FCEyN UBA

Bases de Datos

Normalización de Relaciones

Índice

1. Introducción - Modelo Relacional

El presente apunte está orientado a explicar sinteticamente los conceptos de dependencias funcionales y normalización. Es una reescritura de apuntes previos de la materia.

Algunas definiciones:

Un **dominio D** es un conjunto de valores atómicos. Por lo que respecta al modelo relacional, atómico significa indivisible.

Una relación se tiene un esquema (o intención de la relación) y una extensión (estado o instancia)

El **esquema de la relación,** que se escribe $R(A_1, A_2, ..., A_n)$ consiste en un nombre de relación R y un conjunto de atributos $A_1, A_2, ..., A_n$. En algunos casos los atributos se escribiremos cómo A, B, C, etc. o como X, Y, Z, etc.

La **aridad** de una relación es la cantidad de atributos que tiene. Un atributo A_i es el nombre del rol que ejerce algún dominio **D** en un esquema de relación. Si D es el dominio de A_i se escribe como $dom(A_i)$.

Una extensión de una relación (o una instancia de R) que la notamos como $r(R)$ o simplemente r es un conjunto de tuplas $t_i(i = 1, 2, ..., m)$, donde cada tupla t_i es, a su vez un conjunto de pares $t_i = \{< A_1 : v_{i1} >, < A_2 : v_{i2} > ... < A_n : v_{in} >\}$ y, para cada par $< A_j : v_{ij} >$, se cumple que v_{ij} es un valor de $dom(A_j)$.

2. Dependencias Funcionales

2.1. Definición

Tomamos a $t_1(X)$ cómo el valor del atributo X en la tupla t_1.

Definición

Decimos que X determina funcionalmente a Y (o que Y es determinado funcionalmente por X) en R y lo notamos $X \rightarrow Y$ si para toda $r(R)$ se verifica que si $t_1(X) = t_2(X)$ entonces necesariamente $t_1(Y) = t_2(Y)$

Si r cumple con todas las dependencias funcionales entonces decimos que r es LEGAL. Las dependencias funcionales son restricciones del dominio del problema. Es posible que un atributo sea determinado funcionalmente por mas de un otro atributo, ej: $X, Y \rightarrow Z$

Ejemplo 1

Supongamos que queremos registrar para una facultad los datos personales de los alumnos, las materias en las que se inscribieron y los exámenes que rindieron.

Para esto definimos el siguiente esquema de relación:

FACULTAD(LU, NOMBRE, MATERIA, IFEC, EFEC, NOTA)

(donde IFEC es la fecha de inscripción y EFEC es la fecha de examen) y el siguiente conjunto de dependencias funcionales F:

- $LU \rightarrow NOMBRE$ (no puede haber dos alumnos con el mismo LU)
- $LU, MATERIA \rightarrow IFEC$ (se pude inscribir una sola vez en cada materia)
- $LU, MATERIA, EFEC \rightarrow NOTA$ (hay una sola nota por examen)

NOTA: Al no estar $LU, MATERIA \rightarrow EFEC$ se puede rendir varias veces la misma materia

2.2. Inferencias de F

Definición Inferencia

Decimos que si F INFIERE f que se nota como $(F \models f)$ toda r que satisface F debe necesariamente satisfacer también f

Ejemplo 2

Del conjunto de dependencias funcionales del Ejemplo 1, podemos inferir:

$F \models LU, MATERIA \rightarrow NOMBRE, IFEC$

2.2.1. Axiomas de Amstrong - Reglas de Inferencia

Sean X e Y conjuntos de atributos.

1. Reflexividad: Si $Y \subseteq X$ entonces $X \rightarrow Y$
2. Aumento: Para cualquier W, si $X \rightarrow Y$ entonces $XW \rightarrow WY$
3. Transitividad: Si $X \rightarrow Y$ e $Y \rightarrow Z$ entonces $X \rightarrow Z$

De 1) inferimos trivialmente que siempre se cumple $X \rightarrow X$.

Por inercia se tiende a pensar que Si $X \rightarrow Y$ entonces $Y \rightarrow X$, pero esto, aunque a veces puede ser verdadero, en general es falso. En lógica se conoce como la falacia de afirmación del consecuente. Que $X \rightarrow Y$ **no** dice nada sobre que Y pueda o no determinar X

2.2.2. Clausura de F (F^+)

Se denomina clausura de un conjunto de dependencias funcionales al conjunto de todas las dependencias funcionales que pueden inferirse del mismo aplicando los axiomas.

