Universidad de Buenos Aires
Facultad de Ciencias Exactas y Naturales
Departamento de Computación
Cátedra de Base de Datos

Autor: Alejandro Eidelsztein

<u>NORMALIZACIÓN</u>

1. MODELO RELACIONAL

R : Esquema de relación

A_1, A_2, ..., An o A, B, C,..., etc. atributos de R

r: Instancia de R

t_1, t_2, ..., etc. tuplas de r

2. DEPENDENCIAS FUNCIONALES (DF)

2.1. DEFINICION DE DF

Decimos que vale X --> Y en R si para toda r se verifica que si $t_1(X)=t_2(X)$ entonces necesariamente $t_1(Y)=t_2(Y)$

Decimos que “X determina funcionalmente Y”, o que “Y es determinado funcionalmente por X”
X, conjunto de atributos de R, lado izquierdo.
Y, conjunto de atributos de R, lado derecho
X e Y no tienen que ser necesariamente disjuntos
t_1 y t_2 dos tuplas cualesquiera de r

Si r cumple con todas las dependencias funcionales entonces decimos que r es LEGAL

Las dependencias funcionales las establece el diseñador de la BD

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)

y el siguiente conjunto de dependencias funcionales F:

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

NOTA: Al no estar LU, MATERIA --> EFEC se puede rendir varias veces la misma materia

2.2. INFERENCIAS DE F

Decimos que si F INFIERE f (F |= f) toda r que satisface F debe necesariamente satisfacer también f

EJEMPLO 2:

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

F |= LU, MATERIA --> NOMBRE, IFEC

2.3. REGLAS DE INFERENCIA (AXIOMAS DE ARMSTRONG)

1) Reflexividad: Si Y ⊆ X entonces X-- > Y
2) Aumento: Para cualquier W, si X-- > Y entonces XW-- > WY
3) Transitividad: Si X-- > Y e Y-- > Z entonces X-- >Z

De 1) inferimos las triviales

Se sigue que siempre se cumple X-- > X

Por inercia se tiende a pensar que Si X-- >Y entonces Y-- > X, pero esto, aunque a veces puede ser verdadero, en general es falso.

2.4. CLAUSURA DE F (F+)

Conjunto de todas las dependencias funcionales que pueden inferirse de F aplicando los axiomas

F+ = {X -- > Y / F |= X -- > Y}

EJEMPLO 3:

R(A,B)
F: A -- > B

F+ = {A-->B, A-->A, B-->B, AB-->A, AB-->B, AB-->AB, A-->AB}

EJEMPLO 4:

R(A,B,C)
F= {AB -- >C, C-- >B}

F+ = {A-->A, AB-->A, AC-->A, ABC-->A, B-->B, AB-->B, BC-->B, ABC-- >B, C-- > C, AC-- >C, BC-- >C, ABC-- >C, AB-- >AB, ABC-- >AB, AC-- >AC, ABC-- >AC, BC-- >BC, ABC-- >BC, ABC-- >ABC, AB-- >C, AB-- >AC, AB-- >BC, AB-- >ABC, C-- >B, C-- >BC, AC-- >B, AC-- >AB}

2.5. REGLAS ADICIONALES

4) Unión: X -- > Y y X -- > Z entonces X -- > YZ
5) Pseudotransitividad: Para cualquier W, X -- > Y e YW -- > Z entones XW -- > Z
6) Descomposición: X -- > YZ entonces X -- > Y y X -- > Z

Las reglas adicionales se demuestran aplicando los axiomas

EJEMPLO 5:

Demostraremos la regla de unión:

1. X -- > Y (dada)
2. X -- > Z (dada)
3. X -- > XY (aumento de 1 con X)
4. XY -- > YZ (aumento de 2 con Y)
5. X -- > YZ (transitividad de 3 y 4)

Si es falsa se demuestra con una instancia que sea contraejemplo:

EJEMPLO 6:

{X -- > Z, Y -- > Z} |= X -- > Y ?

