Normalización

Tecnólogo en Informática, sede Paysandú
Bases de Datos 1

Normalización
Temario

- **Introducción**
- **Conceptos relacionados**
- **Formas Normales**
  - *1NF*
  - *2NF*
  - *3NF*
  - *BCNF*
- **Algoritmos de diseño**

- Referencias:
  - *Fundamentals of Database Systems* [E-N], 5ta. Edición, Caps. 10, 11

Normalización
Introducción

- En el **proceso** de normalización se somete un esquema relación (ER) a una serie de pruebas para "*certificar*" si pertenece o no a una cierta forma normal.

- Puede considerarse como un proceso durante el cual los ER insatisfactorios se descomponen repartiendo sus atributos entre ERs más pequeños que poseen **propiedades deseables**.

Normalización
Introducción

- Las formas normales, sin considerar otros factores, no garantizan un buen diseño de BD.

- Propiedades adicionales:
  - ***Join sin pérdida***: no se presentará el problema de las tuplas falsas
  - ***Preservación de dependencias***: todas las dfs están representadas en alguna relación

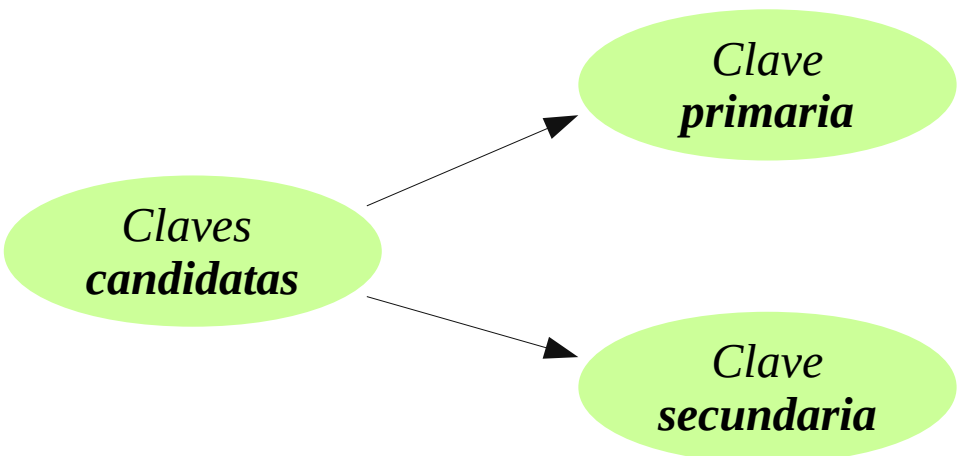
Normalización
Superclave y clave

- **Superclave**
  - Una superclave de $R=\{A1, \ldots, An\}$ es un conjunto de atributos $S \subseteq R$ tal que no existen dos tuplas *t1* y *t2* en ningún *r* tal que $t1[S]=t2[S]$.

- **Clave**
  - Una clave *K* es una superclave que cumple que, si se le quita alguno de sus atributos de *K*, deja de ser superclave.

Normalización
Clave candidata

- ***Clave candidata, clave primaria***
  - Si una relación tiene más de una clave, cada una es una *clave candidata*. Una de ellas es arbitrariamente designada como *clave primaria*. El resto son *secundarias*.



Normalización
Formas Normales

- Primera Forma Normal (*1NF*)
- Segunda Forma Normal (*2NF*)
- Tercera Forma Normal (*3NF*)
- Forma Normal Boyce-Codd (*BCNF*)

Normalización
Primera Forma Normal (1NF)

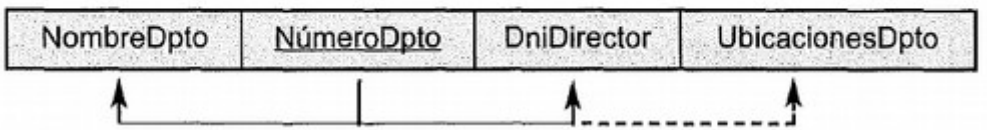
- **Definición**
  - Los dominios de los atributos deben incluir **solo valores atómicos** (los atributos no pueden ser multivaluados ni compuestos)

Normalización
Primera Forma Normal (1NF)

Figura 10.8. Normalización en 1FN. (a) Un esquema de relación que no está en 1FN. (b) Ejemplo de un estado de relación DEPARTAMENTO. (c) Versión 1FN de la misma relación con redundancia.

(a)

DEPARTAMENTO



(b)

DEPARTAMENTO

NombreDpto	NúmeroDpto	DniDirector	UbicacionesDpto
Investigación	5	333445555	{Valencia, Sevilla, Madrid}
Administración	4	987654321	{Gijón}
Sede central	1	888665555	{Madrid}

(c)

DEPARTAMENTO

NombreDpto	NúmeroDpto	DniDirector	UbicaciónDpto
Investigación	5	333445555	Valencia
Investigación	5	333445555	Sevilla
Investigación	5	333445555	Madrid
Administración	4	987654321	Gijón
Sede central	1	888665555	Madrid

Definiciones

- **Atributo Primo**
  - Un atributo del ER *R* es primo **si es miembro** *de alguna clave* de *R*.

Normalización
Definiciones

- **Dependencia total**
  - $X \rightarrow Y$ es una df total si la eliminación de cualquier atributo A de X hace que la df *deje de ser válida* (no tiene atributos redundantes a la izq.)

- **Dependencia parcial**
  - $X \rightarrow Y$ es una df parcial si es posible eliminar un atributo A de X, y la df *sigue siendo válida*.

Normalización
Segunda Forma Normal (2NF)

- **Definición**
  - Un ER *R* está en *2NF* si ningún atributo **no primo** *A* de *R* depende **parcialmente** de cualquier clave de *R*.