$F^+ = \{X \to Y / F \models X \to Y\}$

Ejemplo 3

$R(A, B)$
$F : A \to B$
$F+ = \{A \to B, A \to A, B \to B, AB \to A, AB \to B, AB \to AB, A \to AB\}$

Ejemplo 4

$R(A, B, C)$
$F = \{AB \to C, C \to BB\}$

$F^+ = \{A \to A, AB \to A, AC \to A, ABC \to A, B \to B, AB \to B, BC \to B,$
$ABC \to B, C \to C, AC \to C, BC \to C, ABC \to C, AB \to AB, ABC \to AB,$
$AC \to AC, ABC \to AC, BC \to BC, ABC \to BC, ABC \to ABC,$
$AB \to C, AB \to AC, AB \to BC, AB \to ABC, C \to B, C \to BC, AC \to B, AC \to AB\}$

2.2.3. Reglas Adicionales

1. Unión: Si $X \to Y$ y $X \to Z$ entonces $X \to YZ$
2. Pseudotransitividad: Para cualquier W, Si $X \to Y$ e $YW \to Z$ entonces $XW \to Z$
3. Descomposición: Si $X \to YZ$ entonces $X \to Y$ y $X \to Z$

Las reglas adicionales se demuestran aplicando los axiomas

2.3. Clausura de un Conjunto de Atributos

La clausura de un conjunto de atributos X que notamos X^+, con respecto a un conjunto de dependencias funcionales F, es el conjunto de todos los atributos A tal que $X \to A$, o sea:
$X^+ = \{A \in R / F \models X \to A\}$
Una forma de calcular X+ es computar una secuencia de conjuntos de atributos $X_0, X_1, \ldots$ aplicando las siguientes reglas:

1. X_0 es X
2. X_{i+1} es X_i *unión* el conjunto de atributos A tal que hay alguna dependencia funcional $Y \to Z$ en F, A está en Z e $Y \subseteq Xi$
3. Aplicamos repetidas veces la regla (2) hasta que $X_i = X_{i+1}$

Nota

Como $X = X_0 \subseteq \cdots \subseteq X_i \subseteq R$, y R es finito, eventualmente llegaremos a que $X_i = X_{i+1}$, que es la parada del algoritmo.

2.4. Superclave y Clave

Hay dos conceptos importantes en el diseño de bases de datos relacionales: el concepto de *superclave* y de *clave*. Ambos los utilizaremos ampliamente cuando veamos las formas normales que hacen al diseño correcto. Siempre que se habla de clave o superclave se toma un conjunto de dependencias funcionales asociado a la relación.

SuperClave

Dado un conjunto de atributos X, un esquema de relación R y un conjunto de dependencias funcionales F: decimos que X es **superclave** de R si y solo si $X \to R \in F^+$

Clave

Sera X superclave de R, si no existe ningún $Z \subset X$ tal que $Z \to Y \in F^+$ entonces X es **clave** de R.

Una clave es entonces un conjunto minimal de atributos tal que determina al toda la relación. En el Ejemplo 1, la clave es $\{LU, MATERIA, EFEC\}$ ¿Por qué?

Una relación R puede tener una o varias claves a las que llamaremos en general claves candidatas (CK).

Un ejercicio típico es hallar todas las claves de R. Una forma de hacerlo es computando primero los atributos que no están en ningún lado derecho del conjunto de dependencias funcionales, llamémoslo X. Si $X^+ = R$, X es la única CK. Sino, hay que probar todos los casos. Es decir, comenzamos con $X \cup A$, para cada A. Si $(XA)^+ = R$, XA es CK, y todos los que incluyan a XA serán superclave. Si $(XA)^+ \neq R$, agregamos un atributo más a XA, sea este B, y computamos $(XAB)^+$, y así sucesivamente.

