Dependencias Funcionales

Bibliografía: Fundamentos de bases de datos
Korth , Silberschatz

Conceptos básicos

- Las DF son un tipo particular de **restricción**.
- Permiten expresar hechos acerca de la realidad que se está modelando con la BD.

Se definió **superclave…**

- Sea R un esquema de relaciones. Un subconjunto K de R es una superclave de R si, en cualquier relación legal r(R),
- $\forall$ **t1,t2** $\in$ **r** / **t1≠t2** $\rightarrow$ **t1[K]≠t2[K].**
  - Es decir, dos tuplas en cualquier relación legal r(R) no pueden tener el mismo valor en el conjunto de atributos K.

Dependencia Funcional (DF)

- La noción de DF generaliza la noción de superclave:
- Sea $\alpha \subseteq R$ y $\beta \subseteq R$. La DF $\alpha \rightarrow \beta$
- se cumple en R si en cualquier relación legal r(R),
- $\forall$ **t1 , t2** $\in$ **r** / **t1[α] = t2 [α]** también se cumple que **t1[β] = t2 [β]**

Dependencia Funcional (DF)

- Utilizando la notación de la DF, decimos que K es una superclave de R si **K → R.**
- Es decir, **K** es una **superclave** si siempre que **t1[K]=t2[K],** también se cumple que **t1[R]=t2[R]** (es decir, **t1 = t2**)
- Las DF permiten expresar **restricciones** que **no pueden expresarse** con **superclaves**.

Ejemplo

- Consideremos la relación r y veamos qué DF se satisfacen.

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

A → C   se satisface.

- Las dos tuplas con valor a1 en A tienen el mismo valor en C, c1.
- Las dos tuplas con valor a2 en A tienen el mismo valor en C, c2.
- No existen otros pares de tuplas distintos que tengan el mismo valor en A.

C → A   no se satisface.

- Sean   t1=(a2, b3, c2, d3) y t2=(a3 ,b3, c2, d4)
- tienen el mismo valor en C, c2 y
- distintos valores en A, a2 y a3, respectivamente.

➔ hemos encontrado un par de tuplas t1 y t2 tales que tl [C] = t2 [C] pero t1 [A] ≠ t2 [A].

- r satisface muchas otras DF.
- Por ejemplo:
- AB → D
- A → A y
- las demás DF triviales

(**una DF de la forma $\alpha \rightarrow \beta$ es trivial si** $\beta \subseteq \alpha$)

- Llamaremos **F** al conjunto de DF
- Usaremos las DF de dos formas:
- Para especificar restricciones en el conjunto de relaciones legales. (**F se cumple en R**). Es decir: una dependencia que se cumple en un esquema.
- Para probar si una relación es legal bajo un conjunto dado de DF . (**r satisface a F**). Es decir: una relación que satisface una dependencia.

Ejemplo

- Sean los esquemas de relación del ejemplo **bancario**:
- En una **instancia** de la relación **cliente**, se satisface **calle → ciudad-cliente**. Pero, en el mundo real dos ciudades pueden tener calles con el mismo nombre. Por tanto, **no incluiríamos** calle → ciudad-cliente **en el conjunto de DF** que se deben cumplir.

Ejemplo

- En la relación **préstamo** se satisface **número-préstamo → cantidad**. En el mundo real exigimos que cada préstamo tenga una única cantidad. Por tanto, **imponemos la restricción** de que se cumpla número-préstamo → cantidad en esquema-préstamo.

Ejemplo

- En la relación **sucursal** se satisface nombre-sucursal → activo y activo → nombre-sucursal. Exigimos que **nombre-sucursal → activo** se cumpla en esquema-préstamo y no queremos exigir que se cumpla **activo → nombre-sucursal**.

Al diseñar una BDR se listan las DF que se deben cumplir siempre.

En el **ejemplo** bancario:

En Esquema-**sucursal**:

nombre-sucursal → ciudad-sucursal

nombre-sucursal → activo

En Esquema-**cliente**:

nombre-cliente → ciudad-cliente

nombre-cliente → calle

En Esquema-**préstamo**:

número-préstamo → cantidad

número-préstamo → nombre-sucursal

En Esquema-**depósito**:

número-cuenta → saldo

número-cuenta → nombre-sucursal

Cierre de un conjunto de DF

- Dado un conjunto F de DF , podemos probar que se cumplen otras ciertas DF .
- Se dice que **F implica lógicamente dichas DF** .