Normalización
Segunda Forma Normal (2NF)

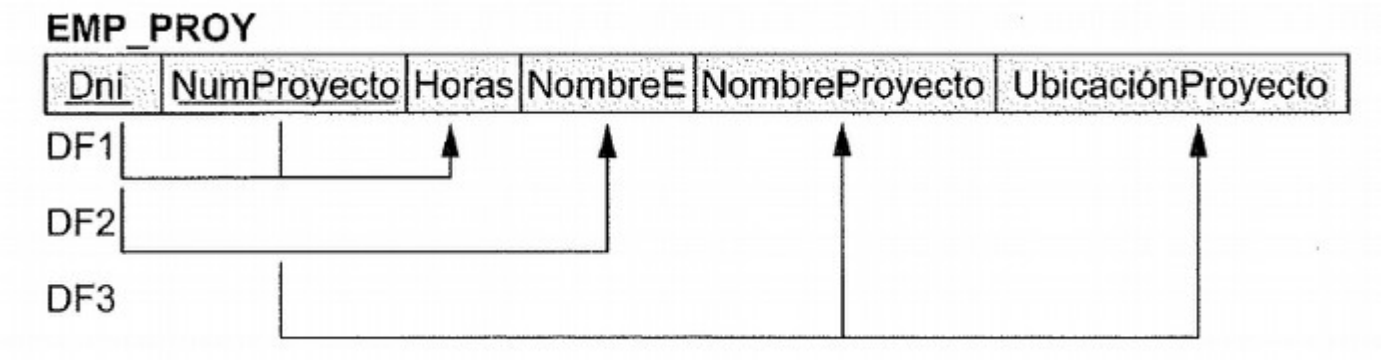
- **Ejemplo:**
  - *Empleado (<u>empleado, habilidad</u>, lugar_trabajo)*
  - $F = \{empleado, habilidad \rightarrow lugar\_trabajo,$
    $empleado \rightarrow lugar\_trabajo\}$

Normalización
Segunda Forma Normal (2NF)

- **Ejemplo:**
  - *Empleado (empleado, habilidad, lugar_trabajo)*
  - $F = \{empleado, habilidad \rightarrow lugar\_trabajo,$
    $empleado \rightarrow lugar\_trabajo\}$

*lugar_trabajo es **no primo** y **parcialmente** dependiente de una clave de Empleado*

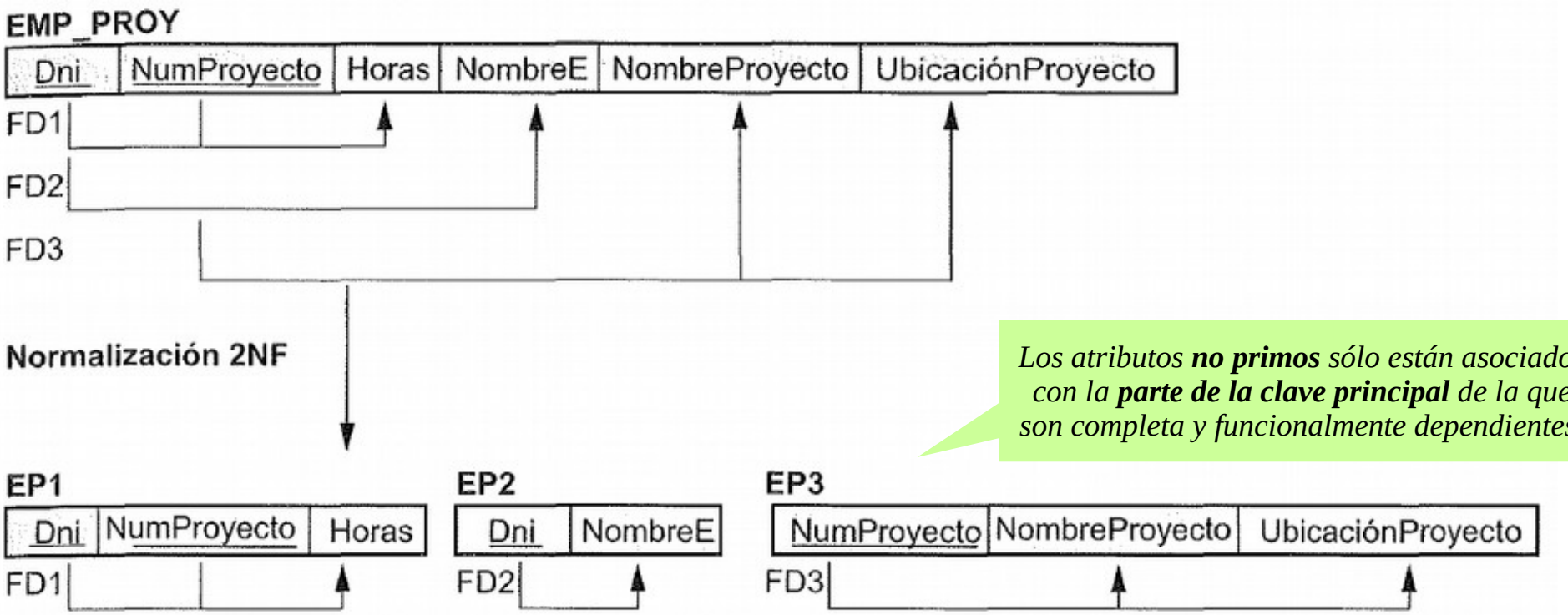
Normalización
Segunda Forma Normal (2NF)



- El atributo **no primo** *NombreE* viola *2NF*, por la DF2
- Los atributos **no primos** *NombreProyecto* y *UbicaciónProyecto* violan *2NF*, por la DF3

*Son **parcialmente** dependientes de una clave de EMP_PROY*

Normalización
Segunda Forma Normal (2NF)



*Los atributos **no primos** sólo están asociados con la **parte de la clave principal** de la que son completa y funcionalmente dependientes.*

Normalización
Definiciones

- **Dependencia transitiva**
  - $X \rightarrow Y$ en un ER R es una df transitiva si existe un conjunto de atributos Z que no sea un subconjunto de una clave de R, y se cumplen tanto $X \rightarrow Z$ como $Z \rightarrow Y$.

Tercera Forma Normal (3NF)

- **Definición:**
  - Un ER *R* está en *3NF* si está en *2NF* y ningún atributo **no primo** de *R* depende **transitivamente** de una clave de *R*.