X	Y	Z
1	2	5
1	3	5
2	2	5
2	3	5

2.6. CLAUSURA DE UN CONJUNTO DE ATRIBUTOS X (X+)

X+ con respecto a F, es el conjunto de todos los atributos A tal que X -- > A, o sea:

X+ = {A $\in$ R / F |= X -- > A}

Una forma de calcular X+ es computar una secuencia de conjuntos de atributos X_0, X_1, … aplicando las siguientes reglas:

1) X_0 es X
2) X_{i+1} es Xi Unión el conjunto de atributos A tal que hay alguna dependencia funcional Y --> Z en F, A está en Z e Y $\subseteq$ Xi

Aplicamos repetidas veces la regla (2) hasta que Xi = X_{i+1}

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

2.7. PROPIEDAD DE LA CLASURA

Dados F y X -- > Y, luego F |= X -- > Y Sii Y $\subseteq$ X+

Si X+ = R entonces X es superclave de R

EJEMPLO 7:

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

R(A,B)
F: A -- > B
F+ = {A-- >B, A-- >A, B-- >B, AB-- >A, AB-- >B, AB-->AB, A-->AB}

B+= B
A+= AB = R, por lo tanto A es clave

EJEMPLO 8:

R(A,B,C,D,E)
F= {AB-- >C, C-- >D, BD-- >E}
AB+ ?

X_0= AB	por AB-- > C
X_1= ABC	por C-- >D
X_2= ABCD	por BD-- >E
X_3= ABCDE	

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

2.8. SUPERCLAVE Y CLAVE

Si X --> R entonces decimos que X es SUPERCLAVE
Si además no existe ningún Z ⊂ X tal que Z --> R entonces X también es CLAVE

En el Ejemplo 1, la clave es {LU, MATERIA, EFEC} ¿Por qué?

R puede tener una o varias claves a las que llamaremos en general *claves candidatas* (CC)

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, llamémoslo X. Si X+ =R, X es la única CC. Sino, hay que probar todos los casos. Es decir, comenzamos con X ∪ A, para cada A. Si XA+ =R, XA es CC, y todos los que incluyan a XA serán superclave. Si XA+ ≠R, agregamos un atributo más a XA, sea este B, y computamos XAB+, y así sucesivamente.

2.9. EQUIVALENCIA DE CONJUNTOS DE DEPENDENCIAS FUNCIONALES

Decimos que dos conjuntos de dependencias funcionales F y G sobre R son equivalentes (F ≡ G) si

F+ = G+

También si F |= G y G |= F (si F cubre a G y G cubre a F)

EJEMPLO 9:

R(A,B,C)

F= {A-- > B, B-- >C, C-- >A}
G= {B-- >A, C-- >B, A-- >C}

F ≡ G

2.10. CUBRIMIENTO MINIMAL

Dado F buscamos un Fm tal que Fm ≡ F y además Fm tiene:

1) Todo lado derecho tiene un único atributo (regla de descomposición)
2) Todo lado izquierdo es reducido (no tiene atributos redundantes)
(B ⊂ X es redundante para X-- >A si A ∈ (X – {B})+)
3) No contiene dependencias funcionales redundantes (en general las que se obtienen por transitividad, X-- > A es redundante si (F – {X-- >A}) ≡ F)

Pude haber varios cubrimientos minimales para un mismo F

EJEMPLO 10:

R(A,B,C,D)

F= {A-- >BD, B-- >C, C-- >D, BC -- >D}

1) G= {A-- >B, A-- >D, B-- >C, C-- >D, BC-- >D}
2) G= {A-- >B, A-- >D, B-- >C, C-- >D, C-- >D}, B es redundante en BC-- >D
3) G= {A-- >B, B-- > C, C-- > D}, A-- >D es redundante y C-- >D esta duplicada

Fmin := G

3. PERDIDA DE INFORMACION

Analizaremos ahora los problemas (*anomalías*) que se pueden presentar en un esquema y las posibles soluciones.

EJEMPLO 11:

Volvamos al esquema 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 una alumna se casa y decide cambiar por su apellido de casada 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 bajas: Si en algún momento decidimos borrar los datos de los exámenes rendidos por un alumno perderemos los datos personales del mismo.

