Bases de Datos

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

5.13. Formas Normales de Codd.

- **Teoría de la Normalización**: Técnica formal para agrupar datos que determina la ***bondad*** de un diseño de una base de datos.
  - 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 DEPENDENCIAS FUNCIONALES 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).
  - Se basa en la descomposición de una relación en subrelaciones EQUIVALENTES A LA ORIGINAL (preservan dependencias y verifican LJ).

5.13. Formas Normales de Codd

- **Teoría de la Normalización**
  - Se centra en lo que se conoce como **formas normales**. Se dice que un esquema de relación está en una determinada forma normal si satisface un conjunto específico de propiedades.

  - Las tres primeras formas normales (1FN, 2FN y 3FN) tienen por objetivo realizar una **descomposición sin pérdida** a partir del concepto de DF.
    - **Preservación de Atributos** $R = \cup_{(i=1,k)} R_i$
    - **Join sin pérdida** $\prod_{R1}(r) \infty \prod_{R2}(r) \infty \ldots \infty \prod_{Rm}(r) = r \quad \forall\, r$
    - **Preservación de Dependencias** $L^+ = \{\cup_{(i=1,k)} L_i\}^+$

5.13. Formas Normales de Codd
1FN

- **Primera Forma Normal :** Una relación **R** está en **1FN** si todos los atributos contienen un valor atómico (simple, indivisible).
  - Es una restricción inherente al propio modelo relacional.
  - ***Proceso de Normalización***:
    - Si un atributo es multivaluado, es posible representarlo como valores atómicos, mediante 2 opciones:

      Opción 1: crear una tupla para cada valor diferente ⇒ clave = (clave + atributo)

      Opción 2: crear una relación con los diferentes valores del atributo

5.13. Formas Normales de Codd

1FN

- Normalizar:

LIBRO

COD_LIBRO	TITULO	AUTOR
654654	Data models	Tsichiritzis Lochovosky
665465	A guide to DB2	Date
876545	Bases de Datos	Gardarin Valduriez

Opción1 para poner la relación LIBRO en 1FN

LIBRO

COD_LIBRO	TITULO	AUTOR
654654	Data models	Tsichiritzis
654654	Data models	Lochovosky
665465	A guide to DB2	Date
876545	Bases de Datos	Gardarin
876545	Bases de Datos	Valduriez

5.13. Formas Normales de Codd

1FN

- Normalizar:

LIBRO

COD_LIBRO	TITULO	AUTOR
654654	Data models	Tsichiritzis Lochovosky
665465	A guide to DB2	Date
876545	Bases de Datos	Gardarin Valduriez

Opción 2 para poner la relación LIBRO en 1FN

LIBRO

COD_LIBRO	TITULO
654654	Data models
665465	A guide to DB2
876545	Bases de Datos

AUTOR

COD_LIBRO	AUTOR
654654	Tsichiritzis
654654	Lochovosky
665465	Date
876545	Gardarin
876545	Valduriez

5.13. Formas Normales de Codd

2FN

- **DF Total:** Una DF, X➔Y, se dice que es una **dependencia total** cuando $\nexists X' \subset X: X' \rightarrow Y$.
- **Segunda Forma Normal:** Una relación R está en **2FN** si está en 1FN y, además, si cada atributo no primo A, la dependencia, **Clave➔A** se cumple y es total. Es decir, no pueden existir dependencias funcionales parciales de atributos no primos respecto a ninguna clave.
  - Un esquema que no se encuentre en 2FN, puede traducirse en varios esquemas que sí lo estén.
  - ***Proceso de Normalización***:
    1. Crear tantas nuevas relaciones como dependencias funcionales no sean completas.

Atributo PRIMO: Aquel que forma parte de cualquier clave candidata

5.13. Formas Normales de Codd
2FN

- ***Ejemplo 2FN*:**
  - **SUMINISTRADOR**(APELLIDO, ARTICULO, DIRECCION, PRECIO)

    L = {PROVEEDOR, ARTICULO→PRECIO; PROVEEDOR→DIRECCION}

    K = {PROVEEDOR,ARTICULO}

  - **¿está en 2FN?**
    - **NO**. Existe un atributo NO CLAVE (DIRECCIÓN) que **no depende totalmente** de la clave.