  - Un ER *R* está en *3NF* si, siempre que una df $X \rightarrow A$ se cumple en *R*, o bien:

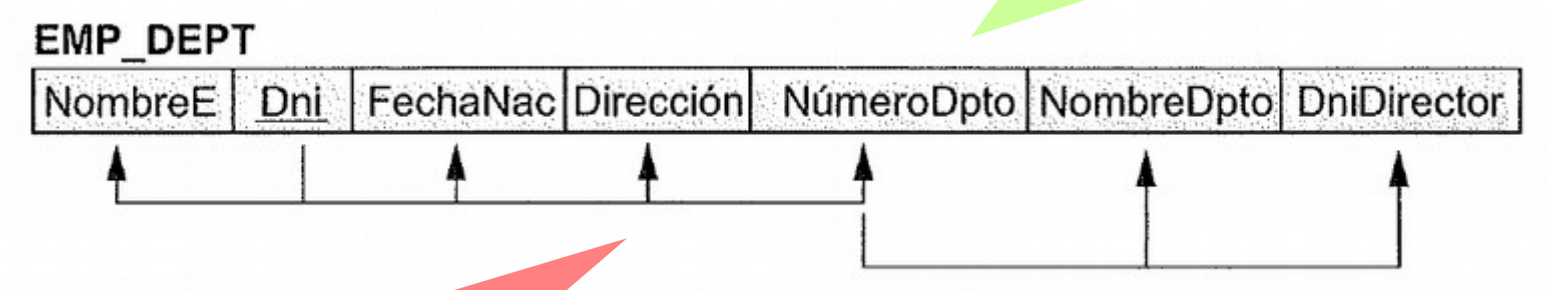
    a) ***X*** es una **superclave** de *R*, o

    b) ***A*** es un **atributo primo** de *R*.

Normalización
Tercera Forma Normal (3NF)

- Según la primer definición:

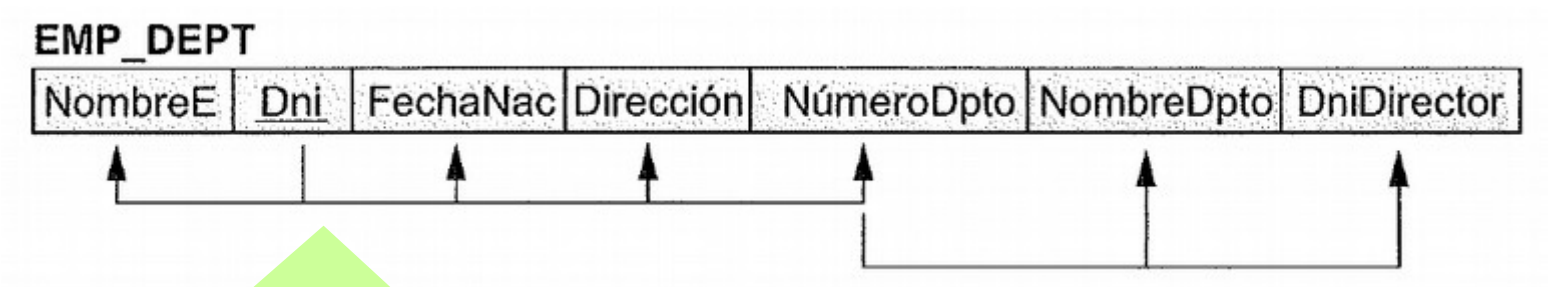
Está en *2NF*:
ya que no existen dependencias parciales en una clave



No está en *3NF:*
debido a la dependencia transitiva de *DniDirector* (y también de *NombreDpto*) en *Dni* a través de *NúmeroDpto*

Tercera Forma Normal (3NF)

- Según la segunda definición:

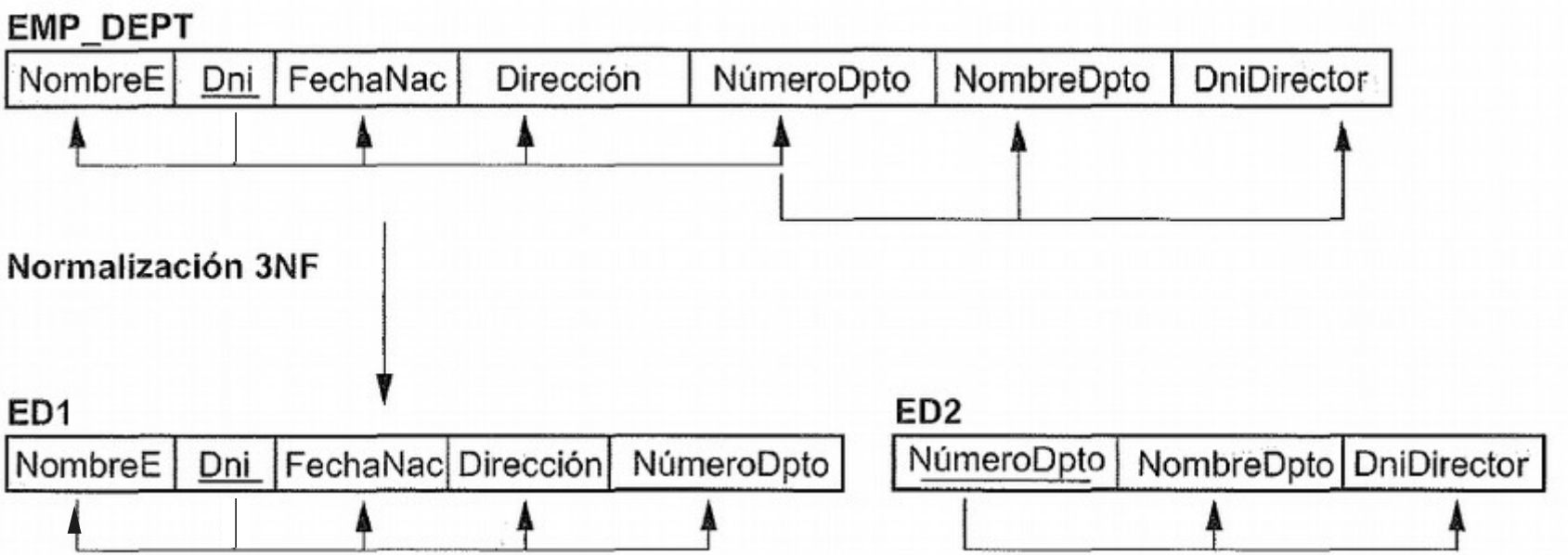


DF1 está en 3*NF*:
El atributo a la izquierda de la dependencia (*Dni*) es superclave

