (o irreducible) de dependencias funcionales

Un conjunto S de dependencias funcionales es irreducible si y solamente si cumple las siguientes propiedades:

1. La parte derecha de cada dependencia funcional de S tiene sólo un atributo.
2. La parte izquierda de cada dependencia funcional de S es irreducible en el sentido en que si se elimina algún atributo, necesariamente cambia el cierre de S .
3. No se puede eliminar ninguna dependencia funcional de S sin cambiar su cierre.

Definición 9. Recubrimiento mínimo de un conjunto de dependencias funcionales

Al conjunto mínimo de dependencias funcionales S_1 equivalente a S_2 se le denomina recubrimiento mínimo de S_2 .

Se puede demostrar que todo conjunto de dependencias funcionales tiene al menos un recubrimiento mínimo, por lo que se plantea el siguiente lema.

Lema 2.

Todo conjunto S de dependencias funcionales tiene un conjunto de dependencias funcionales equivalente en el que el lado derecho de cada dependencia funcional tiene un único atributo.

Demostración:

La regla de descomposición asegura que una dependencia funcional $X \rightarrow \{B_1, \dots, B_n\}$ se puede dividir en n dependencias funcionales $X \rightarrow \{B_1\}, \dots, X \rightarrow \{B_n\}$.

Teorema 1.

Todo conjunto S de dependencias funcionales tiene al menos un recubrimiento mínimo.

Algoritmo 4

Entrada: Conjunto S de dependencias funcionales

Salida: S' Recubrimiento mínimo de S

El algoritmo cuenta de 3 pasos que deben ser aplicados en el orden indicado:

1. Dejar lados derechos unitarios.
Sea $S' = \{ \}$. Por cada dependencia $X \rightarrow A_1 \dots A_n \in S$, con A_i atributos, incluir en S' las n dependencias $X \rightarrow A_1, \dots, X \rightarrow A_n$.
2. Eliminar atributos redundantes.
Para cada dependencia funcional $X \rightarrow Y \in S'$, $X = \{A_1, \dots, A_n\}$ y para cada $i=1 \dots n$, comprobar si $(X - \{A_i\} \rightarrow Y) \in (S')^+$, es decir si en S' se tiene $Y \subseteq (X - \{A_i\})^+$.
Si esto ocurre, A_i es redundante en la dependencia y debemos eliminarlo haciendo $S' = S' - \{X \rightarrow Y\} \cup \{X - \{A_i\} \rightarrow Y\}$.
3. Eliminar dependencias redundantes.
Para cada $X \rightarrow Y \in S'$, ver si $(X \rightarrow Y) \in (S' - \{X \rightarrow Y\})^+$, es decir si en $(S' - \{X \rightarrow Y\})$ se cumple $Y \in X^+$.

Si esto sucede $X \rightarrow Y$ es redundante y debemos eliminarla: $S' = S' - \{X \rightarrow Y\}$

Observación

Dependiendo del orden en el que se consideren las dependencias funcionales, y en el caso del paso 2 los atributos de los lados izquierdos podemos obtener distintos recubrimientos mínimos, aunque todas serán equivalentes.

Ejemplo 11. Encontrar un recubrimiento mínimo para el siguiente conjunto de dependencias funcionales S formado por los siguientes elementos.

$$\{A\} \rightarrow \{B, C\}$$

$$\{B\} \rightarrow \{C\}$$

$$\{A\} \rightarrow \{B\}$$

$$\{A, B\} \rightarrow \{C\}$$

$$\{A, C\} \rightarrow \{D\}$$

1. Primer paso: Obtenemos el siguiente conjunto S'

$$\{ \{A\} \rightarrow \{B\}, \{A\} \rightarrow \{C\}, \{B\} \rightarrow \{C\}, \{A\} \rightarrow \{B\}, \{A, B\} \rightarrow \{C\}, \{A, C\} \rightarrow \{D\} \}$$

Antes de aplicar el segundo paso, vemos que se repite $\{A\} \rightarrow \{B\}$, por lo que en el conjunto resultado no se repite.

2. Segundo paso. Eliminar atributos redundantes.

Partimos de $S' = \{ \{A\} \rightarrow \{B\}, \{A\} \rightarrow \{C\}, \{B\} \rightarrow \{C\}, \{A, B\} \rightarrow \{C\}, \{A, C\} \rightarrow \{D\} \}$.

Tenemos que comprobar $\{A, B\} \rightarrow \{C\}$ y $\{A, C\} \rightarrow \{D\}$.

Vemos si sobra A en $\{A, B\} \rightarrow \{C\}$. Calculamos $\{B\}^+$ en S' . $\{B\}^+ = \{B, C\}$. Como incluye $\{C\}$ el atributo $\{A\}$ es redundante. Tenemos entonces

$S' = \{ \{A\} \rightarrow \{B\}, \{A\} \rightarrow \{C\}, \{B\} \rightarrow \{C\}, \{B\} \rightarrow \{C\}, \{A, C\} \rightarrow \{D\} \}$. Como está repetido lo quitamos:

$$S' = \{ \{A\} \rightarrow \{B\}, \{A\} \rightarrow \{C\}, \{B\} \rightarrow \{C\}, \{A, C\} \rightarrow \{D\} \}$$

Vemos si sobra A en $\{A, C\} \rightarrow \{D\}$. Calculamos $\{C\}^+$ en S' . $\{C\}^+ = \{C\}$. Como no incluye $\{D\}$ el atributo $\{A\}$ no es redundante en esta dependencia.

