DISEÑO Y OPTIMIZACIÓN. ALGORITMOS

Bases de datos (GIITIN)
Grado en Ingeniería Informática en Tecnologías de la Información

Elena Montañés Roces

Claves candidatas: reglas

- En ninguna clave candidata estarán…
  - ... los atributos que siempre aparecen en los consecuentes de las dependencias funcionales.
- En todas las claves candidatas estarán…
  - ... todos los atributos que no aparecen en ninguna dependencia funcional
  - … todos los atributos que siempre aparecen en el antecedente de las dependencias funcionales
- El resto de atributos estarán o no en las claves candidatas dependiendo si los atributos del punto anterior son suficientes o no

Claves candidatas: ejemplo

- R=(titulación, escuela, alumno, asig, créditos) y F={titulación→escuela; asig→créditos, titulación}
  - El atributo alumno no aparece en ninguna dependencia, luego estará en todas las claves candidatas
  - El atributo asig solo aparece en los antecedentes de las dependencias, luego estará en todas las claves candidatas.
  - Los atributos créditos y escuela solo aparecen en los consecuentes de las dependencias, luego no estarán en ninguna clave candidata.
  - El atributo titulación no hace falta en ninguna clave candidata porque alumno y asig ya forman una clave candidata.

Cierre de un conjunto de DF (I)

- Alternativa 1:
  - Mediante un algoritmo que utiliza las reglas de Armstrong….
    - …pero es de orden exponencial con respecto al número de atributos, luego debe evitarse utilizarlo.
- Alternativa 2:
  - Mediante el algoritmo para calcular el cierre de un conjunto de atributos…
    - ...no aplicable por necesitar las obtener las partes de un conjunto

Cierre de un conjunto de DF (II)

- Axiomas de Armstrong
  - Reflexividad: Si $\beta \subseteq \alpha$ entonces $\alpha \rightarrow \beta$
    - Como {nmat} $\subseteq$ {nmat, nombre} entonces nmat, nombre $\rightarrow$ nmat
  - Aumentatividad: Si $\alpha \rightarrow \beta$ y $\delta \subseteq \mu$ entonces $\alpha\mu \rightarrow \beta\delta$
    - Como {nombre} $\subseteq$ {nombre, ciudad}, si escuela$\rightarrow$créditos, entonces escuela, nombre, ciudad$\rightarrow$créditos, nombre
  - Transitividad: Si $\alpha \rightarrow \beta$ y $\beta \rightarrow \delta$ entonces $\alpha \rightarrow \delta$
    - Si nombre$\rightarrow$créditos y créditos$\rightarrow$escuela, entonces nombre $\rightarrow$ escuela

Cierre de un conjunto de DF (III)

- Reglas derivadas
  - Unión: Si $\alpha \rightarrow \beta$ y $\alpha \rightarrow \delta$ entonces $\alpha \rightarrow \beta\delta$
    - Si nombre →créditos y nombre →escuela, entonces nombre →créditos, escuela
  - Descomposición: Si $\alpha \rightarrow \beta\delta$ entonces $\alpha \rightarrow \beta$ y $\alpha \rightarrow \delta$
    - Si nombre→créditos, escuela entonces nombre →créditos y nombre →escuela
  - Pseudotransitividad: Si $\alpha \rightarrow \beta$ y $\delta\beta \rightarrow \mu$ entonces $\alpha\delta \rightarrow \mu$
    - Si nombre→créditos y ciudad, créditos→escuela, entonces nombre, ciudad→escuela

Cierre de un conjunto de atributos

- El Cierre de un Cjto. de Atributos $\alpha \subseteq R$ bajo el conjunto de DF F es el conjunto de atributos determinados funcionalmente por α a partir de las dependencias de F
- Algoritmo

```
CjtoAtrib CierreAtributos (CjtoAtrib α, CjtoDF F){
    CjtoAtrib C=α;
    Repetir
        Para cada β→δ ∈ F hacer
            Si β⊆ C entonces C=C∪δ
        fin para
    hasta C no varíe
}
```

Aplicaciones (I)