DF2 no está en *3NF:*
Los atributos a la derecha de la dependencia no son primos, ni tampoco *NúmeroDpto* es superclave en *EMP_DEPT*

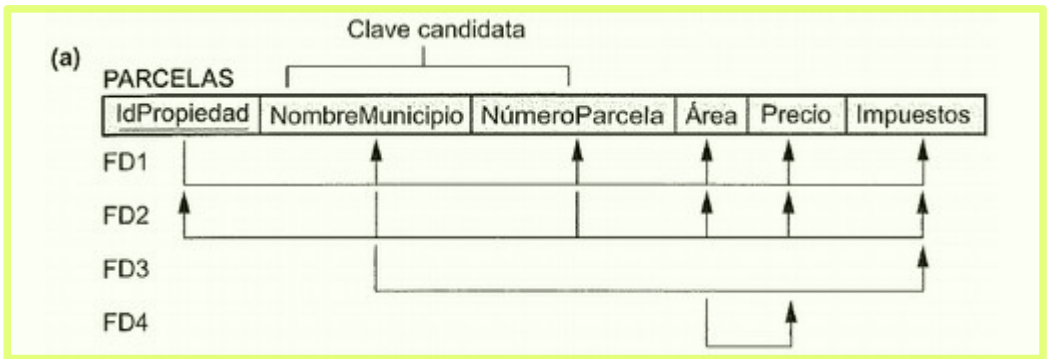
Normalización

Tercera Forma Normal (3NF)



Normalización

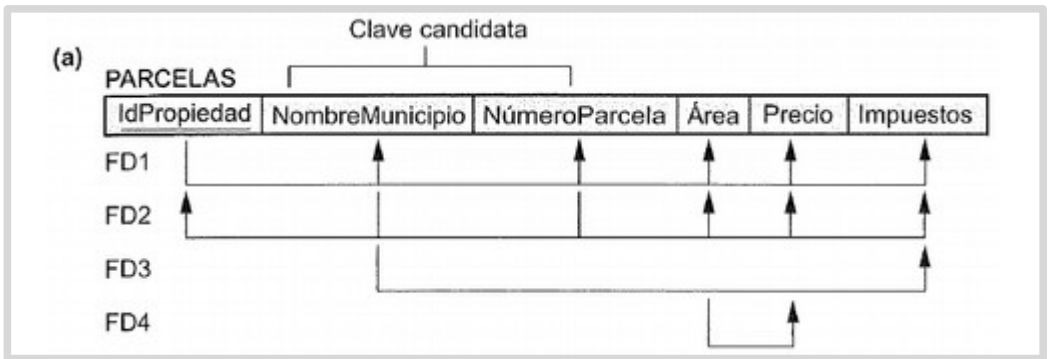
Tercera Forma Normal (3NF)



1NF

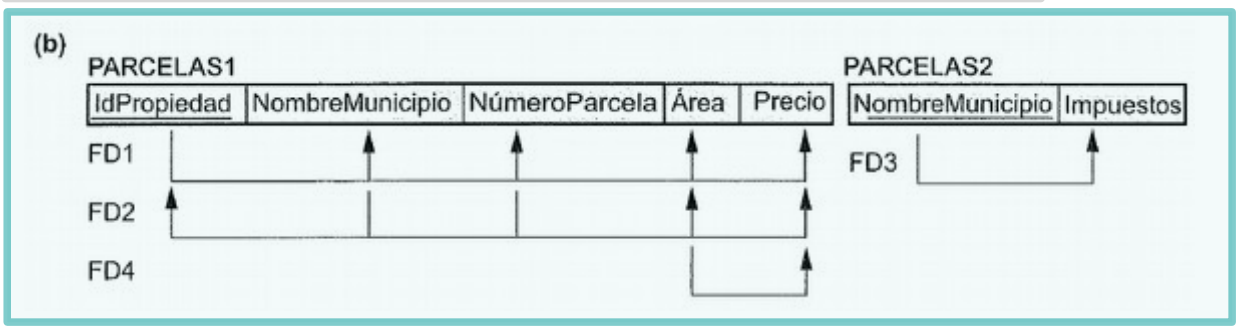
(d) PARCELAS 1NF

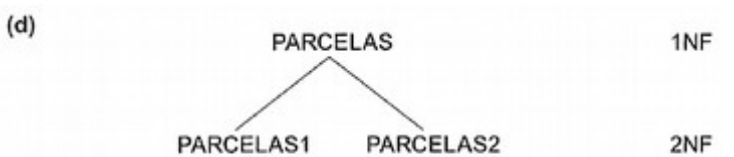
Normalización
Tercera Forma Normal (3NF)