2.4.1. Propiedad de la clausura

Dados F y $X \to Y$, luego $F \models X \to Y$ si y solo si $Y \subseteq X^+$
Si $X^+ = R$ entonces X es superclave de R

Ejemplo 5

Tomando el mismo R y F del Ejemplo 3, tenemos:

$R(A, B)$
$F : A \to B$
$F+ = \{A \to B, A \to A, B \to B, AB \to A, AB \to B, AB \to AB, A \to AB\}$
$B^+ = B$
$A^+ = AB = R$ entonces A es superclave

Ejemplo 6

$R(A, B, C, D, E)$
$F = AB \to C, C \to D, BD \to E$
$AB^+?$

$X_0 = AB$
$X_1 = ABC$ por $AB \to C$
$X_2 = ABCD$ por $C \to D$
$X_3 = ABCDE$ por $BD \to E$

$X^+ = ABCDE = R$, por lo tanto AB es superclave

2.5. Equivalencia de conjuntos de dependencias funcionales

Decimos que dos conjuntos de dependencias funcionales F y G sobre R son equivalentes $(F \equiv G)$ si
$F^+ = G^+$
También si $F \models G$ y $G \models F$ (si F cubre a G y G cubre a F)

Ejemplo 7

$R(A, B, C)$
$F = \{A \to B, B \to C, C \to A\}$
$G = \{B \to A, C \to B, A \to C\}$
$F \equiv G$

2.6. Cubrimiento Minimal

Un recubrimiento minimal de un conjunto de dependencias funcionales F es un conjunto de dependencias Fm tal que $Fm \equiv F$ y además Fm cumple:

1. Todo lado derecho tiene un único atributo (regla de descomposición).
2. Todo lado izquierdo es reducido (no tiene atributos redundantes) ($B \subset X$ es redundante para $X \to A$ si $A \in (X^{\smile}\{B\})^{+}$).
3. No contiene dependencias funcionales redundantes (en general son las que se obtienen por transitividad). $X \to A$ es redundante si $(F - \{X \to A\}) \equiv F$.

.

Pude haber varios cubrimientos minimales para un mismo conjunto de dependencias funcionales F.

Ejemplo 8

$R(A, B, C, D)$
$F = \{A \to BD, B \to C, C \to D, BC \to D\}$

1. $G = \{A \to B, A \to D, B \to C, C \to D, BC \to D\}$
2. $G = \{A \to B, A \to D, B \to C, C \to D, C \to D\}$, B es redundante en $BC \to D$
3. $G = \{A \to B, B \to C, C \to D\}$, $A \to D$ es redundante y $C \to D$ esta duplicada

$Fm = G$

3. Anomalías y Descomposición

Analizaremos ahora los problemas (anomalías) que se pueden presentar en un esquema y las posibles soluciones. Para verlo mas claramente utilizaremos el la relación del Ejemplo 1:

FACULTAD(LU, NOMBRE, MATERIA, IFEC, EFEC, NOTA)

Analicemos los problemas que podría tener el esquema:

1. **Redundancia de información**: El nombre del alumno se repite por cada materia en la que se inscribe y por cada examen que rinda. Lo mismo pasa con el nombre de la materia y la fecha de inscripción si la rinde varias veces.
2. **Anomalías de actualización**: Si un alumno decide cambiar su nombre tenemos que actualizar varias tuplas y podríamos cometer errores u omisiones.

3. **Anomalías de inserción**: No podemos dar de alta a un alumno hasta que se haya inscripto en la primer materia o aún peor hasta que haya rendido el primer examen. Podríamos hacerlo pero tendríamos que poner valores nulos en campos que forman la clave como Materia, IFec y EFec. FACULTAD (lu1, nom1, - , - , -, -) Si se toma un examen y no se presenta nadie también tendríamos que poner valores nulos en algunos campos de la clave. FACULTAD (-, -, -, mat1 , - , efec1, -)

4. **Anomalías de borrado**: Si en algún momento decidimos borrar los datos de los exámenes rendidos por un alumno perderemos los datos personales del mismo.

Antes de continuar demos una definición intuitiva de descomposición:

Descomposición

Descomponer una relación es dividir el esquema en varios subesquemas de tal manera que TODOS los atributos del esquema esten presentes en al menos un subesquema.