- Comprobar si una dependencia está en F^+ (sin obtener F^+)

  $$\alpha \rightarrow \beta \in F^+ \leftrightarrow \beta \subseteq \alpha^+ \text{ bajo } F$$

- En realidad, no se va a necesitar calcular F^+ si no saber si una dependencia está en F^+

Aplicaciones (II)

- En particular, averiguar si un conjunto de atributos α es superclave y/o clave candidata de una relación R
  - α es superclave de R si $\alpha \rightarrow R \in F^+$ ($\leftrightarrow R \subseteq \alpha$+bajo F)
  - α es clave candidata de R si $\alpha \rightarrow R \in F^+$ ($\leftrightarrow R \subseteq \alpha$+ bajo F) y $\alpha \rightarrow R$ plena (o completa).
    - Una dependencia es plena (o completa) si el consecuente no depende de un subconjunto del antecedente.
    - $\nexists\, \delta \subseteq \alpha$ t.q. $\delta \rightarrow \beta$
    - Si F={nombre, creditos → escuela ; nombre → escuela} entonces nombre, creditos → escuela NO es plena (o completa).
    - Si F={nombre, creditos → escuela ; escuela → créditos} entonces nombre, creditos → escuela SÍ es plena (o completa).

Recubrimiento canónico (I)

- Fc es un recubrimiento canónico o irredundante de F si:
  - Todas sus dependencias no triviales y plenas (o completas)
  - No contiene atributos extraños (o raros)
    - A es extraño o raro en $\alpha \rightarrow \beta$ si $A \in \alpha$ y $(\alpha\text{-}A) \rightarrow \beta \in F^+$ ($\leftrightarrow \beta \subseteq (\alpha\text{-}A)^+$ bajo F)
  - No contiene dependencias redundantes
  - $\alpha \rightarrow B \in \{F\text{-}\{\alpha \rightarrow B\}\}^+$ ($\leftrightarrow B \subseteq \alpha^+$ bajo $F\text{-}\{\alpha \rightarrow B\}$)

Recubrimiento canónico (II)

- Dado un F, es posible encontrar distintos F_c dependiendo del orden de…
  - …eliminación de los atributos extraños
  - …eliminación de dependencias redundantes
- Todos los F_c de F:
  - …son equivalentes a F
  - …son equivalentes entre sí.

Descomposición sin pérdida (n=2)

- Sea R un esquema de relación que cumple F y $R_1, \ldots, R_n$ una descomposición de R
  - Si n=2 (1° Principio de Risannen). La descomposición es sin pérdida si:
    - Bien $R_1 \cap R_2 \rightarrow R_1 \in F^+$ ($\leftrightarrow$ $R_1 \subseteq (R_1 \cap R_2)+$ bajo F $\leftrightarrow$ $R_1 \cap R_2$ es superclave de R_1)
    - O bien $R_1 \cap R_2 \rightarrow R_2 \in F^+$ ($\leftrightarrow$ $R_2 \subseteq (R_1 \cap R_2)+$ bajo F $\leftrightarrow$ $R_1 \cap R_2$ es superclave de R_2)

Ejemplo: Descomp. sin pérdida (n=2)

- R=(titulación, escuela, alumno, asig, créditos) y F={titulación→escuela; asig→créditos, titulación}
  - R_1=(titulación, escuela)
  - R_2=(alumno, asig, créditos, titulación)

$(R_1 \cap R_2)^+ = \{\text{titulación}\}^+ = \{\text{titulación, escuela}\} = R_1$

⇓

sin pérdida

Descomposición sin pérdida (n>2)

- Sea R un esquema de relación que cumple F y $R_1, \ldots, R_n$ una descomposición de R
  - Si n>2 (Algoritmo)
    - Tabla de tantas filas como R_i y tantas columnas como atributos de R.
    - Conocida la información de una tupla en los ri ser capaz de reproducir la tupla en $r=(a_1,\ldots,a_n)$.
    - Iguales valores del antecedente→Iguales valores del consecuente

Ejemplo: Descomp. sin pérdida (n>2)