Vemos si sobra C en $\{A, C\} \rightarrow \{D\}$. Calculamos $\{A\}^+$ en S' . $\{A\}^+ = \{A, B, C, D\}$. Como incluye $\{D\}$ el atributo $\{C\}$ es redundante en esta dependencia. Tenemos:

$$S' = \{ \{A\} \rightarrow \{B\}, \{A\} \rightarrow \{C\}, \{B\} \rightarrow \{C\}, \{A\} \rightarrow \{D\} \}$$

3. Tercer paso. Eliminar dependencias redundantes.

$\{A\} \rightarrow \{B\}$ no sobra porque en $(S' - \{\{A\} \rightarrow \{B\}\})$ se tiene que $\{A\}^+ = \{A, C, D\}$, que no incluye $\{B\}$.

$\{A\} \rightarrow \{C\}$ sobra porque en $(S' - \{\{A\} \rightarrow \{C\}\})$ se tiene que $\{A\}^+ = \{A, B, D, C\}$, que incluye $\{C\}$.

Tenemos por tanto:

$$S' = \{ \{A\} \rightarrow \{B\}, \{B\} \rightarrow \{C\}, \{A\} \rightarrow \{D\} \}$$

Análogamente tendremos que ni $\{B\} \rightarrow \{C\}$ ni $\{A\} \rightarrow \{D\}$ sobran, por lo que ya tenemos un recubrimiento mínimo en S' .

Formas normales y normalización

La forma normal de una relación se refiere a la mayor condición de forma normal que satisface un esquema de relación, indicando así el grado hasta el que se ha normalizado. Lograremos la normalización mediante descomposiciones de una tabla en varias. Al hacer esta descomposición se debe lograr que el esquema obtenido sea equivalente al original. En particular dos propiedades que se deben cumplir para poder asegurarlo son:

- La propiedad de preservación de dependencias, que asegura que las dependencias funcionales originales se mantienen en algún esquema de relación después de la descomposición.
- La propiedad de reunión (join) no aditiva (o sin pérdida), que evita el problema de la generación de tuplas incorrectas comentado anteriormente.

Las formas normales más habituales, por orden ascendente de exigencia de las propiedades deseadas, son:

- Primera (1FN)
- Segunda (2FN)
- Tercera (3FN)
- Boyce/Codd (FNBC)
- Cuarta (4FN)

En general, los diseños prácticos exigen 3FN, aunque hay circunstancias especiales en las que 2FN puede ser suficiente.

Hay otras formas normales más exigentes y que se refieren a otras propiedades deseables. Sin embargo, la utilidad práctica de estas formas normales es cuestionable cuando las restricciones en que se basan son difíciles de entender o identificar por los diseñadores y usuarios. Así, en la práctica se usan hasta la forma normal de Boyce/Codd o hasta la cuarta.

El proceso de asegurar una forma normal para un esquema se denomina normalización.

A continuación se estudiarán las formas normales mencionadas y para cada una de ellas se indicará el procedimiento que asegura que un esquema que no esté en una forma normal determinada se convierta en un conjunto de esquemas que asegure esa forma normal.

Primera forma normal

Actualmente se considera como parte de la definición formal de relación, porque establece que

Definición 10

Los dominios de los atributos sólo pueden ser atómicos, para evitar atributos multivalorados, compuestos y sus combinaciones. En definitiva evita las relaciones dentro de las relaciones.

Ejemplo 12.

Si se asume que en la entidad Centros, un centro puede tener más de un teléfono, podríamos tener una representación como la siguiente.

| <i>Centros</i> | | |
|-------------------|-------------|-----------------|
| <u>NombreC</u> | DirecciónC | Teléfonos |
| Informática | Complutense | {123, 321, 213} |
| Matemáticas | Complutense | {456} |
| CC. Empresariales | Somosaguas | {789, 987} |

Sin embargo, esto supondría el uso del atributo multivalorado Teléfonos. Hay tres posibilidades de representar la entidad para satisfacer la primera forma normal:

1. Eliminar el atributo Teléfonos y crear una nueva relación que asocie en cada fila un centro con un teléfono. La clave de la primera relación debe formar parte de la clave de la segunda relación. Presenta como inconveniente añadir una nueva relación al esquema de la base de

datos y redundancia. Presenta anomalías cuando se borra un centro y no se borran los teléfonos asociados. La integridad referencial asegura evitar las anomalías.

| Centros | |
|-------------------|-------------------|
| NombreC | DirecciónC |
| Informática | Complutense |
| Matemáticas | Complutense |
| CC. Empresariales | Somosaguas |

| Teléfonos | |
|-------------------|-----------------|
| NombreC | Teléfono |
| Informática | 123 |
| Informática | 321 |
| Informática | 213 |
| Matemáticas | 456 |
| CC. Empresariales | 789 |
| CC. Empresariales | 987 |

2. Ampliar la clave de la relación de manera que incluya al atributo multivalorado. Presenta como inconveniente añadir redundancia que provoca anomalías.

| Centros | | |
|-------------------|-------------------|-----------------|
| NombreC | DirecciónC | Teléfono |
| Informática | Complutense | 123 |
| Informática | Complutense | 321 |
| Informática | Complutense | 213 |
| Matemáticas | Complutense | 456 |
| CC. Empresariales | Somosaguas | 789 |
| CC. Empresariales | Somosaguas | 987 |