  - **Proceso de Normalización a 2FN**:
    - Descomponer y crear una nueva relación para cada parte de la clave con su atributo o atributos dependientes.
    - Crear otra relación con las claves originales y los atributos que dependan funcionalmente DE MANERA TOTAL de ellas.

SUMINISTRADOR (PROVEEDOR, DIRECCION) K = (PROVEEDOR)
PRODUCTO (PROVEEDOR, ARTICULO, PRECIO) K = (PROVEEDOR,ARTICULO)

5.13. Formas Normales de Codd
3FN

- Se basa en el concepto de Dependencia Funcional Transitiva

- **Tercera Forma Normal:** Una relación se encuentra en 3FN si, satisface la 2FN y ninguno de los atributos no primos dependen transitivamente de una clave candidata. Es decir, no pueden existir dependencias funcionales transitivas de atributos no primos respecto a alguna de la (s) clave(s).
  - **Proceso de Normalización a 3FN**:
    - Descomponer la relación en los atributos definidos por la dependencia funcional responsable de la transitividad.

Atributo PRIMO: Aquel que forma parte de cualquier clave candidata

5.13. Formas Normales de Codd

3FN

- Una relación que no esté en 3FN presenta problemas de **redundancia**:
- ***Ejemplo***:

TEMPLE (NUMEM,NOMEM,SALAR,NUMDE,NOMDE,TIDIR)
L = { NUMEM → (NOMEM,SALAR,NUMDE)
NUMDE → (NOMDE,TIDIR)}
$\mathbf{K}_{\mathbf{TEMPLE}}$ {NUMEM}

TEMPLE	NUMEM	NOMEM	SALAR	NUMDE	NOMDE	TIDIR
	1	Emp1	1000	110	Ventas	T
	2	Emp2	2000	110	Ventas	T
	3	Emp3	1000	110	Ventas	T
	4	Emp4	3000	110	Ventas	T
	5	Emp5	5000	121	Ventas	T

¿Está en 2FN?

¿Existe Redundancia?

5.13. Formas Normales de Codd
3FN

- ***Ejemplo* 3FN:**
  - **COCHE** (NM, MARCA, TIPO, POTENCIA, COLOR)
    L = {NM → COLOR, TIPO → POTENCIA, TIPO → MARCA, NM → TIPO}
    K= {NM}
  - **¿está en 3FN?**
    - **NO**. Porque existen atributos no clave (MARCA, POTENCIA) que dependen otro atributo no clave (TIPO) a través de las DF's TIPO→MARCA y TIPO→POTENCIA
  - En el ejemplo anterior, MARCA y POTENCIA dependen transitivamente de TIPO, por tanto no está en 3FN.
    - ***Proceso de Normalización a 3FN***

COCHE (NM, TIPO, COLOR) **K= {NM}**
MODELO (TIPO, MARCA, POTENCIA) **K= {TIPO}**

5.13. Formas Normales de Codd

- **Resumen**

FN	COMPROBACIÓN	NORMALIZACIÓN
1FN	Una relación no debería tener ningún atributo no atómico.	Formar relaciones nuevas para cada atributo no atómico
2FN	Para las relaciones en las que la clave primaria contiene múltiples atributos, ningún atributo no clave debería depender funcionalmente de una parte de la clave primaria.	Descomponer y crear una nueva relación para cada clave parcial con su atributo o atributos dependientes. Asegurarse de que se mantiene una relación con la clave primaria original y todos los atb's que dependan funcionalmente de ella.
3FN	No pueden existir dependencias transitivas por parte de un atributo no clave respecto a una clave primaria (superclave).	Descomponer y crear una relación que incluya el atributo o atributos no clave que determinen funcionalmente a otro u otros atributos no clave.