3.1. Perdida de Información

Por las anomalías detalladas más arriba decidimos **descomponer** FACULTAD en :

ALUMNO (LU, NOMBRE)

RESULTADO(LU, MATERIA, NOTA)

EXAMEN(MATERIA, EFEC)

INSCRIPTO(LU, MATERIA, IFEC)

Supongamos que hacemos la consulta:
¿En qué fechas dio examen el alumno con LU= “123/01”?
Para responder esto podríamos hacer la siguiente consulta:
$\sigma_{LU=\text{"}123/01\text{"}}(RESULTADO \bowtie_{MATERIA} EXAMEN)$

El resultado no es correcto porque la junta asocia al alumno con todas fechas de examen de las materias que haya rendido. El resultado tiene muchas más tuplas (*espurias*) que las de la respuesta correcta.

Decimos que esta descomposición es con **PERDIDA DE INFORMACION**

Una descomposición correcta sería:

ALUMNO (LU, NOMBRE)

EXAMEN (LU, MATERIA, EFEC, NOTA)

INSCRIPTO (LU, MATERIA, IFEC)

Sea $R = (A_1, A_2, \ldots, A_n)$ y F conjunto de dependencias funcionales y ρ una descomposición de R

$\rho = R_1, R_2, \ldots R_k$

tal que $\bigcup_{i=1}^{k} R_i = R$

Decimos que ρ es una descomposición sin pérdida de información (lossless join o SPI) si para cada instancia r de R que satisface F, se verifica:

$$r = \overset{k}{\underset{i=1}{\bowtie}} \pi_{R_i}(r)$$

Entonces es importante determinar, dados R, F y ρ, si ρ es sin perdida de información ((lossless join, SPI)).

Ejemplo 9

Sean:
$R = (A, B, C)$
$F = \{A \to B\}$
$\rho = \{(A, B); (B, C)\}$

Ocurre que ρ **no** es *lossless join* con respecto a F, por ejemplo, sea:
$r = \{(a_1, b_1, c_1), (a_2, b_1, c_2)\}$
Tenemos que:
$\pi_{AB}(r) = \{(a_1, b_1), (a_1, b_2)\}$
$\pi_{BC}(r) = \{(b_1, c_1), (b_2, c_2)\}$

$$\pi_{AB}(r) \bowtie \pi_{BC}(r) = \{(a_1, b_1, c_1), (a_1, b_1, c_2), (a_2, b_1, c_1), (a_2, b_1, c_2)\}$$

Que es un superconjunto de r.

Si en cambio hacemos $\rho = \{(AB)(AC)\}$ veremos que es LOSSLESS JOIN o SPI.

3.1.1. Descomposición Binaria

Decimos que una descomposición de R, $\rho = (R_1, R_2)$ es *lossless join* con respecto a un conjunto de dependencias funcionales F, si y sólo si:
La dependencia funcional $(R_1 \cap R_2) \to (R_1 - R_2)$ está en F^+
o
La dependencia funcional $(R_1 \cap R_2) \to (R_2 - R_1)$ está en F^+

O sea que si la intersección de los atributos de R_1 y R_2 forman una superclave para uno de los dos esquemas, entonces ρ es SPI.

3.1.2. Algoritmo de TABLEAU

La regla de la descomposición binaria sólo se aplica cuando se descompone una relación en dos subesquemas. Para el caso más general tenemos un algoritmo que lo cubre. Es el algoritmo del *Tableau*:

Dados R, F y $\rho = (R_1, R_2, \ldots, R_k)$, inicialmente se construye un tableau (matriz) T inicial T_0 donde las columnas son los atributos y las filas los subesquemas de ρ. Luego completamos el tableau con símbolos distinguidos (a_j) si $A_j \in R_i$ y con los que denominamos no distinguidos (b_{ij}) si $A_j \notin R_i$..

Luego vamos modificando el tableau aplicando las dependencias funcionales de las siguiente forma, para cada $X \to A$ si $fila_i[X] = fila_h[X]$ y $fila_i[A] \neq fila_h[A]$ hacemos

(i) si $fila_i[A] = a_j$ hacemos $fila_h[A] = a_j$ o

(ii) si $fila_i[A] = b_{ij}$ y $fila_h[A] = b_{hj}$ hacemos $fila_h[A] = b_{ij}$

Así vamos generando una serie $T_0, T_1, T_2, \ldots, T^*$. terminamos cuando no se puedan hacer más cambios en el tableau final (T^*) aplicando las dependencias funcionales. Si T^* contiene una fila con todos símbolos distinguidos entonces ρ es **SPI**, de lo contrario ρ es con pérdida de información.