Por lo mencionado 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:

$\sigma_{\text{LU="123/01"}}$ (RESULTADO $|X|_{\text{MATERIA}}$ 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.

La descomposición correcta es:

ALUMNO (LU, NOMBRE)
EXAMEN (LU, MATERIA, EFEC, NOTA)
INSCRIPTO (LU, MATERIA, IFEC)

¿Por qué?

3.1. FORMALIZANDO

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

Una descomposición ρ 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 (lossless join, SPI) si para cada instancia r de R que satisface F, se verifica:

$r = \bowtie_{i=1,k} \Pi_{R_i}(r)$

Entonces es importante determinar, dados R, F y ρ, si ρ es SIN PERDIDA (lossless, SPI)

EJEMPLO 12:

Sean R= (A,B,C)
F= {A-- >B}
ρ = (AB, BC)

No es lossless join con respecto a F, por ejemplo:

$r = \{a_1 b_1 c_1,\ a_2 b_1 c_2\}$

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

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

Que es un superconjunto de r.

Si en cambio hacemos ρ = (AB, AC) veremos que es LOSSLESS JOIN.

3.2. PROPIEDAD DE DESCOMPOSICION BINARIA

Decimos que una descomposición ρ de R, ρ = (R1,R2) es Lossless Join con respecto a un conjunto de dependencias funcionales F, Sii

1) La dependencia funcional (R1 ∩ R2) -- > (R1 – R2) está en **F+**

o

2) La dependencia funcional (R1 ∩ R2) -- > (R2 – R1) está en **F+**

O sea que si la intersección de los atributos de R1 y R2 forman una superclave para uno de los dos esquemas, entonces ρ es SIN PERDIDA
Ver Ejemplo 12.

3.3. ALGORITMO DEL TABLEAU

Hay un algoritmo más general para determinar si una descomposición es lossless. Es el algoritmo del Tableau:

Dados R, F y $\rho=(R_1, R_2,\ldots,R_k)$, inicialmente se construye un tableau T inicial T_0 donde las columnas son los atributos y las filas los subesquemas de ρ.

Luego completamos el tableau con símbolos distinguidos (aj) si $A_j \in R_i$ o con no distinguidos (bij) si $A_j \notin R_i$.

Luego vamos modificando el tableau aplicando las dependencias funcionales de la siguiente forma: Para cada X-- >A si (fila_i[X]=fila_h[X]) y (fila_i[A]≠fila_h[A]), entonces:

i) si fila_i[A]=aj hacemos fila_h[A]=aj o
ii) si fila_i[A]=bij y fila_h[A]=bhj hacemos fila_h[A]=bij

Así vamos generando una serie T_0, T_1, T_2, ..., T*.

Paramos cuando no se puedan hacer más cambios en el tableau final (T*) por aplicación de las dependencias funcionales.

Si T* contiene una fila con todos símbolos distinguidos entonces ρ es SPI, de lo contrario ρ es con pérdida.

T_0:

	A_1	A_2	.	.	A_n
R_1					
R_1					
.					
.					
R_k					

Veamos algunos ejemplos:

EJEMPLO 13:

R= ABCD
F= {A-- >B, AC-- >D}
ρ = (AB, ACD)

	A	B	C	D	
AB	a_1	a_2	b_{13}	b_{14}	
ACD	a_1	b_{22}	a_3	a_4	
					A-- >B ($b_{22}=a_2$)

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

Vemos que la segunda fila tiene todos símbolos distinguidos y por lo tanto ρ es SPI.

EJEMPLO 14:

R= ABCDE
F= {A-- >C, B-- >C, C-- >D, DE-- >C, CE-- >A}
ρ = (AD, AB, BE, CDE, AE)

	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

A-- >C ($b_{23}=b_{53}=b_{13}$)
B-- >C ($b_{33}=b_{13}$)

	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

C-- >D ($b_{24}=b_{34}=b_{54}=a_4$)
DE-- >C ($b_{13}=a_3$)
CE-- >A ($b_{31}=b_{41}=a_1$)

	A	B	C	D	E	
AD	a_1	b_{12}	a_3	a_4	b_{15}	
AB	a_1	a_2	a_3	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	

Vemos que la tercera fila tiene todos símbolos distinguidos y por lo tanto ρ es SPI.