- Siempre es posible descomponer una relación en **3FN** sin pérdida:
  - Preservación de atributos
  - Preservación de DF's
  - JOIN sin pérdida

5.14. Descomposición en 3FN de Codd con preservación de DF's

- **Algoritmo**
  - **Entrada**: Un esquema de relación R(T, L) tal que L es un **r.n.r**
  - **Salida**: Una descomposición con preservación de dependencias ($\rho = \{R_1, R_2, \ldots, R_k\}$) de R tal que cada R_i está en 3FN con respecto a L_i.
  - **Proceso**:
    1. para cada antecedente X de L:
       - crear Ri = {X, A1, A2, …, Ak}
         - donde $X \rightarrow A1, \ldots, X \rightarrow Ak$ sean las únicas DF's en L con X antecedente
    2. Crear un esquema R_p con los atributos ***sobrantes***, los que no se haya podido colocar en ninguna relación para garantizar la preservación de los atributos.

- ***Ejemplo***: Sintetizar a 3FN con preservación de DF's el esquema siguiente

COCHE (NM, MARCA, TIPO, POTENCIA, COLOR) **K** (NM)
L = {NM → COLOR, TIPO → POTENCIA, TIPO → MARCA, NM → TIPO} (es un **r.n.r.**)

L1 = {NM → COLOR, NM → TIPO} $\mathbf{K_{R1}}$ = (NM)
R1 =(NM, TIPO, COLOR)

L2 = {TIPO → POTENCIA, TIPO → MARCA} $\mathbf{K_{R2}}$ = (TIPO)
R2 =(TIPO, POTENCIA, MARCA)

ρ ={R1, R2}

5.14. Descomposición en 3FN de Codd con preservación de DF's

- **Algoritmo**
  - **Entrada**: Un esquema de relación R(T, M) tal que L es un r.n.r
  - **Salida**: Una descomposición con preservación de dependencias ($\rho = \{R_1, R_2, \ldots, R_k\}$) de R tal que cada R_i está en 3FN con respecto a L_i.
  - **Proceso**:
    1. para cada antecedente X de L:
       - crear Ri = {X, A1, A2, …, Ak}
         - donde $X \rightarrow A1, \ldots, X \rightarrow Ak$ sean las únicas DF's en L con X antecedente
    2. Crear un esquema R_p con los atributos ***sobrantes***, los que no se haya podido colocar en ninguna relación para garantizar la preservación de los atributos.
- ***Ejemplo***: Sintetizar a 3FN con preservación de DF's el esquema siguiente

R(T,L)

T = {A,B,C,D,E} **k**={ACE}

L = {A → B, C → D}

R1(A,B) K_{R1}={A} R2(C,D) K_{R2}={C} **R3(E)** **K_{R3}={E}**

ρ ={R1, R2, R3}

5.14. Descomposición en 3FN de Codd con preservación de DF's

- **Algoritmo**
  - **Entrada**: Un esquema de relación R(T, L) tal que L es un r.n.r
  - **Salida**: Una descomposición con preservación de dependencias ($\rho = \{R_1, R_2, \ldots, R_k\}$) de R tal que cada R_i está en 3FN con respecto a L_i.
  - **Proceso**:
    1. para cada antecedente X de L:
       - crear Ri = {X, A1, A2, …, Ak}
         - donde $X \rightarrow A1, \ldots, X \rightarrow Ak$ sean las únicas DF's en L con X antecedente
    2. Crear un esquema R_p con los atributos ***sobrantes***, los que no se haya podido colocar en ninguna relación para garantizar la preservación de los atributos.
- ***Ejercicio***: Sintetizar a 3FN con preservación de DF's el esquema siguiente

R(T,L)

T = {DNIE, NumProy, SalarioE, TelE, Numde, NomProy, LocProy}

L = {DNIE→SalarioE, TelE, Numde
NumProy→NomProy, LocProy
DNIE, NumProy→SalarioE, TelE, Numde, NomProy, LocProy}

5.15. Descomposición en 3FN de Codd con preservación de DF's y verificación de la propiedad LJ

- El algoritmo anterior se puede modificar para producir una descomposición de una relación R que:
  - **Conserve las DF's**
  - **Verifique la propiedad LJ**
  - **Cada Ri de la descomposición esté en 3FN**