1NF

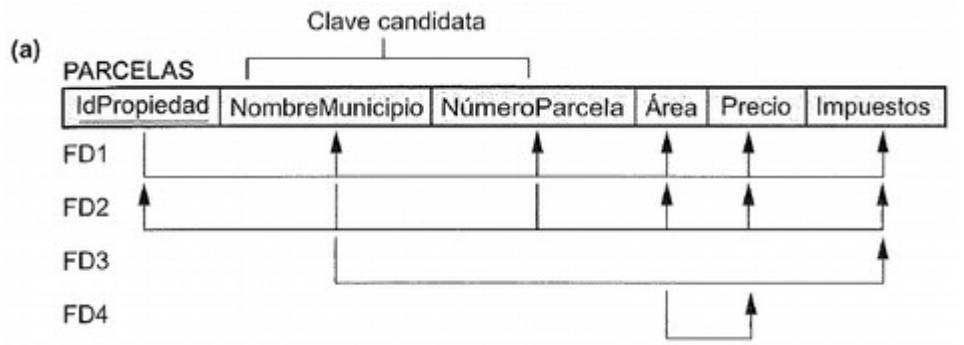
2NF



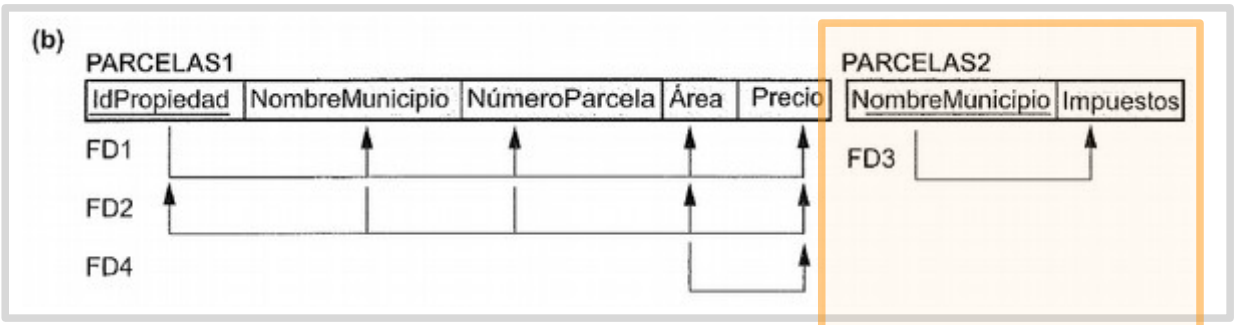


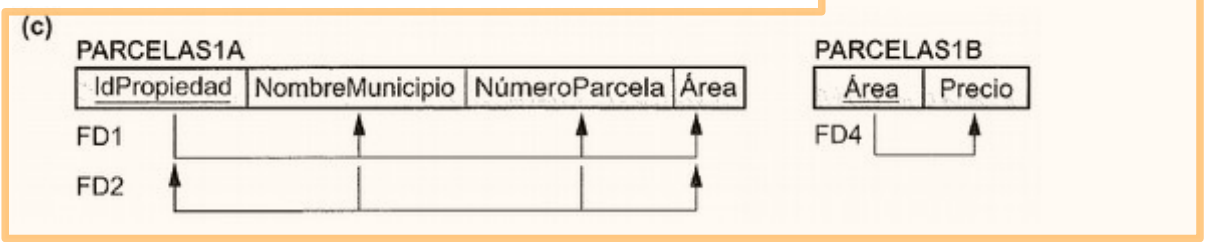
Normalización

Tercera Forma Normal (3NF)

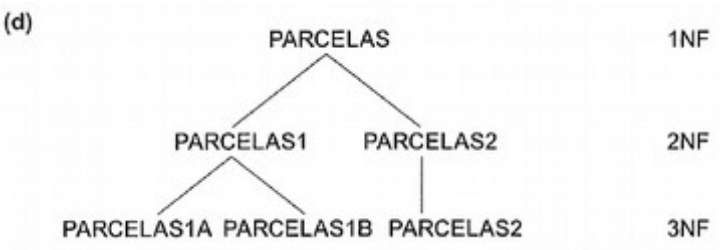


2NF





3NF



Forma Normal Boyce-Codd (BCNF)

- **Definición:**
  - Un ER *R* está en *BCNF* si, siempre que una df $X \rightarrow A$ se cumple en *R*, entonces *X* es una **superclave** de *R*.
  - Observar que, a diferencia de *3NF*, en *BCNF* no importa si *A* es un atributo primo (condición *b*).

Forma Normal Boyce-Codd (BCNF)

- **Ejemplo:**
  - Sea *R(nombre, teléfono, afición, dirección)*
  - $F=\{nombre, teléfono, afición \rightarrow dirección$

    $dirección \rightarrow teléfono\}$

Forma Normal Boyce-Codd (BCNF)

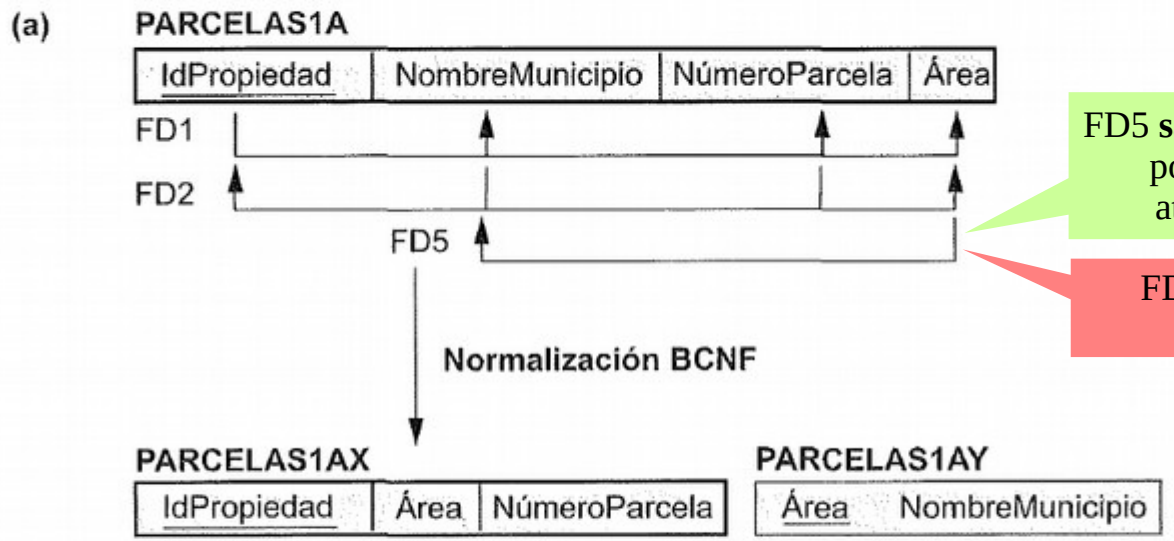
- **Ejemplo:**
  - Sea *R(<u>nombre, teléfono, afición</u>, dirección)*
  - $F = \{nombre, teléfono, afición \rightarrow dirección$

    $dirección \rightarrow teléfono\}$

Viola *BCNF* porque *dirección* no es superclave en *R*

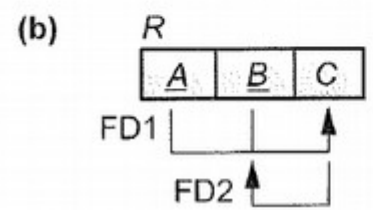
Normalización
Forma Normal Boyce-Codd (BCNF)

Figura 10.12. Forma normal de Boyce-Codd. (a) Normalización BCNF de PARCELAS1A en la que se pierde la dependencia funcional FD2 en la descomposición. (b) Una relación esquemática con FDs; está en 3FN pero no en BCNF.