4. PERDIDA DE DEPENDENCIAS FUNCIONALES

Además de preservar la información (SPI) una descomposición ρ debería preservar las dependencias funcionales (SPDF)

4.1. PROYECCION DE UN CONJUNTO DE DEPENDENCIAS FUNCIONALES

Dados R, y $\rho = (R_1, R_2, \ldots, R_k)$ decimos que la proyección de F sobre un conjunto de atributos Z ($\Pi_Z (F)$) es el conjunto de dependencias funcionales X-- > Y $\in$ F+ tal que $XY \subseteq Z$

ρ preserva F si la unión de todas las dependencias funcionales en $\Pi_{Ri} (F)$ para i= 1, 2, …., k implican F , o sea:

$F^+ = (\mathbf{U}_{i=1,k} \ \Pi_{Ri} (F))^+$

NOTA: ρ podría ser SPI pero no SPDF o al revés

EJEMPLO 15:

a)
R (A,B,C)
F= {AB-- > C, C-- > A}
ρ = (BC, AC)

ρ es SPI pero sin embargo no es SPDF (no se preserva AB-- > C)

b)
R(A,B,C,D)
F= {A-- >B, C-- >D}
ρ = (AB, CD)

ρ no es SPI pero sin embargo es SPDF

EJEMPLO 16: (para hacer énfasis en que hay proyectar F+)

Dados:
R(A,B,C,D)
F= {A-- >B, B-- >C, C-- >D, D-- >A}
ρ = (AB, BC, CD)

ρ es SPDF?

Vemos que en F+ cada atributo determina a todos los otros

Intuitivamente podemos suponer que cuando proyectamos F se pierde D-- >A pero esto no es así porque cuando decimos que proyectamos F estamos proyectando F+

Si hacemos los cálculos veremos que finalmente:
{B-- >A, C-- >B, D-- >C} |= D-- >A

4.2. ALGORITMO PARA TESTEAR PERDIDA DE DEPENDENCIAS

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

La idea es computar Z ∪ ((Z ∩ Ri) + ∩ Ri) donde Z inicialmente es igual al lado izquierdo de la dependencia funcional X -- > Y que se desea testear.
(La clausura de (Z ∩ Ri) es con respecto a F)
Dados R, F y ρ, queremos verificar si se preserva X -- > Y

Procedimiento:

Z:= X;
while Z cambie **do**
for i:= 1 **to** k **do**
  Z:= Z ∪ ((Z ∩ Ri)+ ∩ Ri);

Si Z es el resultado final y, además, Y ⊆ Z luego X -- > Y Є ($\bigcup_{i=1,\ldots,k}$ Π Ri (F))+
Si esto esto se cumple para toda dependencia funcional X -- > Y entonces podemos afirmar que ρ es SPDF.

EJEMPLO 17:

Sea R= (A,B,C,D,E)
F= {AB-- >C, B-- >D, C-- >E, D-- >C, E-- >B}
ρ= (ABC, CD, BDE)

Queremos verificar si se preserva C-- >E

Aplicamos el algoritmo:

Comenzamos haciendo Z = C

Luego iteramos (R1=ABC, R2=CD, R3=BDE):

$C \cup ((C \cap ABC)^+ \cap ABC) = BC$

$BC \cup ((BC \cap CD)^+ \cap CD) = BCD$

$BCD \cup ((BCD \cap BDE)^+ \cap BDE) = BCDE$

Vemos que si seguimos interando no se producen más cambios al valor obtenido Z = BCDE. También verificamos que se cumple $Y \subseteq Z$, ya que $E \subseteq BCDE$.
Luego C-- >E se preserva.

5. FORMAS NORMALES

Se obtienen por DESCOMPOSICION de R (esquema universal) o por SINTESIS a partir de las dependencias funcionales

La idea es evitar la redundancia de información que es lo que produce las anomalías de altas, bajas y modificación.

