Bases de Datos

Tema: Teoría de diseño de Bases de Datos Relacionales (II)

Bibliografía

- *Ramez A. Elmasri, Shamkant B. Navathe*. ***Fundamentos de Sistemas de Bases de Datos*** *(5º edic.).* Prentice-Hall. 2007 [cap. 10]
- De Miguel, A., Piattini. ***Fundamentos y modelos de bases de datos*** *(2ª edic.).*, Rama. [*cap. 8*]
- A. Silberschatz, Korth, Sudarshan. ***Fundamentos de Bases de Datos*** *(5ª edic.).* McGraw-Hill [*cap. 7*]

5.9. Introducción a la Normalización

➔ *Ejemplo de diseño inadecuado*

ESCRIBE

AUTOR	NACIONALIDAD	COD_LIBRO	TITULO	EDITORIAL	AÑO
Date, C.	Norteamericana	23433	Databases	Adisson-W.	1990
Date, C.	Norteamericana	54654	SQL Standard	Adisson-W.	1986
Date, C.	Norteamericana	53235	Guide To Ingres	Adisson-W.	1988
Codd, E.	Norteamericana	97875	Relational M.	Adisson-W.	1990
Gardarin	Francesa	34245	Base de Datos	Paraninfo	1986
Gardarin	Francesa	55366	Comparación BD	Eyrolles	1984
Valduriez	Francesa	86754	Comparación BD	Eyrolles	1984
Kim, W.	Norteamericana	32176	OO Databases	ACM Press	1989
Lochovsky	Canadiene	23456	OO Databases	ACM Press	1989

- PROBLEMAS:
  - Redundancia
  - Anomalías (de inserción, borrado y modificación)

5.9. Introducción a la Normalización

- La normalización de datos puede considerarse un proceso de análisis de los esquemas de relación basándose en sus **DFs y claves** para alcanzar las propiedades deseables de:
  - minimizar la redundancia
  - minimizar las anomalías de inserción, eliminación y actualización

- El proceso de normalización proporciona a los diseñadores los siguientes aspectos:
  - un marco formal para analizar las relaciones basándose en sus claves y en las DF entre atributos.
  - una serie de pruebas que pueden efectuarse sobre relaciones individuales de modo que la BD relacional pueda normalizarse hasta el grado deseado (1FN, 2FN, 3FN, FNBC).

5.10. Descomposición en esquemas

- El proceso de normalización por descomposición parte de la *relación universal* (relación compuesta por todos los atributos) **descomponiéndola** en subrelaciones sin anomalías.

- Dado un esquema R(T, L), donde $T=\{A_1, A_2, \ldots, A_n\}$, una **descomposición** de R es la sustitución de R por un conjunto de relaciones $R_1, R_2, \ldots, R_k$, tales que:
  - $R_i= \prod_{A1,\ A2,\ \ldots Aj}(R)$
  - $R_1 \infty R_2 \infty \ldots \infty R_k$ da lugar al **mismo esquema** de R y la **unión de sus atributos** es $A_1, A_2, \ldots, A_n$, es decir, el conjunto de atributos del esquema inicial.

5.10. Descomposición en esquemas → *Ejemplo*

- Posible descomposición para la relación **LIBRO**

LIBRO

COD_LIBRO	EDITORIAL	PAIS
654654	Ra-Ma	España
665465	Ra-Ma	España
876545	Paraninfo	España
987456	Anaya	España
965842	Addison-w.	EEUU

LIBRO1

COD_LIBRO	PAIS
654654	España
665465	España
876545	España
987456	España
965842	EEUU

EDITORIAL1

EDITORIAL	PAIS
Ra-Ma	España
Paraninfo	España
Anaya	España
Addison-w.	EEUU

5.11. Descomposición con la propiedad de unión sin pérdida de información (LJ)

- La descomposición anterior presenta *problemas*:

LIBRO1

COD_LIBRO	PAIS
654654	España
665465	España
876545	España
987456	España
965842	EEUU

∞

EDITORIAL1

EDITORIAL	PAIS
Ra-Ma	España
Paraninfo	España
Anaya	España
Addison-w.	EEUU

COD_LIBRO	EDITORIAL	PAIS
654654	Ra-Ma	España
654654	**Paraninfo**	**España**
654654	**Anaya**	**España**
665465	Ra-Ma	España
665465	**Paraninfo**	**España**
665465	**Anaya**	**España**
876545	**Ra-Ma**	**España**
876545	Paraninfo	España
876545	**Anaya**	**España**
……	…...	……

