Dependencias Multivaluadas

- Licenciatura e Ingeniería en Informática
- 2do. año

DEPENDENCIAS MULTIVALUADAS

- Sea una instancia *r* de un esquema de relación *RS = (ATR, DMN, dom, M, SC)* . Se dice que $X \rightarrow\rightarrow Y$ (y se lee *"X multidetermina a Y"*) cuando el hecho de que existan en *r* dos tuplas *t1* y *t2* tales que *t1(X)=t2(X)* implica que existen en *r* otras dos tuplas *t3* y *t4* tales que:
- $t1(X) = t2(X) = t3(X) = t4(X)$
- $t3(Y) = t1(Y) \wedge t3(ATR\text{-}X\text{-}Y) = t2(ATR\text{-}X\text{-}Y)$
- $t4(Y) = t2(Y) \wedge t4(ATR\text{-}X\text{-}Y) = t1(ATR\text{-}X\text{-}Y)$

DEPENDENCIAS MULTIVALUADAS

X	Y	ATR-X-Y	
x1	y1	z1	t1
x1	y2	z2	t2
...	...	...	
x1	y2	z1	t4
x1	y1	z2	t3

Intuición primera: Se pueden intercambiar los valores de Y para t1 y t2 y armar dos nuevas tuplas que también deben estar en r.

Intuición segunda: Dado un valor para X hay un *conjunto* de valores de Y asociado a X y no asociado a ATR-X-Y.

DEPENDENCIAS MULTIVALUADAS

Lema Si una instancia *r* de un esquema de relación *RS = (ATR, DMN, dom, M, SC)* satisface ***X->->Y***, entonces *r* satisface la mvd ***X ->-> ATR - XY***

La mvd X ->-> Ø se satisface trivialmente en cualquier instancia de un esquema que contenga a *X* entre sus atributos.

DEPENDENCIAS MULTIVALUADAS

Ejemplo: Sea un esquema

RS = ({A,B,C,D}, DMN, dom, M, SC) y sea *r* una instancia de *RS:*

A	B	C	D
a	*b*	*c*	*d*
a	*b*	*c'*	*d'*
a	*b*	*c*	*d'*
a	*b*	*c'*	*d*
a	*b'*	*c'*	*d*
a'	*b*	*c*	*d'*

Esta instancia satisface *AB->->BC,* entonces también satisface *AB->->D*

Propiedades de las mvds

Relación entre mvd's y la descomposición con join sin pérdida.

Teorema: Sea *r* una instancia del esquema *RS*, con *X, Y, Z* subconjuntos de *ATR* tales que ***Z = ATR - XY.***

La instancia *r* satisface la mvd *X ->-> Y* si se descompone con join sin pérdida en los esquemas de relación

RS1 = ({**X,Y**}, DMN1, dom1, M1, SC1) y

RS2 = ({**X,Z**}, DMN2, dom2, M2, SC2)

Corolario: Sea *r* una instancia de un esquema *RS* y sean

X, Y subconjuntos de *ATR.*

Si *r* satisface *X --> Y,* entonces satisface la mvd *X ->-> Y.*

Teorema

- Sea SC un conjunto de dfs definidas sobre ATR. Sean X, Y, Z subconjuntos de TAR con Z=ATR-XY.
  Si Y **no** está contenida en X+ y Z no está contenida en X+, entonces existe una instancia r del esquema RS que **satisface SC** y que **no satisface la mvd X->->Y**
- Por el teorema podemos ver que las únicas mvds implicadas por un conjunto de dfs son aquellas de la forma X->->Y, donde Y está contenido en X+ o ATR-XY está contenido en X+.
- Esto quiere decir que **no existen mvds** que se satisfagan siempre en r y **que no correspondan directamente a dfs**.

Ejemplo: Sea RS=({A,B,C,D,E,I),DMN,dom,M,SC)
SC={A->BC,C->D}

Entonces SC implica A->->BCD y A->->C

pero SC no implica A->->DE (DE no pertenece a A+)

Reglas de inferencia para mvds

La siguiente lista de reglas de inferencia para mvds es válida y completa.