Por otro lado las formas normales nos dan un piso para el proceso de descomposición (cuando todos los subesquemas quedan en la FN deseada paramos)

5.1. CLAVES

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*

5.2. ATRIBUTOS PRIMOS

Decimos que un atributo es PRIMO si es miembro de ALGUNA clave candidata

Si no, decimos que es NO PRIMO

5.3. DEPENDENCIA FUNCIONAL PARCIAL (ATRIBUTOS PARCIALMENTE DEPENDIENTES)

Decimos que X-- > Y es dependencia funcional PARCIAL (o que Y depende parcialmente de X) si para algún $B \in X$ $(X - \{B\})$ -- > Y

Si no, decimos que la dependencia funcional es TOTAL (o que Y depende totalmente de X)

5.4. SEGUNDA FORMA NORMAL (2FN)

R esta en 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.

5.5. TERCERA FORMA NORMAL (3FN)

R esta en 3FN si para TODA dependencia funcional no trivial X-- > A sobre R,

a) X es superclave de R o
b) A es primo

O en forma equivalente:

R está en 3FN si para TODA dependencia funcional no trivial X-- >Y sobre R, o bien X es una superclave de R o Y es un subconjunto de alguna clave de R.

5.6. FORMA NORMAL DE BOYCE-CODD (FNBC)

R está en FNBC si para TODA dependencia funcional no trivial X-- > A sobre R, X es una superclave de R

O en forma equivalente:
R está en FNBC si para TODA dependencia funcional X-- >Y en F+, o bien Y $\subseteq$ X o X es una superclave de R.

Algunas propiedades para tener en cuenta:

- 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 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?

EJEMPLO 18:

Veremos un ejemplo en donde no se puede hallar una descomposición en FNBC que sea SPDF

DIRECCION (BARRIO, CIUDAD, CPOSTAL)

Dependencias Funcionales:
CPOSTAL -- > CIUDAD
BARRIO, CIUDAD -- > CPOSTAL

Claves:
{CPOSTAL, BARRIO}
{BARRIO, CIUDAD}

La dependencia funcional CPOSTAL -- > CIUDAD viola la FNBC, pero ninguna descomposición de este esquema va a preservar la dependencia funcional BARRIO, CIUDAD -- > CPOSTAL

En forma más general, dados R(A,B,C) y F= {AB-- >C, C-- >A o C-- >B} no es posible hallar una descomposición de R que esté en FNBC y que sea SPDF.

EJEMPLO 19:

De descomposición sin usar algoritmos:

Volvemos a considerar el Ejemplo 1:

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

F:
LU --> NOMBRE (el nro. de libreta es único)
LU, MATERIA --> IFEC (se pude inscribir una sola vez en cada materia)
LU, MATERIA, EFEC --> NOTA (hay una sola nota por examen)

Clave: {LU, MATERIA, EFEC}

No está ni en 2FN

Descomponemos:

ALUMNO (LU, NOMBRE), clave: {LU}, está en FNBC

RESTO (LU, MATERIA; IFEC; EFEC; NOTA), clave: {LU,MATERIA,EFEC}, no está ni en 2FN

Volvemos a descomponer:

INSCRIPTO (LU, MATERIA, IFEC) clave: {LU, MATERIA}, está en FNBC

EXAMEN (LU, MATERIA, EFEC, NOTA), clave: {LU,MATERIA,EFEC}, está en FNBC

Vemos que ρ = (ALUMNO, INSCRIPTO, EXAMEN) está en FNBC

¿ ρ es SPI? ¿ ρ es SPDF?