Tuplas Espurias

5.11. Descomposición con la propiedad de unión sin pérdida de información (LJ)

- En el proceso de normalización **por descomposición** debe comprobar, no sólo que cada relación esté en una determinada forma normal (1FN, 2FN, 3FN), además cada una de las relaciones obtenidas por descomposición deberá cumplir una serie de propiedades, en concreto:
  1. **Unión sin pérdida**: Garantiza que se obtengan las mismas instancias de la relación original a partir de las subrelaciones.
  2. **Preservación de atributos**: Todo atributo de la relación original R deberán aparecer al menos en una relación Ri de la descomposicón.
  3. **Preservación de dependencias**: Las mismas restricciones que existían en la relación original deben existir sobre las subrelaciones.
- Si una descomposición cumple estas tres propiedades, se dice que es una **DESCOMPOSICIÓN SIN PÉRDIDA**.

5.11. Descomposición con la propiedad de unión sin pérdida de información (LJ)

- En el proceso de normalización por descomposición debe comprobar que cada una de las relaciones obtenidas por descomposición cumpla una serie de propiedades, entre ellas, la de **unión sin pérdida**:
  - Sea **R** un esquema relación que se descompone en los esquemas $R_1,\ldots R_k$, y **L** un conjunto de dependencias funcionales
  - La descomposición tiene la propiedad de unión sin pérdida de información respecto de L, es decir es ***join sin pérdida*** (*lossless join, LJ*), si para toda ocurrencia r del esquema R, r(R)

$$\mathbf{r = \prod_{T1}(r) \infty \prod_{T2}(r) \infty \ldots \infty \prod_{Tk}(r)}$$

    - Es decir, cada relación *r* se reconstruye como el join natural de sus proyecciones sobre cada R_i.
    - En caso de que una descomposición no posea esta propiedad, es posible que surjan tuplas espurias.

5.11. Descomposición con la propiedad de unión sin pérdida de información (LJ)

- En el caso de la relación LIBRO, se podría haber descompuesto en los siguientes esquemas:

  LIBRO2 (COD_LIBRO, PAIS)
  EDITORIAL2 (COD_LIBRO, EDITORIAL)

  - En este caso, **LIBRO= LIBRO2 ∞ EDITORIAL2**.
    - **NO aparecen tuplas espurias.**

  - La condición necesaria y suficiente para que una descomposición se produzca sin pérdida de información es que ***el atributo común de las dos relaciones sea clave, al menos, en una de ellas***.

5.11.1 Test de la propiedad LJ → *Algoritmo*

- Sea una descomposición de R (T, L) con:
  $T=\{A_1, A_2,\ldots, A_m\}$
  $L=\{x \to y \ / \ x \cup y \subseteq T\}$
  $\rho = \{R_1,\ldots, R_k\}$.
- Para verificar el cumplimiento de la propiedad LJ en la descomposición ρ se va a utilizar el siguiente algoritmo:
  1. Construir matriz con **m** columnas (una por cada atributo $\mathbf{A_j}$) y **k** filas (una por cada esquema de relación $\mathbf{R_i}$)
  2. Colocar en la fila **i** columna **j** la letra $\mathbf{a_j}$ si $\mathbf{A_j}$ está en $\mathbf{R_i}$, sino colocar $\mathbf{b_{ij}}$.
  3. Considerar ahora cada una de las dependencias de **L**.
     - Si para $\mathbf{X \to Y}$ se encuentran dos filas que coincidan las entradas correspondientes a **X**, se igualan las correspondientes a **Y**:
       - si el símbolo es $\mathbf{a_j}$, lo hacemos $\mathbf{a_j}$
       - si ambos son **b**, igualamos los subíndices de cualquiera de ellos a los del otro.
  4. Si se encuentra una fila llena de **a's**, la descomposición verifica la propiedad LJ y no la verifica en caso contrario.

5.11.1 Test de la propiedad LJ

➔ *Ejemplo*

- R(T, L), tal que:
  - T = {A, B, C, D, E}
    L = {A → C, B → C, C → D, DE → C, CE → A}
    ρ = {AD, AB, BE, CDE, AE}
  - **Paso 1**:Construir matriz con **m** columnas (una por cada atributo $\mathbf{A_j}$) y **k** filas (una por cada esquema de relación $\mathbf{R_i}$)

	A	B	C	D	E
AD					
AB					
BE					
CDE					
AE					

5.11.1 Test de la propiedad LJ

➔ *Ejemplo*

R(T, L), tal que:

T = {A, B, C, D, E}

L = {A → C, B → C, C → D, DE → C, CE → A}

ρ = {AD, AB, BE, CDE, AE}

- **Paso 2**: Colocar en la fila **i** columna **j** la letra $\mathbf{a_j}$ si $\mathbf{A_j}$ está en $\mathbf{R_i}$, sino colocar $\mathbf{b_{ij}}$.