M1. AUMENTACIÓN MULTIVALUADA:
{X->->Y} |= WX ->-> VY si $V \subseteq W$

M2. TRANSITIVA MULTIVALUADA:
{X->->Y, Y->->Z} |= X ->-> Z-Y

M3. UNIÓN MULTIVALUADA:
{X->->Y, X->->Z} |= X ->-> YZ

M4. INTERSECCIÓN MULTIVALUADA:
{X->->Y, X->->Z} |= X ->-> $Y \cap Z$

Reglas de inferencia para mvds

M5. DIFERENCIA MULTIVALUADA:
{X->->Y, X->->Z} |= X ->-> Y-Z, X ->-> Z-Y

M6. COMPLEMENTACIÓN:
{X->->Y} |= X ->-> ATR -Y

FM1. REPLICACIÓN:
{X->Y} |= X ->-> Y

FM2. COALESCENCIA:
{X->->Y, W->Z} |= X -> Z

Si Z $\subseteq$ Y, y existe W tal que W $\cap$ Y = Ø

FM3. CUASITRANSITIVA:
{X->->Y, Y->Z} |= X ->-> Z-Y

Ejemplo

Sea RS = ({A,B,C,G,H,I}, DMN, dom, M, SC) donde
SC = {A ->-> B, B ->-> HI, CG ->-> H }.

Se pueden deducir:

A ->-> CGHI (aplicando M6 a A ->-> B)
A ->-> HI (M2 a A ->-> B y B ->-> HI)
B-->H (FM2 a B ->-> HI y CG-->H y CG $\cap$ HI = Ø)
A ->-> CG (M5 a A ->-> (CGHI - HI)).

Algoritmo: Cálculo de B(M,X), base de dependencias de X respecto a M

ENTRADA: un esquema RS, M conjunto de mvds obtenido de SC, $X \subseteq ATR$.

SALIDA : B(M,X)

```
BMX := ATR – X
MIENTRAS (BMX cambió) HACER
    PARA CADA (V ->->W ∈ M) y (Y ∈ BMX) y (Y ∩ W != Ø) y (Y ∩ V = Ø)
    HACER
        BMX := (BMX - Y) U (Y ∩ W) U (Y - W)
    FINPARA
FINMIENTRAS
RETURN (BMX)
```

Ejemplo

Se calculará B(M,{A B}) para un esquema primitivo de relacion PRS con:

ATR = {A B C D E F G}

SC = {AB ->-> CD, C ->-> F, C-->E}

El algoritmo arranca con

M = {AB ->-> CD, C ->-> F, C ->-> E}

El conjunto BMX de conjuntos disjuntos de atributos es: BMX = {C D E F G}. En pasos sucesivos, este conjunto BMX se transforma en: {CD EFG}, luego en {CD F EG} y finalmente en {CD F E G}

Cálculo de la clausura de un conjunto de mvds

El cálculo de SC*, cuando contiene fds y mvds, presenta las mismas dificultades (o peores) que el cálculo de la clausura de un conjunto de fds.

En la práctica nos vamos a limitar a testear si una mvd está en el conjunto clausura, sin calcularlo.

Verificación de si X ->-> Y $\in$ SC*:

1. A partir de SC, construir un conjunto M que contiene a todas las mvds de SC, y por cada fd Z-->(A1 A2 ... Ak) $\in$ SC, sustituirla por Z->->Ai, i=1..k.

2. Calcular la base de dependencias de X respecto al conjunto M, B(M,X).

3. Ver si Y - X = C1 U ... U Cn,   Ci $\in$ B(M,X)

Joins sin pérdida y preservación de mvds
Teorema:

Sea RS = (ATR, DMN, dom, M, SC) un esquema de relación, donde SC es un conjunto de fds y mvds.

Sea una descomposición de RS en dos subesquemas

RS1 = (ATR1, DMN1, dom1, M1, SC1) y

RS2 = (ATR2, DMN2, dom2, M2, SC2)

Entonces está descomposición es de join sin pérdida si y sólo se cumple al menos una de las siguientes implicaciones:

1. $ATR1 \cap ATR2 \twoheadrightarrow ATR1$
2. $ATR1 \cap ATR2 \twoheadrightarrow ATR2$

Definición: Descomposición preservadora de mvds

La descomposición {RS1, RS2, ..., RSn} es una descomposición preservadora de dependencias si para cada conjunto de instancias r1, r2, ..., rn respectivamente de los esquemas RS1, RS2, ..., RSn, tales que cada ri satisface a SC'i,

existe una instancia r de RS que satisface SC y para la cual

ri = π(r, ATRi), para i=1..n

Dependencias join

Es útil poder limitar el conjunto de instancias *legales* de un esquema RS a aquellas para las cuales una descomposición dada sea de join sin pérdida.

Definición: **Dependencia join:**

Sea ATR = {A1, ..., An} y

X1,...,Xm subconjuntos de {A1,...,An}.

Se dice que existe una dependencia join ***(X1,...,Xm)**

Si y sólo si la descomposición del esquema en m subesquemas con conjuntos de atributos X1,...,Xm respectivamente **es de join sin pérdida**.