- R=(titulación, escuela, alumno, asig, créditos) y F={titulación→escuela; asig→créditos, titulación}
  - R_1=(titulación, escuela), R_2=(asig, créditos, titulación) y R_3=(alumno, asig)

	T	E	Al	As	C
T,E	a_1	a_2	b_{13}	b_{14}	b_{15}
As, C,T	a_1	b_{22}	b_{23}	a_4	a_5
Al,As	b_{31}	b_{32}	a_3	a_4	b_{35}

As→C,T

	T	E	Al	As	C
T,E	a_1	a_2	b_{13}	b_{14}	b_{15}
As, C,T	a_1	b_{22}	b_{23}	a_4	a_5
Al,As	a_1	b_{32}	a_3	a_4	a_5

T→E

	T	E	Al	As	C
T,E	a_1	a_2	b_{13}	b_{14}	b_{15}
As, C,T	a_1	a_2	b_{23}	a_4	a_5
Al,As	a_1	a_2	a_3	a_4	a_5

Conservación de dependencias (I)

- Con la definición
  - $F^+=(F1\cup\ldots\cup Fn)^+$ donde $F_i=F^+\cap R_i$
- Ejemplo:
  - R=(titulación, escuela, alumno, asig, créditos) y F={titulación→escuela; asig→créditos, titulación}
  - R_1=(titulación, escuela), R_2=(asig, créditos, titulación) y R_3=(alumno, asig)

F^+={titulación→escuela; asig→créditos, titulación, escuela}

$F_1=F^+\cap R_1$= {titulación→escuela}

$F_2=F^+\cap R_2$={asig→créditos, titulación}

$F_3=F^+\cap R=\varnothing$

$(F_1\cup\ldots\cup F_n)^+$={titulación→escuela ; asig→créditos, titulación}$^+$={titulación→escuela ; asig→créditos, titulación, escuela}=F^+

Entonces, se conservan las dependencias

Conservación de dependencias (II)

- Con el siguiente algoritmo

```
Bool ConservaDependencias (CjtoDF F, Descomp D){
   Para cada α→β∈F hacer
                  CjtoAtrib M=α;
      mientras M no cambie hacer
            Para cada Ri de D hacer
                        M=M ∪ ((M∩ Ri)+ ∩ Ri)
            fin para
      fin mientras
      Si β⊆M retorna falso;
   fin para
   retorna verdadero;
}
```

Conservación de dependencias (III)

- R=(titulación, escuela, alumno, asig, créditos) y F={titulación→escuela; asig→créditos, titulación}
- R_1=(titulación, escuela), R_2=(asig, créditos, titulación) y R_3=(alumno, asig)
  - titulación→escuela

    M={titulación}

    $M=M\cup((M\cap R_1)^+\cap R_1)$={titulación, escuela}

    escuela⊆M ➡ Se conserva
  - asig→créditos, titulación

    M={asig}

    $M=M\cup((M\cap R_2)^+\cap R_2)$={asig, créditos, titulación}

    {créditos, titulación}⊆M ➡ Se conserva

Algoritmo de descomp. en 3FN (I)

Descomp Sintesis3FN (CjtoDF F, Relacion R){
  Descomp D;
  CjtoDF Fc=ObtenerRecubrimientoCanonico(F);
  CjtoDF G=ReglaUnionAntecedentesIguales(Fc);
  Para cada $\alpha \rightarrow \beta \in G$ hacer
    Si $\exists R_i \subseteq \alpha \cup \beta$ entonces sustituir R_i por $\alpha \cup \beta$;
    Si $\alpha \cup \beta \nsubseteq R_i$ t.q. $R_i \in D$ i=1,…,k entonces
      k=k+1;
      $R_k = \alpha \cup \beta$;
      $D=D \cup \{R_k\}$
    fin si
  Si $\nexists R_i \in D$ que contenga una clave candidata de R entonces
      k=k+1;
      R_k=ClaveCandidata(R);
    fin si
  retorna D;
}

Algoritmo descomp. en 3FN (II)