3. Si se conoce la cardinalidad máxima del atributo multivalorado, se pueden crear tantas columnas como la cardinalidad máxima. Presenta como inconveniente el uso de valores Null.

| NombreC | DirecciónC | Teléfono1 | Teléfono2 | Teléfono3 |
|-------------------|-------------------|------------------|------------------|------------------|
| Informática | Complutense | 123 | 321 | 213 |
| Matemáticas | Complutense | 456 | Null | Null |
| CC. Empresariales | Somosaguas | 789 | 987 | Null |

Si el atributo multivalorado es compuesto, como es el caso de representar varias direcciones para un empleado:

Empleados(Id_ empleado, NombreP, {Direcciones(Calle, Ciudad, CódigoPostal)})

Esta relación se puede descomponer en dos:

Empleados(Id_ empleado, NombreP)

DireccionesP(Id_ empleado, Calle, Ciudad, CódigoPostal)

Este procedimiento de desanidamiento se puede aplicar recursivamente a cualquier relación con atributos multivalorados, teniendo en cuenta que es necesario propagar la clave de la relación original a la clave de la nueva relación, que contiene además la clave que identifica unívocamente al atributo multivalorado.

Segunda forma normal

Ejemplo 13.

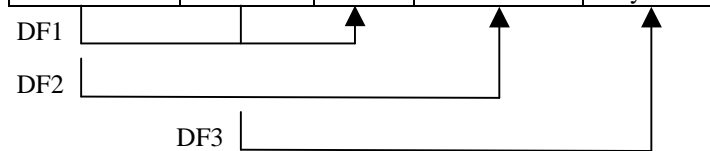
En el ejemplo a continuación se puede observar que existen anomalías de actualización causadas por las dependencias funcionales DF2 y DF3, ya que como sus antecedentes no son clave, puede haber varias filas con el mismo consecuente. Se usa una notación gráfica para la expresión de las dependencias funcionales. Así:

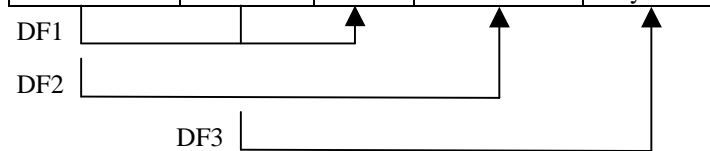
$$DF1 = \{Id_empleado, NúmeroP\} \rightarrow \{Horas\}$$

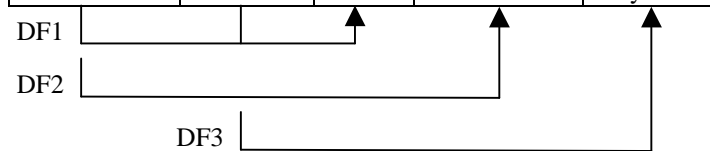
$$DF2 = \{Id_empleado\} \rightarrow \{NombreE\}$$

$$DF3 = \{NúmeroP\} \rightarrow \{NombreP\}$$

| <i>Personal_Proyectos</i> | | | | |
|---------------------------|----------------|--------------|----------------|----------------|
| <u>Id_empleado</u> | <u>NúmeroP</u> | <u>Horas</u> | <u>NombreE</u> | <u>NombreP</u> |
| 123A | P-1 | 16 | Ana Almansa | Proyecto 1 |
| 012D | P-1 | 8 | David Díaz | Proyecto 1 |
| 012D | P-2 | 4 | David Díaz | Proyecto 2 |

DF1: 

 DF2: 

 DF3: 

La segunda forma normal evita este tipo de anomalías. Para evitarlas se basa en el concepto de dependencia funcional completa.

Definición 11. Un atributo se dice *primo* si es miembro de alguna clave candidata.

Definición 12. Dependencia funcional completa

Una dependencia funcional $X \rightarrow Y$ es una dependencia funcional completa si no se cumple $X - \{A_i\} \rightarrow Y, A_i \in X$. Cuando se cumple se dice que se trata de una dependencia funcional parcial.


Definición 13. Segunda forma normal

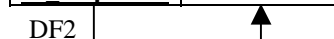
Una relación está en segunda forma normal si cada atributo no primo A la dep. Clave \rightarrow A se cumple y es total.

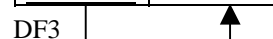
Un esquema que no se encuentre en segunda forma normal puede traducirse en varios esquemas que sí lo estén. El procedimiento es crear tantas nuevas relaciones como dependencias funcionales no sean completas.

Ejemplo 14.

Así, el ejemplo anterior se traduce en:

| <i>PP1</i> | | |
|--------------------|-------------------------------------------------------------------------------------|--------------|
| <u>Id_empleado</u> | <u>NúmeroP</u> | <u>Horas</u> |
| DF1 |  | |

| <i>PP2</i> | |
|--------------------|--------------------------------------------------------------------------------------|
| <u>Id_empleado</u> | <u>NombreE</u> |
| DF2 |  |