5.7. ALGORITMO PARA OBTENER UNA DESCOMPOSICION EN FNBC SPI

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 -- > Y que viola FNBC se parte R en R1 y R2 de la siguiente forma:

```
              R
            /   \        X -- > Y
           /     \
          /       \
   R1= XY        R2= R-Y
```

Vemos que esta descomposición es SPI porque aplica la regla de descomposición binaria:
R1 ∩ R2 = X
R1 – R2 = Y

EJEMPLO 20:

Volvemos a considerar el Ejemplo 1:

FACULTAD (L, N, M, I, E, N1)
F= {L-- >N, LM-- >I, LME-- >N1}
CLAVE: {L,M,E}

```
L  N  M  I  E  N1
     /     \
    /        \      L -- > N   (viola la FNBC)
  /           \
L N          L M I E N1
                 /     \
               /         \    LM -- > I   (viola la FNBC)
             /             \
        L M I            L M E N1
```

ρ = (LN, LMI, LMEN1) esta en FNBC y es SPI

5.8. ALGORITMO PARA OBTENER UNA DESCOMPOSICION EN 3FN SPI Y SPDF

Este algoritmo obtiene una descomposición ρ de R por SINTESIS a partir de una COBERTURA MINIMAL de F. El ρ resultante es SPI y SPDF.

1) Cada dependencia funcional se convierte en un esquema (las que tienen igual lado izquierdo se juntan)

2) Si ninguno de los esquemas resultantes contiene una clave se agrega uno con los atributos de alguna clave

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

EJEMPLO 21:

Veamos otra vez el Ejemplo 1:

FACULTAD (L, N, M, I, E, N1)
F= {L-- >N, LM-- >I, LME-- >N1}
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 ρ = (LN, LMI, LMEN1) está en 3FN, es SPDF (están todas las dependencias funcionales) y también es SPI porque R3=LMEN1 contiene todos los atributos de la clave.

6. DEPENDENCIAS MULTIVALUADAS (DMV)

EJEMPLO 22:

Sea el siguiente esquema:

PROFESIONAL(NOMBRE, TITULO, IDIOMA)

Donde almacenamos la información correspondiente a un grupo de profesionales con los títulos que cada uno posee y los idiomas que cada uno domina.

Sea la siguiente instancia r1:

NOMBRE	TITULO	IDIOMA
hugo	físico	inglés
hugo	matemático	francés
hugo	físico	francés
hugo	matemático	inglés
maría	médica	alemán
maría	médica	italiano
luis	abogado	portugués
luis	abogado	inglés
luis	matemático	portugués
luis	matemático	inglés

Cuál es la clave de este esquema?

Aunque PROFESIONAL no tiene otras dependencias funcionales que las triviales (y por lo tanto está en FNBC) observamos que hay redundancia de información: Tenemos que repetir el título por cada idioma que el profesional sabe y de igual forma tenemos que repetir el idioma por cada título que tiene.
Se presentan anomalías de inserción: Por ejemplo si Hugo aprende un nuevo idioma debemos agregar dos tuplas, lo mismo si obtiene un nuevo título. Si queremos agregar un nuevo profesional seguramente deberemos agregar más de una tupla.

Esto es así porque (NOMBRE-TITULO) y (NOMBRE-IDIOMA) son independientes entre si.

Vemos que este esquema se puede decomponer en **(PROF-TIT)** y **(PROF-IDIOMA)**:

r1.1:

NOMBRE	TITULO
hugo	físico
hugo	matemático
maría	médica
luis	abogado
luis	matemático

r1.2:

NOMBRE	IDIOMA
hugo	inglés
hugo	francés
maría	alemán
maría	italiano
luis	portugués
luis	inglés

y que esta descomposición es SPI dado que r1.1 |X| r1.2 = r1
Si en cambio ahora consideramos esta otra instancia r2 (donde eliminamos algunas tuplas), vemos que la descomposición propuesta no es SPI dado que r2.1 |X| r2.2 es un superconjunto de r2.

r2:

NOMBRE	TITULO	IDIOMA
hugo	físico	inglés
hugo	matemático	francés
hugo	físico	francés

maría	médica	alemán
maría	médica	italiano
luis	abogado	portugués
luis	abogado	inglés
luis	matemático	portugués

r2.1:

NOMBRE	TITULO
hugo	físico
hugo	matemático
maría	médica
luis	abogado
luis	matemático

r2.2:

NOMBRE	IDIOMA
hugo	inglés
hugo	francés
maría	alemán
maría	italiano
luis	portugués
luis	inglés