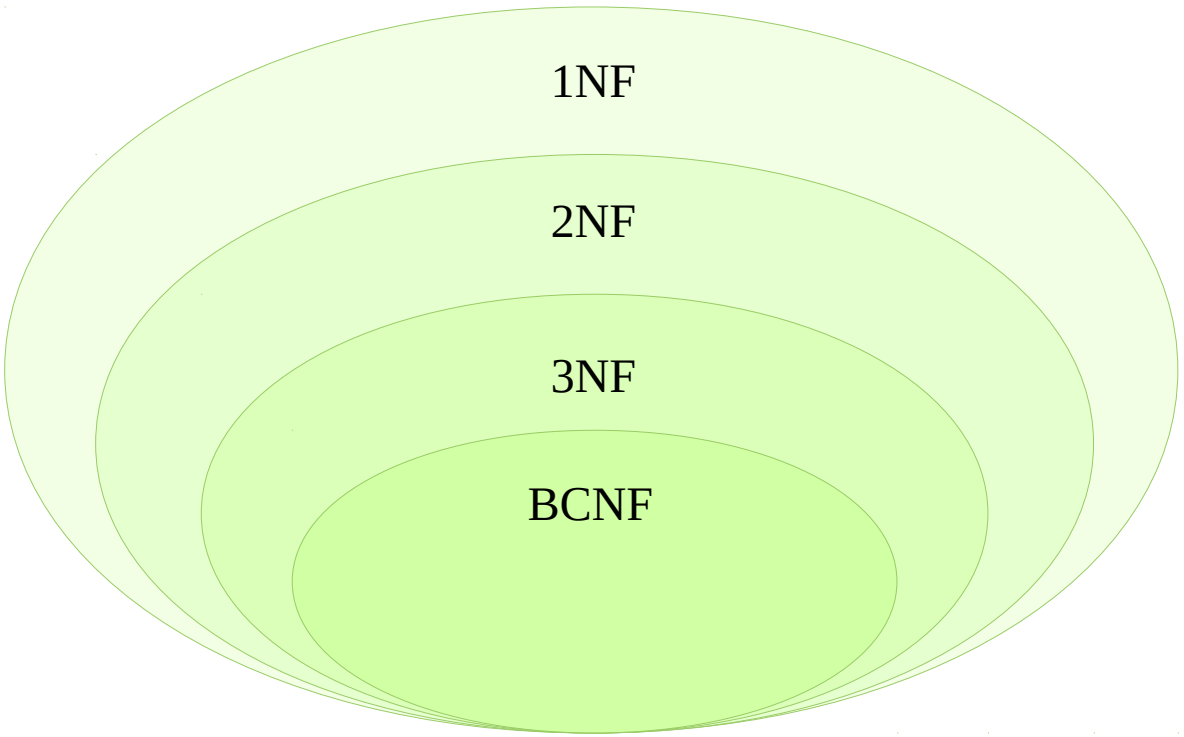


FD5 **satisface *3NF*** en *PARCELAS1A* porque *NombreMunicipio* es atributo primo (condición b)

FD5 **viola *BCNF*** porque *Área* no es superclave



Normalización
Formas Normales



Algoritmos de diseño

Normalización
Algoritmos de diseño

- Descomposición de relaciones
- Preservación de dependencias
- Descomposición en *3NF* con preservación de dfs
- *Join* sin pérdida (*JSP*)
  - Propiedad
  - Test de *join* sin pérdida
- Descomposición en *BCNF* con *JSP*
- Descomposición en *3NF* con *JSP* y preservación de dfs
- Problemas con valores nulos y tuplas colgantes

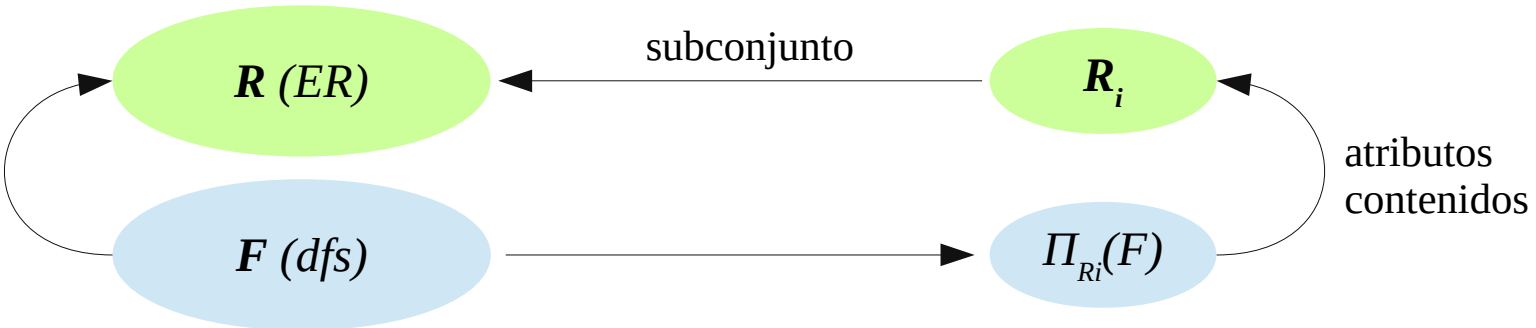
Descomposición de relaciones

- **Esquema relación universal *R***
  - *R=(A1, A2, ..., An)*, que contiene **todos** los atributos de la BD.

- **Descomposición de *R* (*D*)**
  - *D=(R1, R2, ..., Rm)*, que se obtiene mediante los algoritmos que realizan la descomposición utilizando las dependencias funcionales.
  - Se debe verificar: $\cup_{i=1..m}$ ***Ri* = *R***

Preservación de dependencias

- **Proyección de un conjunto de dfs sobre un Esquema de Relación**
  - Dado un conjunto de dfs F sobre R, la proyección de F sobre Ri, $\Pi_{Ri}(F)$, donde Ri es un subconjunto de R, es el conjunto de dfs $X \rightarrow Y$ en F^+ tal que los atributos en $X \rightarrow Y$ estén **todos contenidos** en Ri.