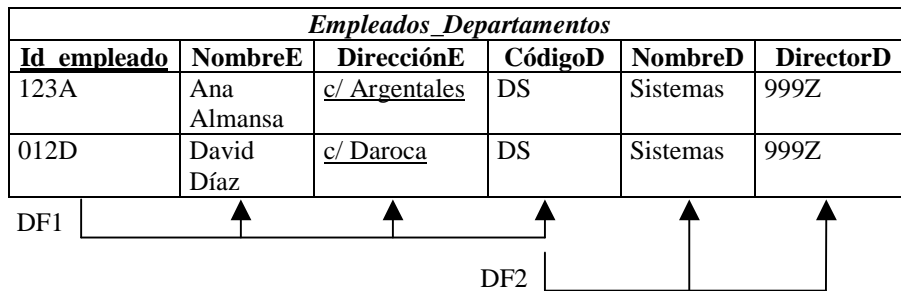
| <i>PP3</i> | |
|----------------|---------------------------------------------------------------------------------------|
| <u>NúmeroP</u> | <u>NombreP</u> |
| DF3 |  |

Hay que observar que este procedimiento asegura que el resultado está, al menos, en segunda forma normal. En particular, y como se puede contrastar con la definición de otras formas normales, el resultado conseguido en este ejemplo se encuentra en 4FN, como se podrá comprobar más adelante.

Tercera forma normal

Ejemplo 15.

En el ejemplo a continuación se puede observar que existen anomalías de actualización causadas por la dependencia funcional DF2. Sin embargo, este esquema está en segunda forma normal porque los dos últimos atributos, que son los que causan las anomalías, dependen completa (y transitivamente) del único atributo que forma la clave candidata.

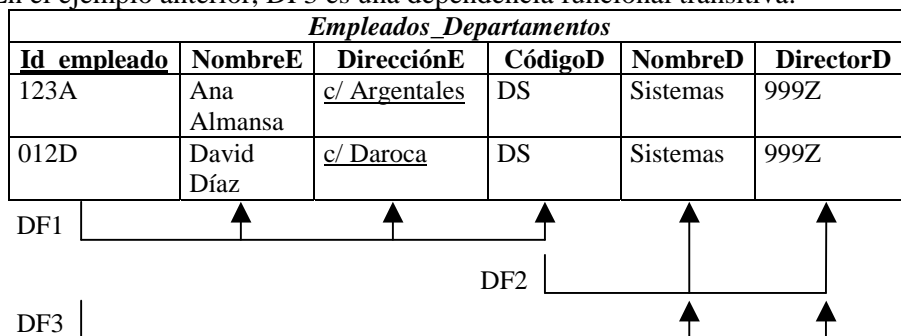


La tercera forma normal se basa en el concepto de dependencia funcional transitiva.

Definición 14. Dependencia funcional transitiva

Una dependencia funcional $X \rightarrow Y$ es una dependencia funcional transitiva si existe un conjunto de atributos Z que ni es clave candidata ni es subconjunto de ninguna clave y además se cumple $X \rightarrow Z$ y $Z \rightarrow Y$. (Z si puede ser un subconjunto estricto de una clave candidata)

En el ejemplo anterior, DF3 es una dependencia funcional transitiva:



Definición 15. Tercera forma normal

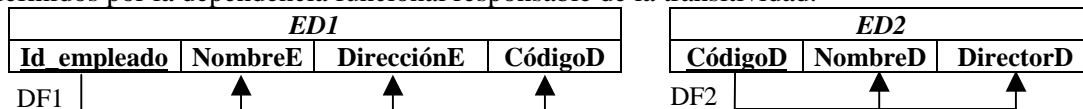
- Un esquema está en tercera forma normal si satisface la segunda forma normal y ninguno de los atributos no primos dependen transitivamente de una clave candidata. (Def. de Codd).

Esta definición se puede reformular como (Carlo Zaniolo, 1982):

- Un esquema está en tercera forma normal con respecto a un conjunto de dependencias funcionales S si: para toda dependencia funcional no trivial $X \rightarrow Y$ de S se cumple que o bien X es superclave o bien Y forma parte de una clave candidata.

La reformulación admite una implementación del test de tercera forma normal más inmediata con la definición de cierre de un conjunto de dependencias funcionales.

El procedimiento para normalizar esta relación consiste en descomponerla en los atributos definidos por la dependencia funcional responsable de la transitividad.



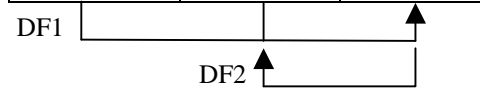
Forma normal de Boyce-Codd

La forma normal de Boyce-Codd (FNBC) es más estricta que la 3FN, aunque su definición es más simple. Veamos primero un ejemplo.

Ejemplo 16.

La siguiente relación está en 3FN, pero no en FNBC, que evita otras redundancias que la 3FN no considera. En este ejemplo se almacena información de los investigadores participantes en proyectos, que pueden ser codirigidos, pero los investigadores principales no pueden dirigir más de uno.

| <i>Proyectos</i> | | |
|---------------------|-----------------|-------------------|
| <u>Investigador</u> | <u>Proyecto</u> | <u>IPrincipal</u> |
| 111A | Proyecto 1 | 123A |
| 222B | Proyecto 1 | 012D |
| 333C | Proyecto 1 | 123A |
| 444D | Proyecto 2 | 789C |
| 444D | Proyecto 1 | 123A |

DF1 

En este ejemplo, que cumple la 3NF, hay una anomalía que se debería poder evitar, porque si no se vigila la dependencia funcional DF2 se podría añadir una tupla de manera que una persona fuese investigadora principal de dos proyectos.