5.15. Descomposición en 3FN de Codd con preservación de DF's y verificación de la propiedad LJ

- Sea una descomposición con preservación de dependencias ($\rho = \{R_1, R_2, \ldots, R_k\}$) de R tal que cada R_i está en 3FN con respecto a L_i (descomposición obtenida por el algoritmo anterior).
- Una forma sencilla de garantizar que ρ preserva las dependencias y verifica la propiedad LJ es añadirle la proyección $\Pi_X(R)$, donde X **es una clave de R**.
- Es decir, la descomposición $\rho' = \rho \cup \{X\}$ será una descomposición de R
  - con todos sus esquemas en 3FN
  - con preservación de dependencias
  - verifica la propiedad LJ

5.15. Descomposición en 3FN de Codd con preservación de DF's y verificación de la propiedad LJ

- Algoritmo
  - **Entrada**: Un esquema de relación R(T, L) tal que L es un **r.n.r**
  - **Salida**: Una descomposición con preservación de dependencias y verificación de la propiedad LJ ($\rho = \{R_1, R_2, \ldots, R_k\}$) de R tal que cada R_i está en 3FN con respecto a L_i.
  - **Proceso**:
    1. para cada antecedente X de L:
       - crear Ri = {X, A1, A2, …, Ak}
         - donde $X \rightarrow A1, \ldots, X \rightarrow Ak$ sean las únicas DF's en L con X antecedente
    2. **Si ninguna de las relaciones Ri contiene una clave de R, crear una relación adicional que contenga atributos que formen una clave de R**
- ***Ejemplo***: Sintetizar a 3FN con preserv. de DF's y verif. de LJ el esquema siguiente:

R(T,L)

T = {A,B,C,D,E} **k**={ACE}
L = {A → B, C → D}

R1(A,B) K_{R1}={A} R2(C,D) K_{R2}={C} **R3(A,C, E)** **K_{R3}={ACE}**

ρ = {R1, R2, R3}

Tras el paso 1, ninguna Ri contiene la clave de R, luego se crea una relación adicional

5.16. Forma Normal de Boyce-Codd

- Con la 3FN todavía pueden seguir existiendo problemas de redundacias
- *Ejemplo*:

ENSEÑAR(ESTUDIANTE, MATERIA, PROFESOR)

L = { (ESTUDIANTE, MATERIA) → PROFESOR, PROFESOR→MATERIA,

(ESTUDIANTE,PROFESOR) → MATERIA}

$\mathbf{K}_{\mathbf{ENSEÑAR}}$ {(ESTUDIANTE,MATERIA), (ESTUDIANTE, PROFESOR)}

ENSEÑAR	ESTUDIANTE	MATERIA	PROFESOR
	Campos	BDI	P1
	Pérez	BDI	P1
	Pérez	S.O.	P3
	Ochoa	BDI	P1
	Ochoa	S.O	P3

¿Está en 3FN?

¿Existe Redundancia?

5.16. Forma Normal de Boyce-Codd

- **Forma Normal de BOYCE-CODD:** Una relación R está en FNBC cuando para toda **Dependencia Funcional no trivial X→A**, se cumple que **X es clave o superclave**

  - ***Ejemplo***:

    ¿está en FNBC?

    ENSEÑAR(ESTUDIANTE, MATERIA, PROFESOR)

    L = { (ESTUDIANTE, MATERIA) → PROFESOR, PROFESOR→MATERIA,

    (ESTUDIANTE,PROFESOR) → MATERIA}

    $\mathbf{K_{ENVIOS}}$ {(ESTUDIANTE,MATERIA), (ESTUDIANTE, PROFESOR)}

  - Es más restrictiva que la 3FN, ya que únicamente permite la 1ª condición de ésta.
  - Surge debido a que la 3FN se mantiene ciertos problemas de redundancia en relaciones que presentan claves candidatas compuestas que se solapan.
  - Por norma general, casi todas las relaciones que están en 3FN también están en FNBC, excepto que la relación tenga dos o más claves candidatas y que éstas además de ser compuestas, tengan al menos un atributo en común.