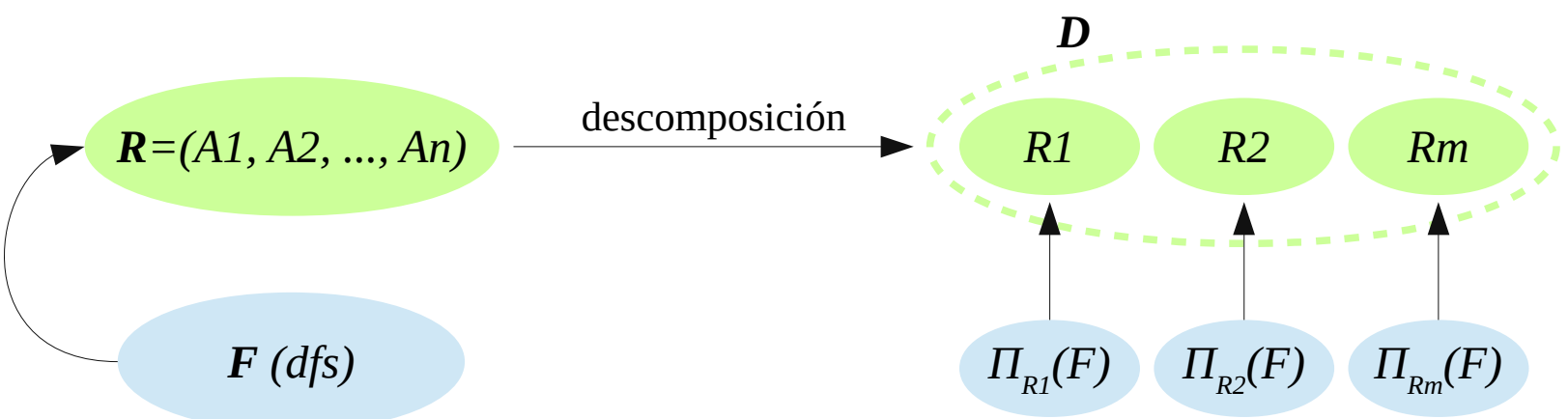


Preservación de dependencias

- **Preservación de dependencias**
  - Una descomposición *D=(R1, R2, ..., Rm)* de *R* preserva las dependencias respecto a *F* si se cumple:

$$((\Pi_{R1}(F)) \cup \ldots \cup (\Pi_{Rm}(F)))^{+} = F^{+}$$

  - La **unión** de las proyecciones de *F* en cada R_i en *D*, es **equivalente** a *F*



Descomposición en 3NF con pres. de dfs

- **Algoritmo:**

1. Encontrar un **cubrimiento minimal** *G* para *F*

2. Para cada miembro izq. *X* de una df que aparezca en *G*,

   crear un ER $\{X \cup A1 \cup A2 \cup ... \cup Am\}$ en *D*, donde $X \rightarrow A1$, $X \rightarrow A2$, ... , $X \rightarrow Am$ sean las únicas dfs en *G* con *X* como miembro izquierdo (*X* es la clave de esta relación)

3. Colocar cualquier atributo restante en **un solo ER** para asegurar la propiedad de *preservación de dependencias*

Descomposición en 3NF con pres. de dfs

- **Ejemplo:**
  - Se tiene la relación universal *U*:

    ***U** (DniEmpleado, NumProy, SalarioEmpleado, TlfEmpleado, NúmeroDpto, NombreProy, UbicaciónProy)*

  - Y las dependencias funcionales:

    ***DF1**: DniEmpleado → SalarioEmpleado, TlfEmpleado, NúmeroDpto*

    ***DF2**: NumProy → NombreProy, UbicaciónProy*

    ***DF3**: DniEmpleado, NumProy → SalarioEmpleado, TlfEmpleado, NúmeroDpto, NombreProy, Ubicación Proy*

Descomposición en 3NF con pres. de dfs

- **Paso 1:** cubrimiento minimal

  G: {DniEmpleado → SalarioEmpleado, TlfEmpleado, NúmeroDpto;

  NumProy → NombreProy, UbicaciónProy}

- **Paso 2:** crear ERs

  R1 (DniEmpleado, SalarioEmpleado, TlfEmpleado, NúmeroDpto)

  R2 (NumProy, NombreProy, UbicaciónProy)

Normalización
Descomposición en 3NF con pres. de dfs

- **Paso 1:** cubrimiento minimal

  G: {DniEmpleado → SalarioEmpleado, TlfEmpleado, NúmeroDpto;

  NumProy → NombreProy, UbicaciónProy}

- **Paso 2:** crear ERs

  R1 (DniEmpleado, SalarioEmpleado, TlfEmpleado, NúmeroDpto)

  R2 (NumProy, NombreProy, UbicaciónProy)

Atributos en común?
R1 ∩ R2 = Ø

Normalización
Join sin pérdida (JSP)

- **Definición:**
  - Una descomposición *D=(R1, R2, ..., Rm)* de *R* tiene la **propiedad de JSP** respecto al conjunto de dfs *F* sobre *R*, si por cada instancia de relación *r* de *R* que satisfaga *F*, **se cumple** lo siguiente:

    $$join\ (\ \Pi_{R1}(r), ..., \Pi_{Rm}(r)\ ) = r$$

  - JSP garantiza que *no se generarán* tuplas falsas cuando se aplica un *join* a las relaciones de la descomposición.