Ejemplo 10

$R = (A, B, C, D)$
$F = \{A \to B, AC \to D\}$
$\rho = \{(A, B), (A, C, C)\}$

Hacemos T_0

	A	B	C	D
AB	a_1	a_2	b_{13}	b_{14}
ACD	a_1	b_{22}	a_3	a_4

Usamos la dependencia $A \to B$ por lo que tendríamos que reemplazar b_{22} con a_2 lo cual nos lleva al tableau T_1

	A	B	C	D
AB	a_1	a_2	b_{13}	b_{14}
ACD	a_1	a_2	a_3	a_4

Cómo la segunda fila tienen todos elementos distinguidos podemos concluir que ρ es **SPI**.

Ejemplo 11

$R = (A, B, C, D, e)$
$F = \{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$
$\rho = \{(A, D), (A, B), (B, E), (C, D, E), (A, E)\}$

Hacemos $\boldsymbol{T_0}$ incializando con los símbolos distinguidos y los no distinguidos.

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{23}	b_{24}	b_{25}
BE	b_{31}	a_2	b_{33}	b_{34}	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	b_{53}	b_{54}	a_5

Usando $A \to C$ debemos igualar b_{23}, b_{53} con b_{13}, $\boldsymbol{T}_1$ queda:

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{13}	b_{24}	b_{25}
BE	b_{31}	a_2	b_{33}	b_{34}	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	b_{13}	b_{54}	a_5

Usando $B \to C$ debemos igualar b_{33} a b_{13} quedando $\boldsymbol{T}_2$ de la siguiente forma

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{13}	b_{24}	b_{25}
BE	b_{31}	a_2	b_{13}	b_{34}	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	b_{13}	b_{54}	a_5

Usando $C \to D$ debemos igualar b_{24}, b_{34}, b_{54} con a_4 quedando $\boldsymbol{T}_3$

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{13}	a_4	b_{25}
BE	b_{31}	a_2	b_{13}	a_4	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	b_{13}	a_4	a_5

Usando $DE \to C$ hacemos $b_{13} = a_3$, asi queda $\boldsymbol{T}_4$

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{13}	a_4	b_{25}
BE	b_{31}	a_2	a_3	a_4	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	a_3	a_4	a_5

Para finalizar usamos $CE \to A$ quedando $\boldsymbol{T}_5$

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{13}	a_4	b_{25}
BE	a_1	a_2	a_3	a_4	a_5
CDE	a_1	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	a_3	a_4	a_5

Cómo nos queda la tercer fila con todos símbolos distinguidos podemos afirmar que la descomposición ρ es **SPI**.

3.2. Perdida de Dependencias Funcionales

Además de preservar la información (SPI) una descomposición ρdebería preservar las dependencias funcionales, lo que llamamos descomposición sin pérdidas de dependencias funcionales (SPDF).

Proyección de un conjunto de DF

Decimos que la proyección de un conjunto de dependencias funcionales F sobre un conjunto de atributos Z, que se escribe $\pi_Z(F)$, es el conjunto de dependencias funcionaes $X \to Y \in F^+$ tal que $XY \subseteq Z$.

Dados R y $\rho = (R_1, R_2, \ldots, R_k)$ y F entonces ρ preserva F si la unión de todas las dependencias funcionales en $\pi_{R_i}(F)$ para $i = 1 \ldots k$ implica lógicamente F
$F^+ = (\bigcup_{i=1}^{k} \pi_{R_i}(F))^+$

Tomar en cuenta que ρ podría ser SPI y SPDF (o ninguna de las dos) o SPI pero no SPDF (o al revés)

3.2.1. Algoritmo para Testear Perdidas de Dependiencias

Hay un algoritmo para probar si una descomposición ρ preserva las dependencias funcionales sin necesidad de calcular F^+.

La idea es computar $Z \cup ((Z \cap R_i)^+ \cap R_i)$ donde Z inicialmente es igual al lado izquierdo de la dependencia funcional $X \to Y$ que se desea testear.
La clausura de $(Z \cap R_i)$ es con respecto a F.