	A	B	C	D	E
AD	a1	b12	b13	a4	b15
AB	a1	a2	b23	b24	b25
BE	b31	a2	b33	b34	a5
CDE	b41	b42	a3	a4	a5
AE	a1	b52	b53	b54	a5

5.11.1 Test de la propiedad LJ

→ *Ejemplo*

R(T, L), tal que:

T = {A, B, C, D, E}

L = {A → C, B → C, C → D, DE → C, CE → A}

ρ = {AD, AB, BE, CDE, AE}

- **<u>Paso 3</u>**: Considerar ahora cada una de las dependencias de **L**.
  - Si para **X→Y** se encuentran dos filas que coincidan las entradas correspondientes a **X**, se igualan las correspondientes a **Y**:
    - si el símbolo es $\mathbf{a_j}$, lo hacemos $\mathbf{a_j}$
    - si ambos son **b**, igualamos los subíndices de cualquiera de ellos a los del otro.

	A	B	C	D	E
AD	a1	b12	b13	a4	b15
AB	a1	a2	b23	b24	b25
BE	b31	a2	b33	b34	a5
CDE	b41	b42	a3	a4	a5
AE	a1	b52	b53	b54	a5

A→C

	A	B	C	D	E
AD	a1	b12	b13	a4	b15
AB	a1	a2	**b13**	b24	b25
BE	b31	a2	b33	b34	a5
CDE	b41	b42	a3	a4	a5
AE	a1	b52	**b13**	b54	a5

5.11.1 Test de la propiedad LJ

→ *Ejemplo*

R(T, L), tal que:

T = {A, B, C, D, E}

L = {A → C, B → C, C → D, DE → C, CE → A}

ρ = {AD, AB, BE, CDE, AE}

A→C	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

B→C	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	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

DE→C	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

5.11.1 Test de la propiedad LJ → *Ejemplo*

R(T, L), tal que:

$T = \{A, B, C, D, E\}$
$L = \{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$
$\rho = \{AD, AB, BE, CDE, AE\}$

- **Paso 3:**

CE→A	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_1	a_2	a_3	a_4	a_5
CDE	b_{41} a_1	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	a_3	a_4	a_5

- **Paso 4**: Si se encuentra una fila llena de **a's**, la descomposición verifica la propiedad LJ y no la verifica en caso contrario.
  - La tercera fila es (a1, a2, a3, a4, a5), así que la descomposición **verifica la propiedad LJ**, pudiendo por tanto reconstruirse cualquier ocurrencia r del esquema R a partir de la unión natural de sus proyecciones.

5.11.1 Test de la propiedad LJ → *Ejemplo*

- R(T, L), tal que:
  - T = {NSS, NOMBREE, NUMEROP, NOMBREP, LOCALIZACIONP, HORAS}
    L = {NSS → NOMBREE; NUMEROP → NOMBREP,LOCALIZACIONP; NSS,NUMEROP→HORAS}

    ρ = {**R1**, **R2**, **R3**}

    R1= **EMP**= {NSS, NOMBREE}

    R2= **PROYECTO**= {NUMEROP, NOMBREP, LOCALIZACIONOP}

    R3= **TRABAJA_EN**= {NSS, NUMEROP, HORAS}

5.11.1 Test de la propiedad LJ

➔ *Teorema*

- **Teorema**: R_1, R_2 es una descomposición **join sin pérdida de *R*** con respecto al conjunto de dependencias funcionales L si y sólo si vale, por lo menos, alguna de las siguientes dependencias de L^+

  $(T_1 \cap T_2) \rightarrow T_1\text{-}T_2$, o bien

  $(T_1 \cap T_2) \rightarrow T_2\text{-}T_1$

  Esto quiere decir que para que una descomposición de R en 2 subrelaciones R_1 y R_2 se produzca sin pérdida de información es necesario que **el atributo o atributos comunes** de las dos relaciones **sea(n) clave en alguna de ellas**.

- **Ejemplo**: Sea el siguiente conjunto de DF's

  Id# → Nombre, Dirección
  C# → Descripción
  Id#,C# → Grado

  - ¿es R_1(Id#, Nombre, Dirección) y R_2(Id#, C#, Descripción, Grado) una descomposición con la propiedad de join sin pérdida (LJ)?