O sea que para poder descomponer PROFESIONAL en forma SPI se debe cumplir *que ciertas tuplas deben existir obligatoriamente*

Definiremos un nuevo tipo de dependencia que llamaremos *dependencia multivaluada (DMV)*

En nuestro caso diremos que se cumple la DMV: NOMBRE ->-> TITULO

(Decimos que NOMBRE multidetermina TITULO o que TITULO es multideterminado por NOMBRE)

También se verifica: NOMBRE ->-> IDIOMA

En forma general diremos que dado:

	X	Y	Z	(donde Z =R–XY)
t1				
t2				
t3				
t4				
…				

La DMV X->->Y se cumple en R, si para cualquier instancia r de R, se cumple que si existen 2 tuplas t1 y t2 en r para las cuales t1[X] = t2[X], luego deben existir otras 2 tuplas t3 y t4 en r tal que:

t1[X] = t2[X] = t3[X] = t4[X]

t3[Y] = t1[Y] y t3[Z] = t2[Z]

t4[Y] = t2[Y] y t4[Z] = t1[Z]

	X	Y	Z	(Z =R–XY)
t1	x1	y1	z2	
t2	x1	y2	z1	
t3	x1	y1	z1	
t4	x1	y2	z2	

Dadas t1 y t2 la DMV asegura que t3 debe existir
La existencia de t4 se desprende del carácter simétrico de la definición

Con esta definición las DFs son un caso particular de las DMVs.

EJEMPLO 23:

Sea R(A,B,C, D) sujeto a la DMV: A->->B

¿Si sabemos que tenemos las tuplas t1=(a b1 c1 d1), t2=(a b2 c2 d2), t3=(a b3 c3 d1) están en r, qué otras tuplas deben estar en r?

	A	B	C	D	
t1	a	b1	c1	d1	
t2	a	b2	c2	d2	
t3	a	b3	c3	d1	
t4	**a**	**b1**	**c2**	**d2**	**(por t1 y t2)**
t5	**a**	**b2**	**c1**	**d1**	**(por t1 y t2)**
t6	**a**	**b1**	**c3**	**d1**	**(por t1 y t3)**
t7	**a**	**b3**	**c1**	**d1**	**(por t1 y t3)**
t8	**a**	**b3**	**c2**	**d2**	**(por t2 y t3)**
t9	**a**	**b2**	**c3**	**d1**	**(por t2 y t3)**

6.1. REGLAS DE INFERENCIA PARA DFs y DMVs:

3 reglas de Armstrong para DFs ya dadas:

7) Reflexividad: Si Y ⊆ X entonces X-- >Y
8) Aumento: Para cualquier W, si X-->Y entonces XW-->WY
9) Transitividad: Si X-->Y e Y-->Z entonces X-->Z

3 reglas adicionales para FDs ya dadas:

10) Unión: X-->Y y X-->Z entonces X-->YZ
11) Pseudotransitividad: Para cualquier W, X-->Y e YW-->Z entones XW--> Z
12) Descomposición: X-->YZ entonces X-->Y y X-->Z

4 reglas específicas para DMVs:

7. Complementación: Si X->->Y entonces X->->R-XY
8. Reflexividad: Si Y ⊆ X entonces X->->Y
9. Aumento: Para cualquier W, si V⊆ W, X->->Y entonces XW->->YV
10. Transitividad: Si X->->Y e Y->->Z entonces X->->Z-Y

3 reglas adicionales para DMVs:

11. Unión: Si X->->Y y X->->Z entonces X->->YZ
12. Pseudotransitividad: Si X->->Y y YW->->Z entonces XW->->Z-WY
13. Descomposición: Si X->->Y y X->->Z entonces X->->(Y∩Z)
X->->(Y- Z)
X->->(Z-Y)

3 reglas que conectan DFs y DMVs:

14. Conversión: Si X-->Y entonces X->->Y
15. Interacción: Si X->->Y y existe un W tal que W∩Y=∅, W-->Z y Z ⊆Y entonces X-->Z
16. Pseudotransitividad mixta: Si X->->Y y XY-->Z entonces X-->Z-Y