Dados R, F y ρ, queremos verificar si se preserva $X \to Y$,

Z:= X;
mientras *Z cambie* **hacer**
  para $i \leftarrow 1$ **a** k **hacer**
    $Z \cup ((Z \cap R_i)^+ \cap R_i)$
  fin
fin

Si al finalizar $Y \subseteq Z$ entonces $X \to Y$ está en F^+ Si esto esto se cumple para toda dependencia funcional $X \to Y$ entonces podemos afirmar que ρ es SPDF. Obviamente se podría finalizar el algoritmo si se detecta que $Y \subseteq Z$ en algún paso.

4. Formas Normales

Existe un conjunto de propiedades o *formas normales* para los esquemas de relación sobre un conjunto de dependencias funcionales. Las formas normales nos

permiten evitar las anomalías que mencionamos anteriormente. Las mas importantes son la tercera forma normal y la forma normal de Boyce-Cood.

Por otro lado las formas normales nos dan un piso para el proceso de descomposición (cuando todos los subesquemas quedan en la forma normal deseada paramos). La idea original era partir de una relación universal e ir descomponiendo hasta llega a la forma normal deseada, o partir de las dependencias funcionales y mediante un proceso de síntesis llegar los esquemas necesarios. Pero también es posible analizar esquemas, verifica que forma normal se encuentran y descomponerlos si hiciera falta.

Como ya mencionamos un esquema R puede tener una o varias claves candidatas (CC) La clave primaria (PK) se elige arbitrariamente entre las CC. Al resto las llamaremos claves secundarias

Atributos Primos

Decimos que un atributo es PRIMO si es miembro de ALGUNA clave candidata, en caso contrario decimos que es NO PRIMO

Decimos que $X \to Y$ es dependencia funcional PARCIAL (o que Y depende parcialmente de X) si para algún subconjunto $Z \subset X$, se verifica que $Z \to Y$. En caso contrario decimos que la dependencia funcional es TOTAL (o que Y depende totalmente de X)

4.1. Primera Forma Normal

Dada nuestra definición de *Relación* para el modelo relacional, todas las relaciones estarían en primera forma normal. El objetivo histórico de su definición es evitar que haya atributos multivaluados o compuestos.

Podemos decir que una relación estaría en primera forma normal si y solo si todos sus atributos son atómicos.

4.2. Segunda Forma Normal

Un esquema de relación R esta en segunda forma normal (2FN) si **todo** atributo no primo A en R **no** es parcialmente dependiente de **alguna** clave de R.

O en forma equivalente: R está en 2FN si **todo** atributo **no primo** A en R es **totalmente** dependiente de **todas** las claves de R.

4.3. Tercera Forma Normal

Un esquema de relación R esta en tercera forma normal (3FN) si para **toda** dependencia funcional no trivial $X \to A$ sobre R, se cumple que

1. X es superclave de R o
2. A es primo

O en forma equivalente: R está en 3FN si para **toda** dependencia funcional no trivial $X \to Y$ sobre R, o bien X es una superclave de R o Y es un subconjunto de alguna clave de R.

4.4. Forma Normal de BOYCE-CODD (FNBC)

Un esquema de relación R está en **FNBC** si para **toda** dependencia funcional no trivial $X \to A$ sobre R, X es una superclave de R

O en forma equivalente: R está en FNBC si para **toda** dependencia funcional $X \to Y$ en F^+, o bien $Y \subseteq X$ o X es una superclave de R.

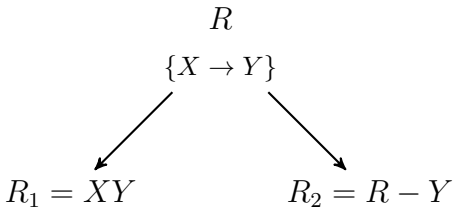
4.5. Propiedades de las Formas normales

- La validez debe ser para F^+ (propiedad para FNBC y 3FN: si todas las dependencias funcionales tienen lado derecho simple, entonces no hay violación en F^+)
- Si R está en FNBC entonces también está en 3FN y en 2FN y si R está en 3FN entonces también está en 2FN
- Todo esquema se puede descomponer en 3FN tal que sea SPI y SPDF
- Hay esquemas que no se pueden descomponer en FNBC y que sean SPDF
- Todo esquema de 2 atributos esta en FNBC. ¿Por qué?