5.12. Descomposición con preservación de dependencias

- Otra propiedad deseable de la descomposición de un esquema de relación $\rho=(R_1, \ldots, R_n)$ es la de **preservación de las dependencias**. Es decir, que las **DF's originales se mantengan**, ya que éstas recogen la semántica del mundo real, las restricciones que ha de cumplir la base de datos.

- No será necesario que las DF's **exactas** especificadas en la relación original aparezcan en las relaciones individuales de la descomposición. Basta con que la **unión de las DF's** que se cumplen en las relaciones individuales **sea <u>equivalente</u>** al conjunto de DF's de la relación original. Es decir, que cada DF del esquema original aparezca directamente en uno de los esquemas R_i , o bien, pueda inferirse de las DF's que aparecen en algún R_i.

5.12. Descomposición con preservación de dependencias

- **¿Cómo saber qué DF's le corresponden a cada Ri tras realizar la descomposición?**
  - Dado un conj. *L* de DF's sobre *R*, las DF's de cada R_i se obtienen mediante la **proyección** del conjunto de DF's de *R* sobre R_i. Dicha proyección está formada por el **conjunto de dependencias X → Y en L^+**, tal que los atributos de $X \cup Y$ estén todos contenidos en R_i.
- *Ejemplo*:
  - Dada la siguiente relación:
    COCHE (NM, MARCA, TIPO, POTENCIA, COLOR)
    L = {NM → TIPO, TIPO → MARCA, TIPO → POTENCIA, NM → COLOR}
    L^+= L ∪ {NM→MARCA, NM→POTENCIA}
  - Se tiene la siguiente descomposición $\rho(R_1, R_2)$ para **COCHE**:
    R_1(NM, TIPO, COLOR) **R_2**(TIPO, MARCA, POTENCIA)
  - **¿Qué DF's corresponden a cada R_i?**
    - L_1= { }
    - L_2= { }

5.12. Descomposición con preservación de dependencias

- Una descomposición $\rho = \{R_1, R_2, \ldots R_k\}$ del esquema R(T, L) es una **descomposición con preservación de dependencias** si la unión de las DF's de las relaciones individuales es **equivalente** al conjunto de DF original. Es decir:

  L es **equivalente** a $\{\cup_{i=(1,k)} L_i\}$, es decir:

$$L^+ = \{\cup_{i=1..n} L_i\}^+$$

5.12. Descomposición con preservación de dependencias

➔ *Ejemplo*

- **COCHE** (NM, MARCA, TIPO, POTENCIA, COLOR)
  L = {NM → COLOR, TIPO → POTENCIA, TIPO → MARCA, NM → TIPO}
  L^+ = L ∪ {NM → POTENCIA, NM → MARCA}

- **<u>Descomposición 1</u>**: ρ= {R1, R2}
  - **R_1** (NM, TIPO, COLOR) **L_1** = {NM → TIPO, NM → COLOR}
  - **R_2** (TIPO, MARCA, POTENCIA) **L_2** = {TIPO → MARCA, TIPO→POTENCIA}

- ¿La descomposición ρ={R1, R2} **preserva las dependencias**?

  ¿¿ $L^+ = \{\cup_{i=(1,k)} L_i\}^+$??

  - **L1 ∪ L2** = {NM → TIPO, NM → COLOR, TIPO → MARCA, TIPO → POTENCIA}
  - **$\{L1 \cup L2\}^+$**= {NM→TIPO, NM→COLOR, TIPO→MARCA, TIPO→POTENCIA, **NM→POTENCIA, NM → MARCA**}

SI conserva las DF's

5.12. Descomposición con preservación de dependencias

➔ *Ejemplo*

- **COCHE** (<u>NM</u>, MARCA, TIPO, POTENCIA, COLOR)
  L = {NM → COLOR, TIPO → POTENCIA, TIPO → MARCA, NM → TIPO}
  L^+ = L ∪ {NM → POTENCIA, NM → MARCA}

- <u>**Descomposición 2**</u>:
  - **R_3** (NM, TIPO) — **L_3** = {NM → TIPO}
  - **R_4** (TIPO, POTENCIA, COLOR) — **L_4** = {TIPO → POTENCIA }
  - **R_5** (TIPO, MARCA) — **L_5** = {TIPO → MARCA}

- ¿La descomposición $\rho=\{R_3, R_4, R_5\}$ **preserva las dependencias**?

  ¿¿ $L^+ = \{\cup_{i=(1,k)} L_i\}^+$??