EJEMPLO 24:

Dados:

R(A,B,C,D,E) y
D= {A->->BC, DE->->C}

Derivar AE->->BD

1. A->->BC (dada)
2. A->->DE (complementación de 1)
3. DE->->C (dada)
4. A->->C (transitividad de 2 y 3)
5. AE->->C (aumento de 4 con W=E y V=ϕ)
6. AE->->BD (complemento de 5)

EJEMPLO 25:

Demostrar la siguiente regla:

Si X->->Y entonces X->->(Y-X)

1. X->->Y (dada)
2. X-->X (reflexividad de DFs)
3. X->->X (conversión)
4. X->->(Y-X) (descomposición entre 1 y 3)

EJEMPLO 26:

Sean:
R(A,B,C,D,E) y
D= {A->->BC, DE->->C}

Aplicando las reglas de inferencia vemos que:

D |= {A->->DE, A->->C, AD->->BE}

La siguiente instancia r cumple con todas estas DMVs:

A	B	C	D	E
a	b	c	d	e
a’	b’	c’	d	e
a’	b’	c	d	e
a	b	c’	d	e
a”	b’	c’	d’	e

EJEMPLO 27:

Sean:
R(A,B,C,D,E) y
D= {A->->BC, D-->C}

Aplicando las reglas de inferencia vemos que:

D |= {A-->C}

La siguiente instancia r cumple con estas DFs y DMVs:

A	B	C	D	E
a	b	c’	d	e
a	b’	c’	d’	e’
a	b’	c’	d	e
a	b	c’	d’	e’

6.2. DMVs TRIVIALES:

Dados R y $X,Y \subseteq R$,

X->->Y es trivial si

R= XY o $Y \subseteq X$

7. DESCOMPOSICION BINARIA SPI CON FDs y DMVs:

Dados R, ρ= (R1, R2) y **D** (conjunto de DFs y DMVs)
decimos que ρ es SPI sii

3) La dependencia multivaluada (R1 $\cap$ R2) ->- > (R1 – R2) está en **D+**
o
4) La dependencia multivaluada (R1 $\cap$ R2) ->- > (R2 – R1) está en **D+**

8. CUARTA FORMA NORMAL (4FN):

R está em 4FN com respecto a un conjunto **D** de DFs y DMVs si para toda DMV no trivial de la forma X->->Y, X es una superclave de R.

De la defininción se desprende que:

Si R está en 4FN, también está en FNBC
Si R está en 4FN, las dependencias no triviales que valen son funcionales

8.1. ALGORITMO PARA DESCOMPONER EN 4FN SPI:

Es análogo al que usamos para descomponer en FNBC, o sea:

Si la dependencia no trivial X->->Y viola 4FN entonces hay que descomponer en

XY y R-Y

y así sucesivamente…

Si hay una dependencia multivaluada X ->-> Y que viola 4FN se parte R en R1 y R2 de la siguiente forma:

```
              R
             /  \        X ->- >Y
            /    \
           /      \
     R1= XY      R2= R-Y
```

EJEMPLO 28:

Sean R(A,B,C,D,E)
D= {A-->BC, C->->DE}

Vemos que R no está en 4FN porque la clave es ADE y C->->DE viola la 4FN.

Entonces descomponemos en:

R1= ABC   R2= CDE

Esta descomposición está en 4FN con respecto a **D** aunque **D**|= A->->B, A->->C que son no triviales y valen en R1 pero A es una clave para R1.

```
            ABCDE
             /   \        C->->DE
            /     \
           /       \
     R2= CDE       R1= ABC
```

BIBLIOGRAFÍA:

Gran parte de los ejemplos de este apunte fueron tomados de los siguientes libros:

- "Introducción a las Bases de Datos Relacionales", de Alberto Mendelzon y Juan Ale, 2000.

- "Principles of Database and Knowledge-Base Systems", Volume I, de Jeffrey Ullman, 1988.

- El algoritmo para hallar todas las claves es una sugerencia de Alejandro Vaisman, UBA-FCEyN.