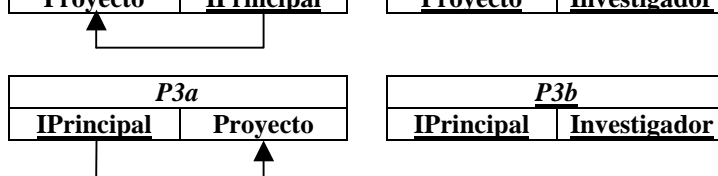
Definición 16. Forma normal de Boyce-Codd (Ian Heath, 1971)

Un esquema está en forma normal de Boyce-Codd con respecto a un conjunto de dependencias funcionales S si para toda dependencia funcional no trivial $X \rightarrow Y$ de S se cumple que X es superclave. (Recordar que una dep. $X \rightarrow Y$ es trivial si $Y \subseteq X$).

Ejemplo 17.

Con esta definición se detecta que el ejemplo anterior no está en FNBC. La descomposición del esquema no es inmediata, hay tres posibilidades:

| | | | |
|---------------------|-------------------|---------------------|---------------------|
| <i>P1a</i> | | <i>P1b</i> | |
| <u>Investigador</u> | <u>IPrincipal</u> | <u>Investigador</u> | <u>Proyecto</u> |
| <i>P2a</i> | | <i>P2b</i> | |
| <u>Proyecto</u> | <u>IPrincipal</u> | <u>Proyecto</u> | <u>Investigador</u> |
| <i>P3a</i> | | <i>P3b</i> | |
| <u>IPrincipal</u> | <u>Proyecto</u> | <u>IPrincipal</u> | <u>Investigador</u> |



Todas estas descomposiciones pierden la dependencia funcional DF1 porque las dependencias funcionales se refieren al contexto local de un esquema, no hacen referencia a esquemas externos. Aún peor, las dos primeras generan tuplas incorrectas en la operación join.

Por ejemplo, en la primera descomposición se pierde la información de los investigadores principales de cada proyecto:

| <i>PIa</i> | |
|---------------------|-------------------|
| <u>Investigador</u> | <u>IPrincipal</u> |
| 111A | 123A |
| 222B | 012D |
| 333C | 123A |
| 444D | 123A |
| 444D | 789C |

| <i>PIb</i> | |
|---------------------|-----------------|
| <u>Investigador</u> | <u>Proyecto</u> |
| 111A | Proyecto 1 |
| 222B | Proyecto 1 |
| 333C | Proyecto 1 |
| 444D | Proyecto 1 |
| 444D | Proyecto 2 |

| <i>PIa • PIb</i> | | |
|---------------------|-----------------|-------------------|
| <u>Investigador</u> | <u>Proyecto</u> | <u>IPrincipal</u> |
| 111A | Proyecto 1 | 123A |
| 222B | Proyecto 1 | 012D |
| 333C | Proyecto 1 | 123A |
| 444D | Proyecto 2 | 123A |
| 444D | Proyecto 2 | 789C |
| 444D | Proyecto 1 | 123A |
| 444D | Proyecto 1 | 789C |

Por tanto, ninguna de estas dos últimas descomposiciones es aceptable.

Sin embargo, hay otras descomposiciones, como la del ejemplo de la segunda forma normal que presenta en particular FNBC, y no pierde ninguna dependencia funcional.

| <i>PP1</i> | | |
|--------------------|----------------|--------------|
| <u>Id empleado</u> | <u>NúmeroP</u> | <u>Horas</u> |
| DF1 | | ↑ |

| <i>PP2</i> | |
|--------------------|----------------|
| <u>Id empleado</u> | <u>NombreE</u> |
| DF2 | ↑ |

| <i>PP3</i> | |
|----------------|----------------|
| <u>NúmeroP</u> | <u>NombreP</u> |
| DF3 | ↑ |

En la práctica, la mayoría de los esquemas que están en 3NF lo están también en FNBC.

Es necesario, por tanto, encontrar el modo de crear descomposiciones que, como mínimo, no generen tuplas incorrectas en la reunión. Si las dependencias funcionales no se aseguran en la descomposición, como en el ejemplo anterior, el diseñador debe tenerlas presente y asegurarlas por programación. Esto implica mantenerlas en el contexto global de varios esquemas, no en el contexto local de uno solo, realizar su reunión y comprobar las dependencias funcionales perdidas, lo cual no es práctico. Por lo tanto, se debe asegurar en un buen diseño que no se pierdan dependencias funcionales.

En particular vamos a pedir a una descomposición de una tabla en varias que cumpla:

- Preservación de los atributos: La unión de los atributos de las tablas generadas deben ser los atributos de la tabla original.
- Preservación de las dependencias funcionales.
- Al hacer joins se deben obtener las mismas tuplas, no tuplas de más.

Para poder determinar formalmente la condición de preservación de dependencias en la descomposición es necesaria una definición preliminar:

Definición 17. Proyección de un conjunto de dependencias funcionales

La proyección de un conjunto de dependencias funcionales S sobre los atributos R_i , denotada por $\pi_{R_i}(S)$, con $R_i \subseteq R$ y R esquema de relación, es el conjunto de dependencias funcionales $X \rightarrow Y$ en S tales que todos los atributos $X \cup Y \subseteq R_i$.

Con esta definición se determinan las dependencias asociadas a cada esquema R_i resultado de la descomposición de R . Sólo queda determinar cuándo se preservan todas.