5.16. Forma Normal de Boyce-Codd

- **FORMA NORMAL de Boyce-Codd:** Una relación R está en FNBC cuando para toda **DF no trivial X→A**:
  - **X es clave o superclave**

- ***Ejemplo:***

  ASISTE (cod_curso,nom_curso, cod_estudiante,nota)

  L= {cod_curso, cod_estudiante→nota, nom_curso→cod_curso, cod_curso→nom_curso}

  K_{ASISTE}= {(cod_curso,cod_estudiante), (nom_curso, cod_estudiante)}

  ¿está en 3FN?

  ¿está en FNBC?

5.17. Algoritmo de descomposición de FNBC con la propiedad LJ

- Dada una relación R en 3FN (con preservación de dependencias y join sin pérdida) se puede descomponer en un conjunto de esquemas en FNBC que cumpla con la propiedad de JOIN SIN PÉRDIDA.

- Sin embargo, **NO SIEMPRE** es posible descomponer una relación en FNBC que conserve las DF's.

5.17. Algoritmo de descomposición de FNBC con la propiedad LJ

- Antes de proceder con el algoritmo, es necesario tener en cuenta la siguiente propiedad:
  - Si $\rho = \{R_1, R_2, \ldots, R_k\}$ es una descomposición de R con la propiedad LJ respecto de L, y si $\sigma = \{S_1, S_2\}$ es una descomposición de R_1 con la propiedad LJ respecto de L_1, entonces $\rho = \{\mathbf{S_1, S_2}, R_2, \ldots, R_k\}$ verifica la propiedad LJ respecto de L.

- Además de la propiedad anterior, se conoce el siguiente teorema ya visto:
  - **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^+ , *es decir* es necesario que **el atributo o atributos comunes** de las dos relaciones **sea(n) clave en alguna de ellas**

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

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

5.17. Algoritmo de descomposición de FNBC con la propiedad LJ

- Algoritmo
  - **Entrada**: Un esquema de relación R(T, L)
  - **Salida**: Una descomposición ($\rho = \{R_1, R_2, \ldots, R_k\}$) de R con la propiedad LJ, de tal forma que cada R_i está en FNBC con respecto a L_i.
  - **Proceso**:
    1. Si $X \rightarrow A$ es una DF de **R** que viola la FNBC, es decir, X no es superclave $\Rightarrow$ descomponer la relación R en:

       R1 tal que **T1** = {$\underline{X}$, A}

       R2 tal que **T2** = T-{A}
    2. Si **R2** no está en FNBC aplicar recursivamente el paso 1

5.17. Algoritmo de descomposición de FNBC con la propiedad LJ

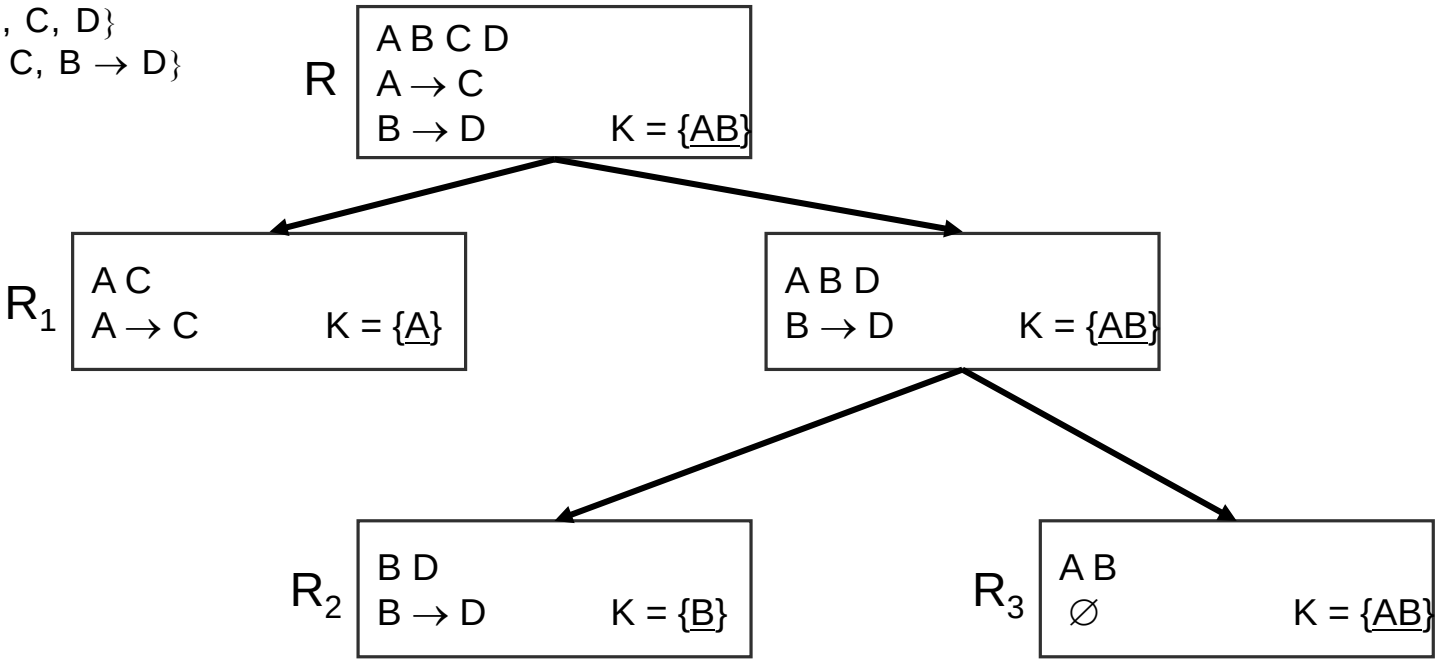
Algoritmo:

1.- Si $X \rightarrow A$ es una DF de R que viola la FNBC $\Rightarrow$ descomponer la relación R en:

$T_1=\{\underline{X}, A\}$ $T_2= R-\{A\}$

2.- Si R_2 no está en FNBC aplicar recursivamente 1

- ***Ejemplo***: Normalizar **R**(T, L) a FNBC teniendo en cuenta que:
  - **T** = {A, B, C, D}
    L = {A → C, B → D}
    K = {<u>AB</u>}



  - La descomposición ρ={R1, R3, R4} verifica la prop. LJ estando los cuatro esquemas en FNBC

5.17. Algoritmo de descomposición de FNBC con la propiedad LJ

Algoritmo:

1.- Si $X \rightarrow A$ es una DF de R que viola la FNBC $\Rightarrow$ descomponer la relación R en:

R_1={X, A} R_2= R-{A}

2.- Si R_2 no está en FNBC aplicar recursivamente 1

- Ejemplo: Normalizar R(T, L) a FNBC teniendo en cuenta que:
  - T = {A,B,C,D,E,F}
    L = {C→E, AF→C, AE→F, CD→B, AD→F}
    K= {AD}

5.17. Algoritmo de descomposición de FNBC con la propiedad LJ

Algoritmo:

1.- Si $X \rightarrow A$ es una DF de R que viola la FNBC $\Rightarrow$ descomponer la relación R en:

R_1={X, A} R_2= R-{A}

2.- Si R_2 no está en FNBC aplicar recursivamente 1

- Ejemplo: Normalizar R(T, L) a FNBC teniendo en cuenta que:
  - T = {A, B, C, D, E, F, G}
    L = {AB $\rightarrow$ C, C $\rightarrow$ D, DE $\rightarrow$ F, G $\rightarrow$ A, AE $\rightarrow$ B}
    K= {GE}

5.17. Algoritmo de descomposición de FNBC con la propiedad LJ

Algoritmo:

1.- Si $X \rightarrow A$ es una DF de R que viola la FNBC $\Rightarrow$ descomponer la relación R en:

R_1={X, A} R_2= R-{A}

2.- Si R_2 no está en FNBC aplicar recursivamente 1

- Ejemplo: Normalizar R(T, L) a FNBC teniendo en cuenta que:
  - T = {A, B, C, D, E, F}
    L = {$A \rightarrow F$, $CD \rightarrow A$, $F \rightarrow B$, $AB \rightarrow E$, $CB \rightarrow D$}
    K= {CA, CB, CD, CF}