O sea que para toda instancia r del esquema se cumple que

$r = \pi(r,X1) * \ldots \pi(r,Xm)$

Observaciones:

Las **mvds** son casos particulares de las **jds**, ya que toda instancia r de RS = ({X,Y,Z}, DMN, dom, M, SC) que satisfaga la X ->-> Y satisface la jd *(XY,XZ).

La forma práctica de deducir una jd a partir de un conjunto SC se hace con el mismo método utilizado para testear join sin pérdida.

En el caso de las fds y de las mvds, fue posible dar un sistema de reglas de inferencia, no para las dependencias join

Toda jd de la forma *(ATR1,ATR2) es equivalente a una mvd. Sin embargo existen otras jds que no son equivalentes a una mvd

NORMALIZACION SEGUN MVDS Y JDS

Forma normal 4NF

Un esquema de relación RS = (ATR, DMN, dom, M, SC) está en cuarta forma normal (4NF) si
Para cada mvd no trivial **X->->Y** con SC |= X ->-> Y,
X es una superclave en RS

En otras palabras, decimos que RS está en 4NF si para todas las mvds X->->Y $\in$ SC*, donde Y es no vacío y XY no incluye a todos los atributos de RS, se cumple que X es superclave de RS

Un esquema de base de datos está en 4NF si cada uno de sus esquemas de relación está en 4NF.

Observaciones:

La definición de 4NF sólo difiere de la de BCNF en que se utilizan mvds en vez de fds.

Todo esquema en 4NF está en BCNF

Si un esquema no está en BCNF, entonces existe una fd no trivial

X-->Y, donde X no es una clave.

Puesto que X-->Y implica que X->->Y (por la regla de replicación), entonces el esquema no puede estar en 4NF.

La analogía entre 4NF y BCNF se aplica también al algoritmo para descomponer un esquema a 4NF. Este algoritmo es idéntico al de descomposición a BCNF, usando mvds en vez de fds.

Por la naturaleza de las mvds y de los joins sin pérdida, el algoritmo genera solamente descomposiciones de join sin pérdida.

Obtención de π *mvd(SC,ATR)*

Computar SC*

Para cada $X \rightarrow Y$ en SC*, si $X \subseteq S$, agregar $X \rightarrow (Y \cap S)$ a $\pi mvd(SC,ATR)$

Para cada $X \rightarrow\rightarrow Y$ en SC*, si $X \subseteq S$, agregar $X \rightarrow\rightarrow (Y \cap S)$ a $\pi mvd(SC,ATR)$

Algoritmo de descomposición en 4NF con join sin pérdida

Descomp := (ATR)

Mientras Descomp cambie

  Para cada Ri de Descomp

  Buscar en π mvd(SC,ATRi) alguna $X \rightarrow\rightarrow Y$ violación de 4NF en ATRi

  Si existe, reemplazar **ATRi por XY y Ri-Y**

Devolver Descomp

Definición: Quinta forma normal

5NF, PJNF (forma normal de proyección dependencia)

Sea RS = (ATR, DMN, dom, M, SC) un esquema de relación donde SC es un conjunto que contiene jds.

Decimos que RS está en quinta forma normal (5NF) con respecto a SC si **para toda jd *(RS1, ..., RSp)** implicada por SC que se aplica en RS,

o bien la jd es trivial

o todos los ATRi de cada RSi es superclave de RS

Un esquema de base de datos está en 5NF si cada uno de sus esquemas de relación está en 5NF

Observación:

Puesto que toda mvd es también una jd, es fácil comprobar que cualquier esquema en PJNF está también en 4NF

Puede no ser posible encontrar una descomposición a PJNF que preserve las dependencias.

FORMAS NORMALES GENERALES

Hasta ahora la normalización se ha enfocado a la definición de algún tipo de restricción (fd, mvd o jd), aplicada a un esquema RS = (ATR, DMN, dom, M, SC), para después emplear dicha restricción para definir una forma normal.

La **Forma normal de dominio-clave** está basada en tres conceptos:

1. **Declaración de dominio.**

Sea A ∈ ATR y sea D ∈ DMN. Sea (A,D) ∈ dom. La declaración de dominio requiere que para toda tupla t, t[A] ∈ D

2. **Declaración de clave.**

Sea K ∈ ATR. La declaración de clave requiere que K sea una superclave del esquema, es decir que K-->ATR. Nótese que todas las declaraciones de clave son fds, pero no viceversa.