Definición 18. Preservación de dependencias funcionales

Una descomposición $D = \{R_1, \dots, R_m\}$ del esquema R preserva las dependencias funcionales con respecto al conjunto de dependencias funcionales S sobre R si la unión de las proyecciones de S sobre cada R_i es equivalente a S , es decir:

$$(\pi_{R_1}(S) \cup \dots \cup \pi_{R_m}(S))^+ = S^+$$

Proposición 1.

Siempre es posible encontrar una descomposición D que preserve las dependencias con respecto a S tal que cada relación $R_i \in D$ esté en 3FN.

Algoritmo 4.

El siguiente algoritmo permite encontrar la descomposición en esquemas en 3FN que preserva las dependencias:

Entrada: Un esquema R y un conjunto de dependencias funcionales S .

Salida: Una descomposición D que preserve las dependencias con respecto a S en 3FN.

1. Encontrar un recubrimiento mínimo T para S . (Se puede usar el algoritmo del tema anterior)
2. Para cada $X \rightarrow Y \in T$ crear un esquema en D con atributos $\{X \cup \{A_1\} \cup \dots \cup \{A_k\}\}$, donde $X \rightarrow \{A_1\}, \dots, X \rightarrow \{A_k\}$ son todas las dependencias funcionales en T cuya parte izquierda es X . (Por tanto, X es clave para el nuevo esquema).
3. Poner el resto de atributos en una única relación para asegurar la preservación de atributos.

Para comprobar que una descomposición preserva las dependencias se puede usar el siguiente algoritmo:

Algoritmo 5.

Entrada: Un esquema R , una descomposición $D = \{R_1, \dots, R_m\}$ y un conjunto de dependencias funcionales S .

Salida: Se determina si D preserva S .

Resultado = True

For Each $X \rightarrow Y \in S$

 Zantiguo = \emptyset

 Znuevo = X

 While Zantiguo \neq Znuevo Do

 Zantiguo = Znuevo

 For $i = 1$ To m (para cada tabla de la descomposición)

 Znuevo := Znuevo $\cup ((R_i \cap Znuevo)^+ \cap R_i)$

 /* Añadimos a Z cada vez, los atributos A t.q. $R_i \cap Z \rightarrow A \in \Pi R_i(F)$ */

 Next

 End While

 If $Y \notin Znuevo$ Then Resultado = Falso

Next

Descomposición y reuniones no aditivas o sin pérdida

Una propiedad necesaria de mantener en la descomposición es que no se creen tuplas con valores incorrectos cuando una descomposición se reúne con la operación join. Hay que asegurar que la descomposición no pierda la información de integridad en las dependencias funcionales que asegure que no se creen tuplas incorrectas.

La pérdida se refiere a la pérdida de información de integridad y la no aditividad se refiere a no producir más tuplas de las que estaban en la relación antes de la descomposición.

Con la siguiente definición se caracteriza la propiedad de reunión sin pérdida o no aditiva.

Definición 19. Propiedad de reunión no aditiva

Una descomposición $D = \{R_1, \dots, R_m\}$ del esquema R presenta la propiedad de reunión no aditiva con respecto al conjunto de dependencias funcionales S sobre R si para todo estado de la relación r de R que satisfaga S , se cumple:

$$|(\pi_{R_1}(r), \dots, \pi_{R_m}(r)) = r|$$

Propiedad 1.

Una descomposición $D = \{R_1, R_2\}$ del esquema R presenta la propiedad de reunión no aditiva con respecto al conjunto de dependencias funcionales S sobre R si y sólo si se cumple alguna de las dos condiciones siguientes:

- $((R_1 \cap R_2) \rightarrow (R_1 - R_2)) \in S^+$, o bien
- $((R_1 \cap R_2) \rightarrow (R_2 - R_1)) \in S^+$

Propiedad 2.

Si se cumple que:

- Una descomposición $D = \{R_1, \dots, R_m\}$ del esquema R presenta la propiedad de reunión no aditiva con respecto al conjunto de dependencias funcionales S sobre R , y
- Una descomposición $D_1 = \{Q_1, \dots, Q_k\}$ del esquema R_i presenta la propiedad de reunión no aditiva con respecto a $\pi_{R_i}(S)$,

entonces la descomposición $D = \{R_1, \dots, R_{i-1}, Q_1, \dots, Q_k, R_{i+1}, \dots, R_m\}$ también presenta la propiedad de reunión no aditiva con respecto al conjunto de dependencias funcionales S sobre R .

Con estas dos propiedades se puede construir un algoritmo que realice una descomposición de un esquema de relación que preserve la propiedad de reunión no aditiva.

Algoritmo 6.

Entrada: un esquema R y un conjunto de dependencias funcionales S sobre R .

Salida: una descomposición en FNBC que preserve la propiedad de reunión no aditiva.

1. $D := \{R\}$.
2. while $Q \in D$, tal que Q no esté en FNBC do
 - {
 - encontrar una dependencia funcional $X \rightarrow Y$ de Q que viole la FNBC;
 - reemplazar Q por dos esquemas de relación $Q - Y$ y $X \cup Y$;
 - };

Preservación de dependencias en la descomposición y reuniones no aditivas

Si se desean preservar estas dos propiedades, en general sólo se puede asegurar que el resultado se encuentre en 3FN.

Ejemplo 18.

El esquema Proyectos se encuentra en 3FN. Admite tres descomposiciones y ninguna de ellas preserva la dependencia DF1. Sólo la última descomposición preserva reuniones no aditivas.