Ejemplo: Sean: el esquema de relaciones R=(A,B,C,G,H,I) y el conjunto de DF:

A → B

A → C

CG → H

CG → I

B → H

La DF **A → H** **se implica lógicamente**.

<u>Demostración:</u>

- Sean t1 , t2 tuplas tales que t1 [A] = t2 [A]
- Como A → B, se deduce de la definición de DF que t1[B]= t2[B]
- Además, como B → H, se deduce de la definición que t1 [H] = t2 [H]
- Por tanto, siempre que t1 y t2 son tuplas tales que t1[A]=t2[A], también se cumple que t1[H] = t2[H] es decir A → H.

Definición

- Sea F un conjunto de DF.
- El **cierre de F (F+)** es el conjunto de DF que F implica lógicamente.
- Dado F, podemos calcular F+ directamente de la definición formal de DF

Axiomas de Armstrong

- Una **técnica** para calcular F+ se basa en tres axiomas o reglas de inferencia para DF. (Armstrong, 1974)
- Aplicando estas reglas repetidamente, podemos encontrar **F+ completo** dado F.

Axiomas de Armstrong

- α, β, γ ... representan conjuntos de atributos;
- A, B, C ... representan atributos individuales;
- $\alpha\ \beta$ representa $\alpha \cup \beta$.

Regla de reflexividad:
- Si α es un conjunto de atributos y $\beta \subseteq \alpha$, entonces se cumple $\alpha \rightarrow \beta$

Regla de aumento:
- Si se cumple $\alpha \rightarrow \beta$ y γ es un conjunto de atributos, entonces se cumple $\gamma\ \alpha \rightarrow \gamma\ \beta$

Regla de transitividad:
- Si se cumple $\alpha \rightarrow \beta$, y se cumple $\beta \rightarrow \gamma$ entonces se cumple $\alpha \rightarrow \gamma$

Axiomas de Armstrong

- Estas reglas son **seguras** porque no generan DF incorrectas. (Correctas: cualquier DF inferida se cumple en R)
- Las reglas son **completas** porque para un conjunto dado F de DF , nos permiten generar F+ completo. (se obtienen todas las DF posibles).

Reglas adicionales derivadas de los axiomas de Armstrong

Regla de unión:

- Si $\alpha \rightarrow \beta$ y $\alpha \rightarrow \gamma$, entonces se cumple $\alpha \rightarrow \beta\gamma$

Regla de descomposición:

- Si $\alpha \rightarrow \beta\gamma$, entonces se cumplen $\alpha \rightarrow \beta$ **y** $\alpha \rightarrow \gamma$

Regla de pseudotransitividad:

- Si $\alpha \rightarrow \beta$ y $\gamma\beta \rightarrow \delta$ entonces se cumple $\alpha\gamma \rightarrow \delta$

Ejemplo

Sea R = (A, B, C, G, H, I) y
F = {A → B, A → C, CG → H, CG → I, B → H}.

Algunos miembros de F+, serán:

A → H

- Como A → B y B → H, aplicamos la regla de **transitividad**.

Es más fácil usar los Axiomas de Armstrong para demostrar A → H de lo que fue deducir directamente de las definiciones como hicimos anteriormente.

Ejemplo

Sea R = (A, B, C, G, H, I) y
F = {A → B, A → C, CG → H, CG → I, B → H}.

CG → HI

- Como CG → H y CG → I,
  la regla de **unión** implica que CG → HI.

Ejemplo

Sea R = (A, B, C, G, H, I) y
F = {A → B, A → C, CG → H, CG → I, B → H}.

AG → I

Necesitamos varios pasos para demostrar AG → I.

- Primero, observar que se cumple A → C.
- Usando la regla de aumento, vemos que AG → CG.
- Además, como tenemos que CG → I,
  así por la regla de transitividad se cumple AG → I.

Cierre de conjuntos de atributos

Definición:

Sea α **un conjunto de atributos**.

Al conjunto de todos los atributos determinados funcionalmente por α bajo un conjunto F de DF

se le llama **cierre de α bajo F** (α**+**).

Cierre de conjuntos de atributos

- α es una **superclave** si α**+ = R**.

α+ = **conjunto de atributos determinados funcionalmente** por α.