- ✓ Garantiza que la descomposición es sin pérdida
  - Al menos una relación contiene una clave candidata de R
- ✓ Garantiza que la descomposición conserve las dependencias
  - Crea una relación por cada dependencia funcional de F

Ejemplo descomp. en 3FN

- R=(titulación, escuela, alumno, asig, créditos) y F={alumno, asig→titulación, créditos; titulación →escuela} no está en 3FN
  - F ya es un recubrimiento canónico: Fc=F
  - No existen dependencias con antecedentes iguales: G=Fc
  - R_1=(alumno, asig, titulación, créditos)
  - R_2=(titulación, escuela)
  - No se añade ninguna relación más puesto que la única clave candidata de R ({alumno, asig}) está ya en R_1

Algoritmo descomp. en FNBC (I)

```
Descomp DescompFNBC (CjtoDF F, Relacion R){
    Descomp D;
    D=D∪{R};
    Mientras ∃ Rᵢ∈D t.q. ∃ α→β∈Fᵢ t.q. α no superclave y β⊈α hacer
            Rᵢ₁=α∪β
            Rᵢ₂=Rᵢ-β
            D=D-{Rᵢ}
            D=D∪{Rᵢ₁, Rᵢ₂}
            Obtener Fᵢ₁, Fᵢ₂ como Fᵢ⁺∩Rᵢⱼ;
    fmientras;
    retorna D;
}
```

Algoritmo descomp. en FNBC (II)

- ✓ Garantiza que la descomposición es sin pérdida
- ✘ No garantiza que la descomposición conserve las dependencias

Ejemplo descomp. en FNBC (I)

- R=(titulación, escuela, alumno, asig, créditos) y F={asig→créditos, titulación; titulación → escuela} no está en FNBC
- 1ª Forma
  - La DF asig →créditos, titulación cumple que ni es trivial ni asig es superclave
    - R_1=(asig, créditos, titulación)
    - R_2=(asig, alumno, escuela)
  - R_1 cumple F_1={asig→créditos, titulación} y {asig} es superclave de R_1 luego R_1 está en FNBC
  - R_2 cumple F_2={asig→escuela} y ni asig→escuela es trivial ni {asig} es superclave de R_2, luego R_2 no está en FNBC.
    - R_{21}=(asig, escuela)
    - R_{22}=(asig, alumno)
  - Ambas R_{21} y R_{22} son binarias, luego ambas están en FNBC
  - R queda descompuesto en (asig, créditos, titulación); (asig,escuela) y (asig, alumno)

Ejemplo descomp. en FNBC (II)

- R=(titulación, escuela, alumno, asig, créditos) y F={asig→créditos, titulación; titulación → escuela} no está en FNBC
- 2^a Forma
  - La DF titulación→escuela cumple que ni es trivial ni titulación es superclave
    - R_1=(titulación, escuela)
    - R_2=(titulación, alumno, asig, créditos)
  - R_1 es binaria, luego R_1 está en FNBC.
  - R_2 cumple F_2={asig→créditos, titulación} y ni asig→créditos, titulación es trivial ni asig es superclave de R_2, luego R_2 no está en FNBC.
    - R_{21}=(asig, créditos, titulación)
    - R_{22}=(asig, alumno)
  - R_{21} cumple F_{21}={asig→créditos, titulación} donde asig es superclave de R_{21}, luego está en FNBC. R_{22} es binaria, luego está en FNBC
  - R queda descompuesto en (titulación, escuela); (asig, créditos, titulación) y (asig, alumno)

Ejemplo descomp. en FNBC (III)

- R=(titulación, escuela, alumno, asig, créditos) y F={asig→créditos, titulación; titulación → escuela} no está en FNBC
  - 1^{a} Forma: R_1=(asig, escuela); R_2=(asig, créditos, titulación) y R_3=(asig, alumno)
    - ✘ No conserva las dependencias (se pierde titulación →escuela)
  - 2^{a} Forma: R_1=(titulación, escuela); R_2=(asig, créditos, titulación) y R_3=(asig, alumno)
    - ✓ Sí conserva las dependencias