Para comprobarlo:

$((R_1 \cap R_2) \rightarrow (R_1 - R_2)) = (\{IPrincipal\} \rightarrow \{Proyecto\}) \in S^+$, que de hecho se trata de la dependencia funcional DF2.

Con el siguiente algoritmo, que es una modificación del 4, se puede realizar una descomposición en 3FN que preserve ambas propiedades.

Algoritmo 7.

Entrada: un esquema R y un conjunto de dependencias funcionales S sobre R .

Salida: una descomposición en 3FN que preserve dependencias y reuniones no aditivas.

1. Encontrar un recubrimiento mínimo T para S .
2. Para cada $X \rightarrow Y \in T$ crear un esquema en D con atributos $\{X \cup \{A_1\} \cup \dots \cup \{A_k\}\}$, donde $X \rightarrow \{A_1\}, \dots, X \rightarrow \{A_k\}$ son todas las dependencias funcionales en T cuya parte izquierda es X . (Por tanto, X es clave para el nuevo esquema).
3. Si ninguno de los esquemas contiene una clave candidata C , crear un nuevo esquema con C .

Observaciones

- Cada nueva relación tendrá asociado su conjunto de dependencias funcionales proyectadas, y por tanto sus claves.
- Si una relación R1 incluye como atributos una clave C de otra de otra relación R2 y C no es clave en R1, debe crearse la correspondiente restricción de foreign key entre R1 y R2.
- Al final puede haber relaciones redundantes, que deben eliminarse. Una relación será redundante si todos sus atributos forman parte de otra.

6.6 Otras formas normales

Dependencias multivaloradas

Las dependencias multivaloradas son restricciones de integridad que expresan relaciones entre los atributos de un esquema que no pueden ser expresables con las dependencias funcionales.

Ejemplo 19.

En la siguiente relación se representan los empleados, sus domicilios y teléfonos, asumiendo que pueden tener más de una vivienda y teléfono, y que no se dispone información acerca del tipo de teléfono, fijo o móvil, por lo que no se puede relacionar con un domicilio. Estos atributos son independientes entre sí. Para mantener la relación consistente es necesario expresar todas las combinaciones de los atributos.

| <i>Empleados</i> | | |
|-------------------------|-------------------------|------------------------|
| <u>Nombre</u> | <u>Dirección</u> | <u>Teléfono</u> |
| Ana Almansa | c/ Argentales | 1 |
| Ana Almansa | c/ Argentales | 2 |
| Ana Almansa | c/ Argentales | 3 |
| Ana Almansa | c/ Amanuel | 1 |
| Ana Almansa | c/ Amanuel | 2 |
| Ana Almansa | c/ Amanuel | 3 |

Mientras que las dependencias funcionales impiden que aparezcan ciertas tuplas en las relaciones, las dependencias multivaloradas obligan a ello.

Las dependencias multivaloradas aparecen cuando en un esquema de relación hay varias relaciones 1:N independientes entre sí.

Definición. Dependencias multivaloradas

Dados dos subconjuntos de atributos X e Y y un esquema R , la dependencia multivalorada $X \twoheadrightarrow Y$ (X multidetermina a Y) especifica la siguiente restricción sobre la relación r del esquema R : si existen en r dos tuplas t_1 y t_2 tales que $t_1[X] = t_2[X]$, entonces deben existir dos tuplas, t_3 y t_4 , tales que:

$$t_1[X] = t_2[X] = t_3[X] = t_4[X]$$

$$t_1[Y] = t_3[Y] \text{ y } t_2[Y] = t_4[Y]$$

$$t_2[Z] = t_3[Z] \text{ y } t_1[Z] = t_4[Z], \text{ donde } Z = R - (X \cup Y)$$

Esta definición es más sencilla de lo que parece si se observa el siguiente gráfico. En definitiva se imponen todas las combinaciones de los valores de los atributos Y y Z . Si Z es vacío o sus valores son únicos, necesariamente $t_1 = t_3$ y $t_2 = t_4$, es decir, estamos hablando sólo de dos tuplas.

| | X | Y | Z |
|-------|-----|-----|-----|
| t_1 | ■ | ■ | ■ |
| t_2 | ■ | ■ | ■ |
| t_3 | ■ | ■ | ■ |
| t_4 | ■ | ■ | ■ |

| | <u>X</u> | <u>Y</u> |
|-------|----------|----------|
| t_1 | verde | verde |
| t_2 | verde | rojo |

Informalmente se dice que siempre que existan dos tuplas con valores iguales de X pero distintos de Y , los valores de Y se deben repetir en tuplas separadas por cada valor distinto de Z . En definitiva, con esta restricción se dice que la relación entre X e Y es independiente de la relación entre X y Z .

Ejemplo 20.

En el ejemplo anterior se observan las restricciones multivaloradas

$\{Nombre\} \twoheadrightarrow \{Dirección\}$ y $\{Nombre\} \twoheadrightarrow \{Teléfono\}$.

Debido a la simetría de la definición (se pueden intercambiar los papeles de Y y Z) se deduce que si se cumple $X \twoheadrightarrow Y$, entonces también se cumple $X \twoheadrightarrow Z$, que se representa de forma compacta como $X \twoheadrightarrow Y | Z$.

Definición. Dependencias multivaloradas triviales

Una dependencia multivalorada $X \twoheadrightarrow Y$ se denomina trivial si $Y \subseteq X$ o $X \cup Y = R$.