3. **Restricción general.**

Una restricción general es un predicado en el conjunto de todas las instancias de un esquema dado, y generalmente se expresa en una forma previamente establecida, por ejemplo, lógica de primer orden. Las dependencias que se han estudiado son ejemplos de restricciones generales.

Ejemplo

En una base de datos bancaria, sea un esquema RS con un conjunto de atributos

ATR = {SUCURSAL, NROCUENTA, SALDO}

Supóngase que todas las cuentas corrientes cuyo número comienza con "9" son cuentas especiales de alta tasa de interés, con un saldo mínimo de $2.500

En este caso se incluiría como restricción general:

"si el primer dígito de t[NROCUENTA] es '9', entonces t[SALDO] $\geq$ 2500"

Observación

Las declaraciones de dominio y las de clave pueden probarse fácilmente en un DBMS real.

Sin embargo, es posible que sea muy costoso (en tiempo y espacio) probar una restricción general.

El objetivo de un diseño de base de datos en forma normal de dominio-clave es permitir la prueba de restricciones generales empleando solamente las de dominio y clave.

Definición:
Forma normal de dominio-clave, DKNF (Domain-Key Normal Form)

Sea:

D un conjunto de restricciones de dominio y

K uno de restricciones de clave para un esquema RS.

Las restricciones generales para RS se denotarán por medio de **G**.

El esquema RS está en la forma normal de dominio-clave (DKNF), si **D U K implica lógicamente a G**

Ejemplo: Descomposición a DKNF

Volviendo a la restricción general de las cuentas corrientes, no está en DKNF.

Para crear un diseño DKNF, se requieren dos subesquemas:

RS1 = ({SUCURSAL, CUENTA, SALDO}, DMN1, dom1, M1, SC1)

RS2 = ({SUCURSAL, CUENTA, SALDO}, DMN2, dom2, M2, SC2)

Se conservan todas las dependencias que se tenían en RS como restricciones generales.

Las limitantes de dominio de **RS1** exigen que el número de cuenta no comience con 9.

Las restricciones de dominio de **RS2** exigen que el número de cuenta comience con 9 y el saldo sea mayor de 2500.

Observación

Comparación de DKNF con las demás formas normales.

En las otras formas normales se supuso (implícitamente) que el dominio de cada atributo era el apropiado.

Para cada forma normal, se permitió una forma restringida de restricción general (un conjunto de fds, mvds o de jds).

Es posible volver a escribir las definiciones de PJNF, 4NF, BCNF y 3NF de manera tal que se vean como **casos especiales de DKNF**.

PJNF como caso especial de DKNF

Sea *RS = ({A1,A2,...,An}, DMN, dom, M, G)* un esquema de relación.

Sea *dom(Ai)* el dominio del atributo *Ai*.

Todas las restricciones de dominio **D** son entonces de la forma *Ai* $\subseteq$ *dom(Ai).* Sean las restricciones generales un conjunto **G** de fds, mvds o jds. Si F es el conjunto de fds en G*,* sea **K** el conjunto de restricciones de clave de todas las fds no triviales en F* de la forma *X-->ATR.*

El esquema RS está en **PJNF** si y sólo si está en **DKNF con respecto a D, K, G.**

Observación

Una consecuencia de DKNF es que se eliminan todas las anomalías de inserción y borrado.

La DKNF representa una forma normal "extrema", porque permite restricciones arbitrarias, no sólo dependencias y permite probar estas restricciones en forma eficiente.

Si un esquema no está en DKNF, puede pasarse a DKNF por medio de una descomposición, tales descomposiciones no siempre preservan las dependencias.

ESQUEMAS NFNF

Un esquema está en 1NF si los dominios de y todos los atributos son atómicos.

Un dominio es atómico si se considera a los elementos del dominio como unidades indivisibles.

La cuestión importante en la 1NF no es el dominio en sí, sino la forma como se emplean los elementos del dominio en la base de datos.

Por ejemplo, el dominio de todos los enteros no sería atómico si se considera a cada número entero como una lista ordenada de dígitos.

Si cuestionamos la legitimidad de suponer que se cumple la 1NF, podemos concebir esquemas que se denominan **no 1NF (Non-First Normal Form, NFNF)**

Observación

Para un usuario final de un sistema de base de datos la suposición que se cumple la 1NF no es inmediata.

La siguiente instancia de un sistema de recuperación de documentos no cumple 1NF, pero es una forma "intuitiva" de representar la información

TITULO	AUTORES	FECHA	PCLAVE
Plan de ventas	*{Smith,Jones}*	*1 abr 79*	*{utilidades, estrategia}*
Reporte de situación	*{Jones, Frick}*	*17 jun 85*	*{utilidades, personal}*