4.6. Algoritmo de descomposción SPI en FNBC

Este algoritmo obtiene una descomposición ρ de R partiendo R aplicando la propiedad de descomposición binaria. El ρ resultante es siempre SPI, pero a veces no es SPDF.

Si hay una dependencia funcional $X \to Y$ que viola FNBC se parte R en R_1 y R_2 de la siguiente forma:

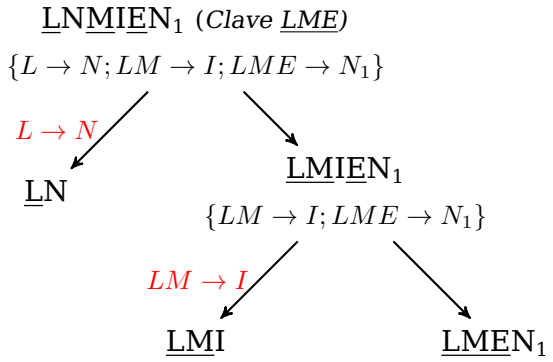


Vemos que esta descomposición es SPI porque aplica la regla de descomposición binaria:
$R1 \cap R2 = X$
$R1 - R2 = Y$

Ejemplo 12

Volvamos a considerar el esquema de relación del ejemplo 1 (para simplificar tomamos la primer letra para identificar los atributos):
FACULTAD(L, N, M, I, E, N_1)
$F = \{(L \to N); (LM \to I); (LME \to N_1)\}$
Clave:$\{L, M, E\}$

Para armar el árbol de descomposición elegimos una dependencia funcional que viole la FNBC (en el primer caso usamos $L \to N$), es la dependencia que anotamos en rojo al lado de la flecha. Usando esa dependencia descomponemos, abajo de cada nodo figuran las dependencias funcionales que van quedando. Siempre hay que volver a revisar cual es la clave de cada descomposición para poder determinar cual dependencia funcional violaría la FNBC.



Finalmente $\rho = \{(L, N), (L, M, I), (L, M, E, N_1)\}$ esta en FNBC y es SPI

4.7. Algoritmo de descomposción SPI y SPDF en 3FN

Este algoritmo obtiene una descomposición ρ de R por *síntesis* a partir de una ***cobertura minimal*** de F. La descomposición resultante ρ cumple con ser SPI y SPDF. Una vez obtenida la cobertura minimal F_M se hace los siguiente:

1. Se crea una subesquema (X, A) para cada dependencia $X \to A$ en F_M.
2. Unificar los que provienen de DFs que tienen igual lado izquierdo, o sea creamos los subesquemas $(X, A_1, A_2, \ldots, A_n)$, donde $X \to A_1, \ldots, X \to A_n$ están en F_M.
3. Si ninguno de los esquemas resultantes contiene una clave se agrega uno con los atributos de alguna clave

4. Eliminar esquemas redundantes: Si alguno de los esquemas resultantes esta contenido totalmente en otro, eliminarlo:

Ejemplo 13

Utlizamos nuevamente el ejemplo 1.
FACULTAD(L, N, M, I, E, N_1)
$F = \{(L \to N); (LM \to I); (LME \to N_1)\}$
Clave:$\{L, M, E\}$
F es cobertura minimal porque:

- Los lados derechos son de un solo atributo
- No hay atributos redundantes en los lados izquierdos: $L^+ = LN$, $M^+ = M$, $E^+ = E$
- No hay dependencias funcionales redundantes

Entonces creamos un esquema de relación por cada una de las dependencias, la descomposición queda $\rho = \{(L, N), (L, M, I), (L, M, E, N_1)\}$.

Está en 3FN, es SPDF (no se pierde ninguna de las dependencias funcionales) y también es SPI.

Referencias

[1] *Database Systems: The Complete Book (2 ed.).* Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer Widom. 2008. Prentice Hall Press, Upper Saddle River, NJ, USA.

[2] *Principles of Database and Knowledge-Base System. Volume I* Jeffrey D. Ullman 1998

[3] *Fundamentals of Database Systems. 7th Ed.* Ramez Elmasri, Shamkant B.Navathe Pearson, 2016