Se denomina trivial porque no aporta ninguna restricción relevante al esquema. En el primer caso, $Y \subseteq X$, sólo se impone que un subconjunto de los valores de X esté asociado siempre a los valores de X , lo cual es trivialmente cierto. El segundo caso se vio en la definición de dependencia multivalorada.

La cuarta forma normal es una generalización de la forma normal de Boyce/Codd que se aplica a esquemas con dependencias multivaloradas.

Ejemplo 21.

En el ejemplo del tema Restricciones de integridad referido a este tipo de dependencias, que se reproduce a continuación, se observan dos atributos, Dirección y Teléfono, que son multivalorados.

| Empleados | | |
|------------------|------------------|-----------------|
| <u>Nombre</u> | <u>Dirección</u> | <u>Teléfono</u> |
| Ana Almansa | c/ Argentales | 1 |
| Ana Almansa | c/ Argentales | 2 |
| Ana Almansa | c/ Argentales | 3 |
| Ana Almansa | c/ Amanuel | 1 |
| Ana Almansa | c/ Amanuel | 2 |
| Ana Almansa | c/ Amanuel | 3 |

La idea de la cuarta forma normal es descomponer esta relación en otras dos, como se muestra a continuación, preservando las dependencias multivaloradas:

| Direcciones | |
|--------------------|------------------|
| <u>Nombre</u> | <u>Dirección</u> |
| Ana Almansa | c/ Argentales |
| Ana Almansa | c/ Amanuel |

| Teléfonos | |
|------------------|-----------------|
| <u>Nombre</u> | <u>Teléfono</u> |
| Ana Almansa | 1 |
| Ana Almansa | 2 |
| Ana Almansa | 3 |

Definición. Cuarta forma normal

Un esquema de relación R está en cuarta forma normal con respecto a conjunto de dependencias S , tanto funcionales como multivaloradas, sobre R si por cada dependencia multivalorada no trivial $X \twoheadrightarrow Y$ de S^+ , X es superclave de R .

Aunque no vamos a verlo existen algoritmos para obtener descomposiciones en cuarta forma normal.

Aún más formas normales

Hay otras formas normales más exigentes que la 4FN. Las dependencias multivaloradas permiten representar restricciones que no se pueden expresar con dependencias funcionales. Otros tipos de restricciones son las dependencias de reunión que generalizan las dependencias multivaloradas y que conducen a otra forma normal denominada forma normal proyección-reunión, o también denominada quinta forma normal.

Ejemplo 22.

En el siguiente ejemplo se tiene que un proveedor suministra un determinado material a un determinado proyecto. Se encuentra en 4FN porque no hay dependencias multivaloradas y, por lo tanto, no se debería descomponer. De hecho, todos sus atributos forman la clave. Sin embargo, supongamos que se añade la siguiente restricción: si un proyecto usa un material que vende un determinado proveedor y necesita otro material que vende el mismo proveedor, entonces este otro material debe comprarlo al mismo proveedor. Por lo tanto, se deben añadir las dos últimas filas.

| Compras | | |
|------------------|-----------------|-----------------|
| Proveedor | Material | Proyecto |
| Proveedor A | Toner | Proyecto 1 |
| Proveedor A | Papel | Proyecto 2 |
| Proveedor B | Toner | Proyecto 2 |
| Proveedor C | Papel | Proyecto 3 |
| Proveedor B | Clips | Proyecto 1 |
| Proveedor B | Toner | Proyecto 1 |
| Proveedor A | Toner | Proyecto 2 |

No obstante, este tipo de dependencias surgen en muy raras ocasiones y hay que buscar ejemplos forzados.

La forma normal de claves y dominios intenta ser una forma normal “definitiva”. Se dice que un esquema está en forma normal de claves y dominios cuando todas las restricciones de la relación están expresadas como restricciones de clave y de dominio. La comprobación de la consistencia sería fácil de determinar. Sin embargo, la dificultad de expresar restricciones de integridad generales hacen que esta forma normal sea de escasa utilidad.

6.7 Resumen

- Las formas normales buscan que las claves sean las protagonistas en el mantenimiento de la integridad, con el objetivo de que el diseño esté protegido por las claves. La idea es detectar las posibles fuentes de redundancias y anomalías y descomponer la relación.
- Hay dos propiedades que son deseables mantener en el proceso de normalización:
 - Preservar las dependencias funcionales.
 - Preservar las reuniones no aditivas.

La segunda es de importancia fundamental, la primera no siempre se puede asegurar.

El mejor diseño que siempre se puede alcanzar conservando ambas propiedades es la 3FN. En otros casos sólo a veces será posible alcanzar formas normales superiores conservando estas propiedades.

Bibliografía

- [ACPT00] P. Atzeni, S. Ceri, S. Paraboschi y R. Torlone, “Database Systems. Concepts, Languages and Architectures”, McGraw-Hill, 2000.

- [EN00] R. Elmasri y S.B. Navathe, "Fundamentals of Data Base Systems", Addison-Wesley, 2000. Apartados
- [SKS98] A. Silberschatz, H.F. Korth y S. Sudarshan, "Fundamentos de Bases de Datos", 3ª edición, McGraw-Hill, 1998. Apartados
- [Ull98] J.D. Ullman, "Principles of Database and Knowledge Base Systems", Vol. I y II, Computer Science Press, 1998. Apartados