  - $L_3 \cup L_4 \cup L_5$ = {NM → TIPO, TIPO → POTENCIA, TIPO → MARCA }
  - $\{L_3 \cup L_4 \cup L_5\}^+$ = {NM→TIPO, TIPO→POTENCIA, TIPO→MARCA, **NM→POTENCIA, NM→MARCA**}

  NO conserva las DF's (pierde NM → COLOR)

5.12.1. Algoritmo de test de preservación de dependencias

- Para comprobar que la descomposición de un esquema preserva dependencias hay que comprobar que la unión de las DF's de las relaciones individuales es **equivalente** al conjunto de DF original :

$$\mathbf{L^+ = \{ \cup_{(i=1,k)} L_i \}^+}$$

- **Problema**: el tiempo de cálculo de L^+ depende *exponencialmente* del número de dependencias de L. Por ello se propone un **algoritmo de test de preservación de dependencias** cuyo tiempo de ejecución es una función *polinomial* del número de dependencias en L.

5.12.1. Algoritmo de test de preservación de dependencias

- **T-operación**: Se define una T-operación sobre el descriptor **Z** respecto del conjunto L de DF's como la sustitución de **Z** por $\mathbf{Z} \cup ((\mathbf{Z} \cap \mathbf{T})^{+} \cap \mathbf{T})$ **donde el cierre se calcula respecto a L**.
- **Algoritmo**:
  - Sea $\mathbf{G} = \cup_{(i=1,k)} L_i$
  - Considerar cada DF $\mathbf{X} \rightarrow \mathbf{Y}$ de L y verificar si $\mathbf{X}^{+}$ **respecto de G contiene a Y**

```
Z := X
Mientras Z cambie
      Para i := 1 a K hacer
            Z := Z ∪ (( Z ∩ Ti)+ ∩ Ti
```

nº de subrelaciones de ρ

  - En cada etapa se modifica el valor del descriptor al aplicar las **k** T_i-operaciones (i=1,.,k).
  - Si al acabar una etapa no hubiese cambio, el resultado es el cierre Z^{+} respecto G (recordar que G es la unión de los L_i).

  - Si Y es un subconjunto de Z ($Y \subseteq Z$), entonces $X \rightarrow Y$ está en G^{+}, es decir, si para toda DF $X \rightarrow Y \in L$, ésta se encuentra en G^{+}, ($X \rightarrow Y \in G^{+}$), hay preservación de dependencias y no la habrá en caso contrario.

5.12.1. Algoritmo de test de preservación de dependencias → *Ejemplo*

- Dado un esquema R(T, L)
  $T = \{A, B, C, D\}$
  $L = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$
  $\rho = \{AB, BC, CD\}$
  ¿preserva las dependencias?

- Es obvio que las dependencias $A \rightarrow B$, $B \rightarrow C$ y $C \rightarrow D$ se conservan en las proyecciones, por lo que será necesario aplicar el Algoritmo para **verificar la conservación de $D \rightarrow A$**

5.12.1. Algoritmo de test de preservación de dependencias ➔ *Ejemplo*

R(T, L), tal que:

$T = \{A, B, C, D\}$

$L = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$

$\rho = \{AB, BC, CD\}$

ALGORITMO

- Sea **G =** $\cup_{(i=1,k)} L_i$
- Considerar cada DF **X→Y** de L y verificar si X^+ respecto de G contiene a Y

  Z := X

  Mientras Z cambie

  Desde i := 1 a k hacer

  $Z := Z \cup ((Z \cap T_i)^+ \cap T_i$

- Calcular si se preserva la DF **D → A**
  - **Z:= D**
  - **k = 1**
    - $D \cup ((D \cap AB)^+ \cap AB) = D$
      $D \cup ((D \cap BC)^+ \cap BC) = D$
      $D \cup ((D \cap CD)^+ \cap CD) = D \cup (D^+ \cap CD) = D \cup (ABCD \cap CD) = CD$
  - **k = 2**
    - $CD \cup ((CD \cap AB)^+ \cap AB) = CD$
      $CD \cup ((CD \cap BC)^+ \cap BC) = CD \cup (ABCD \cap BC) = BCD$
      $BCD \cup (((BCD \cap CD)^+ CD) = BCD$
  - **k = 3**
    - $BCD \cup ((BCD \cap AB)^+ \cap AB) = BCD \cup (B^+ \cap AB) = BCD \cup (ABCD \cap AB) = ABCD$
  - Como $A \subseteq Z$, $D \rightarrow A \in G^+$ ➔ se cumple la preservación de dependencias.