Algoritmo para calcular α+

- Entrada: un conjunto F de DF y el conjunto α de atributos.
- Salida: se almacena en la variable resultado.

```
resultado : = α ;
while (cambios en resultado) do
  for each DF  β → γ in F do
        begin
          if β ⊆ resultado then resultado:= resultado U γ ;
        end
```

Funcionamiento del algoritmo

- Empezamos con **resultado = AG**.
- La **primera** vez que ejecutamos el bucle **while** para probar cada DF encontramos que
  - **A → B** nos hace incluir B en resultado.
  - (A → B está en F, A ⊆ resultado (que es AG), por tanto resultado: = resultado U B).
  - **A → C** hace que resultado se convierta en **ABCG**.
  - **CG → H** hace que resultado se convierta en **ABCGH**.
  - **CG → I** hace que resultado se convierta en **ABCGHI**.

Funcionamiento del algoritmo

- La **segunda** vez que ejecutamos el bucle **while:**
  - no se añaden atributos nuevos a resultado y
  - el algoritmo termina.

- **Este algoritmo es correcto y encuentra α+ completo.**
  - se demuestra aplicando los AA y las reglas asociadas con ellos.

Definición:

- Dos conjuntos de DF son equivalentes si sus clausuras son iguales.

- E y F son equivalentes $\Leftrightarrow$ E+ = F+

Recubrimiento minimal (canónico)

- Un **recubrimiento canónico Fc** es un restricción de un conjunto dado de DF que:

- **minimiza** el número de DF a ser probadas en el caso de una actualización.

- el conjunto de DF **resultantes** da el conjunto de **tablas normalizadas** (1 tabla x cada DF).

Recubrimiento minimal (canónico)

Recubrimiento canónico de F es un conjunto de DF tal que:

- F implica lógicamente a todas las dependencias en Fc, y
- Fc implica lógicamente a todas las dependencias en F.

Además Fc debe cumplir las **propiedades**:

- Cada DF $\alpha \rightarrow \beta$ en Fc no contiene atributos extraños a α.
  - Los **atributos extraños** son atributos que pueden eliminarse de α sin cambiar Fc+.
  - **A** es extraño a α si
    - A $\in$ α y
    - Fc implica lógicamente a (Fc - { $\alpha \rightarrow \beta$ } U { α - A $\rightarrow \beta$ }).

Recubrimiento minimal (canónico)

- Cada DF $\alpha \rightarrow \beta$ en Fc no contiene atributos **extraños** a β.
  - Los **atributos extraños** son atributos que pueden eliminarse de β sin cambiar Fc+.
  - **A** es extraño a β si
    - A $\in$ β y
    - (Fc - { $\alpha \rightarrow \beta$ } U { $\alpha \rightarrow \beta$ - A }) implica lógicamente a Fc.
- Cada lado izquierdo de una DF en Fc es único.
- Es decir, no existen dos DF $\alpha1 \rightarrow \beta1$ y $\alpha2 \rightarrow \beta2$ en Fc
  tales que $\alpha1 = \alpha2$.

Otra definición

- Toda DF de F tiene un solo atributo en el lado derecho.
- No podemos quitar ninguna y seguir teniendo un conjunto equivalente.
- No se puede reemplazar ninguna X $\rightarrow$ A por Y $\rightarrow$ A con Y $\subset$ X y Y $\neq$ X y seguir siendo equivalente.

Cálculo del recubrimiento canónico

- Utilizar la **regla** de **unión** para sustituir cualquier dependencia en F de la forma
  $\alpha1 \rightarrow \beta1$ y $\alpha1 \rightarrow \beta2$ con $\alpha1 \rightarrow \beta1\,\beta2$.
- Probar cada DF $\alpha \rightarrow \beta$ para ver si hay un atributo extraño en α.
- Probar cada DF $\alpha \rightarrow \beta$ para ver si hay un atributo extraño en β.
- Repetir este proceso hasta que no ocurra ningún cambio en el bucle.

Ejemplo: Cálculo del recubrimiento canónico

Sea el esquema (A, B, C) con el conjunto de DF:

A $\rightarrow$ BC
B $\rightarrow$ C
A $\rightarrow$ B
AB $\rightarrow$ C

Ejemplo: Cálculo del recubrimiento canónico

Sea el esquema (A, B, C) con el conjunto de DF:

$A \rightarrow BC$
$B \rightarrow C$
$A \rightarrow B$
$AB \rightarrow C$

- Hay dos DF con el mismo conjunto de atributos en el lado izquierdo:

  $A \rightarrow BC$ y $A \rightarrow B$.

  - Por regla de unión se combinan en: $A \rightarrow BC$.
  - A es extraño en $AB \rightarrow C$ porque $B \rightarrow C$ implica lógicamente a $AB \rightarrow C$ y así

    $((F - \{AB \rightarrow C\}) \cup \{B \rightarrow C\})$ implica lógicamente a Fc.

- La supresión de A de $AB \rightarrow C$ da $B \rightarrow C$ que está en el conjunto de DF.
- Así, el conjunto de DF es: $A \rightarrow BC$

  $B \rightarrow C$

- y se cumplen las propiedades de un recubrimiento canónico.

Diseño de BDR

- El **objetivo** del diseño de una BDR:
- es generar un **conjunto de esquemas de relaciones** que permitan almacenar información **sin redundancia** innecesaria,
- pero que a la vez nos permita **recuperar información fácilmente**.
- Una técnica consiste en diseñar esquemas que tengan una forma normal adecuada.
- Se definirán formas normales usando las DF.
- Los **defectos** que puede tener una BD mal diseñada son:
- repetición de información
- incapacidad para representar cierta información
- pérdida de información
- Se verán en detalle en el tema normalización
- Se solucionan descomponiendo el esquema de relación con problemas en varios esquemas de relaciones.

Descomposición sin pérdida

- Una relación con muchos atributos mal diseñada se puede descomponer en dos ó más esquemas con menos atributos.
- Si esta descomposición no se hace bien puede llegarse a otra forma de diseño defectuoso.
- **Ejemplo:**
- Si el esquema préstamo:
- **esquema-préstamo = (nombre-sucursal, número-préstamo, nombre-cliente, cantidad)**
- se descompone en dos esquemas:
- **esquema-cant = (cantidad, nombre-cliente)**
- **esquema-prest = (nombre-sucursal, número-préstamo, cantidad)**
- se obtienen las siguientes relaciones:

nombre-sucursal	número-préstamo	cantidad
Downtown	17	1000
Mianus	93	500
Perryridge	15	1500
Round Hill	11	900
Perryridge	25	2500
Redwood	23	2000
Brighton	10	2200
Downtown	14	1500
Pownal	29	1200

cantidad	nombre-cliente
1000	Jones
500	Curry
1500	Hayes
900	Turner
2500	Glenn
2000	Smith
2200	Brooks
1500	Jackson
1200	Williams

- Si para alguna consulta se necesita reconstruir préstamo a partir de esta descomposición, se puede obtener mediante: **cant |x| prest**
- La relación resultante es:

nombre-sucursal	número-préstamo	nombre-cliente	cantidad
Downtown	17	Jones	1000
Mianus	93	Curry	500
Perryridge	15	Hayes	1500
Round Hill	11	Turner	900
Perryridge	25	Glenn	2500
Redwood	23	Smith	2000
Brighton	10	Brooks	2200
Downtown	14	Jackson	1500
Pownal	29	Williams	1200
North Town	16	Adams	1300
Downtown	18	Johnson	2000
Downtown	14	Hayes	1500
Perryridge	15	Jackson	1500
Redwood	23	Johnson	2000
Downtown	18	Smith	2000

- La relación resultante contiene tuplas adicionales (en rojo) respecto a préstamo.
- Así las consultas que se efectúen podrían producir resultados erróneos.
- Aunque se tienen más tuplas, se pierde información.
- Este tipo de descomposición se denomina **descomposición con pérdida** y es un mal diseño.
- Es esencial que al descomponer una relación en varias relaciones más pequeñas que la **descomposición sea sin pérdida**.

- **Criterio para determinar si una descomposición tiene pérdida:**
- Sea R un esquema de relaciones y F un conjunto de DF en R.
- R1 y R2 forman una descomposición de R.
- Esta es sin pérdida si por lo menos una de las siguientes DF está en F+:
- R1 intersección R2 → R1
- R1 intersección R2 → R2
- R1 intersección R2 es clave primaria ó candidata de alguna de las dos.

Conservación de las dependencias

- Cuando se actualiza la BD el sistema debe poder comprobar que la actualización no creará una relación ilegal, es decir una que no satisfaga todas las DF dadas.
- **Definición**
- Sea F un conjunto de DF en R y sea R1, R2, .., Rn una descomposición de R.
- La restricción de F a Ri es el conjunto **Fi** de todas las DF en F+ que incluyen únicamente atributos de Ri.
- **Definición**
- Sea F' = F1 U ... U Fn. F' es un conjunto de DF en R. En general, F' <> F, pero puede ser F'+ = F+.
- Decimos que una descomposición que tiene la propiedad **F'+ = F+** es una descomposición que conserva las dependencias.