Normalización
Join sin pérdida (JSP)

- **Propiedad:** *Comprobación de JSP para descomposiciones binarias (dos ERs)*
  - *D = (R1, R2)* de *R* tiene JSP respecto a *F* sobre *R* si
    - la df ***(R1 ∩ R2) → (R1 - R2)*** está en F^+
    - **ó**, la df ***(R1 ∩ R2) → (R2 - R1)*** está en F^+

Normalización
Join sin pérdida (JSP)

- **Algoritmo:** *Comprobación de JSP para descomposiciones*

1. **Crear una matriz** *S* con una fila *i* por cada relación *Ri* en la descomposición *D*, y una columna *j* por cada atributo *Aj* en *R*;
2. **Hacer** $S(i,j) := b_{ij}$ para todas las entradas de la matriz;
3. **Para cada fila** *i* que represente el ER *Ri*
   para cada columna j que represente el atributo *Aj*
   si *Ri* incluye a *Aj* entonces hacer $S(i,j) := a_j$
4. **Repetir hasta** que una ejecución no modifique *S*
   para cada df $X \rightarrow Y$ en *F*
   igualar los símbolos en los atributos de *Y* para aquellas filas que coinciden en los atributos de *X*;
5. **Si una fila** tiene todos símbolos "*a*", la descomposición tiene JSP, en caso contrario, no lo es

Normalización
Join sin pérdida (JSP)

(a) R = {Dni, NombreE, NumProyecto, NombreProyecto, UbicaciónProyecto, Horas} $D = \{R_1, R_2\}$

R_1 = EMP_LOCS = {NombreE, UbicaciónProyecto}

R_2 = EMP_PROY1 = {Dni, NumProyecto, Horas, NombreProyecto, UbicaciónProyecto}

F = {Dni ➔ NombreE; NumProyecto ➔ {NombreProyecto, UbicaciónProyecto}; {Dni, NumProyecto} ➔ Horas}

	Dni	NombreE	NumProyecto	NombreProyecto	UbicaciónProyecto	Horas
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

(Ningún cambio en la matriz después de aplicar las dependencias funcionales.)

Join sin pérdida (JSP)

(a) R = {Dni, NombreE, NumProyecto, NombreProyecto, UbicaciónProyecto, Horas} $D = \{R_1, R_2\}$

R_1 = EMP_LOCS = {NombreE, UbicaciónProyecto}

R_2 = EMP_PROY1 = {Dni, NumProyecto, Horas, NombreProyecto, UbicaciónProyecto}

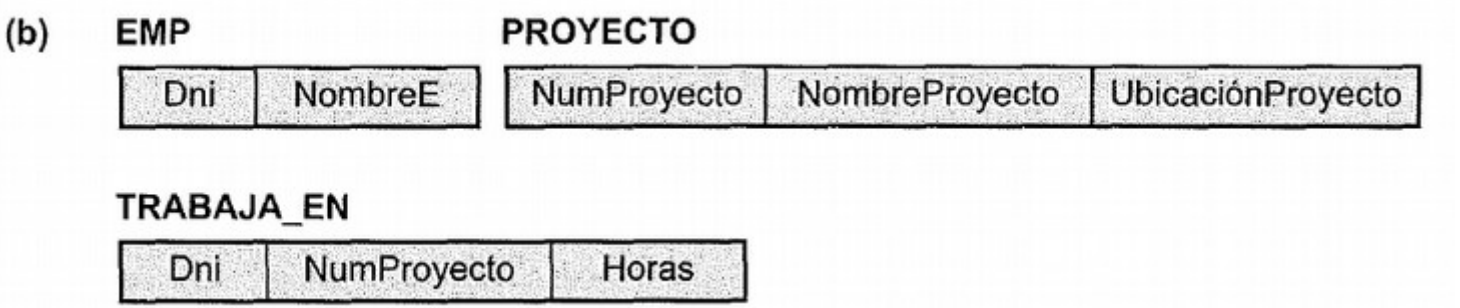
F = {Dni ➔ NombreE; NumProyecto ➔ {NombreProyecto, UbicaciónProyecto}; {Dni, NumProyecto} ➔ Horas}

	Dni	NombreE	NumProyecto	NombreProyecto	UbicaciónProyecto	Horas
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

(Ningún cambio en la matriz después de aplicar las dependencias funcionales.)

La descomposición **no tiene *JSP*** respecto a *F*

Normalización
Join sin pérdida (JSP)



Normalización
Join sin pérdida (JSP)

(c) R = {Dni, NombreE, NumProyecto, NombreProyecto, UbicaciónProyecto, Horas} $D = \{R_1, R_2, R_3\}$

R_1 = EMP = {Dni, NombreE}

R_2 = PROY = {NumProyecto, NombreProyecto, UbicaciónProyecto}

R_3 = TRABAJA_EN = {Dni, NumProyecto, Horas}

F ={Dni ➔ NombreE; NumProyecto ➔ {NombreProyecto, UbicaciónProyecto};{Dni, NumProyecto} ➔ Horas}

	Dni	NombreE	NumProyecto	NombreProyecto	UbicaciónProyecto	Horas
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(Matriz S original al comienzo del algoritmo.)

	Dni	NombreE	NumProyecto	NombreProyecto	UbicaciónProyecto	Horas
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	~~b_{22}~~ a_2	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

(Matriz S después de aplicar las dos primeras dependencias funcionales; la última fila sólo contiene símbolos "a", por lo que paramos)

Normalización
Join sin pérdida (JSP)

(c) R = {Dni, NombreE, NumProyecto, NombreProyecto, UbicaciónProyecto, Horas} $D = \{R_1, R_2, R_3\}$

R_1 = EMP = {Dni, NombreE}

R_2 = PROY = {NumProyecto, NombreProyecto, UbicaciónProyecto}

R_3 = TRABAJA_EN = {Dni, NumProyecto, Horas}

F ={Dni ➔ NombreE; NumProyecto ➔ {NombreProyecto, UbicaciónProyecto};{Dni, NumProyecto} ➔ Horas}

	Dni	NombreE	NumProyecto	NombreProyecto	UbicaciónProyecto	Horas
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(Matriz S original al comienzo del algoritmo.)

	Dni	NombreE	NumProyecto	NombreProyecto	UbicaciónProyecto	Horas
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	~~b_{22}~~ a_2	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

(Matriz S después de aplicar las dos primeras dependencias funcionales; la última fila sólo contiene símbolos "a", por lo que paramos)

La descomposición tiene JSP

Descomposición en BCNF con JSP

- **Algoritmo:**

1. Establecer $D:=\{ R \}$;

2. Mientras haya un ER Q en D que no esté en *BCNF* hacer

{

elegir un ER Q en D que no esté en *BCNF*;

localizar una df $X \rightarrow Y$ en Q que viole *BCNF*;

reemplazar Q en D con dos esquemas $(Q - Y)$ y $(X \cup Y)$;

};

Descomposición en BCNF con JSP

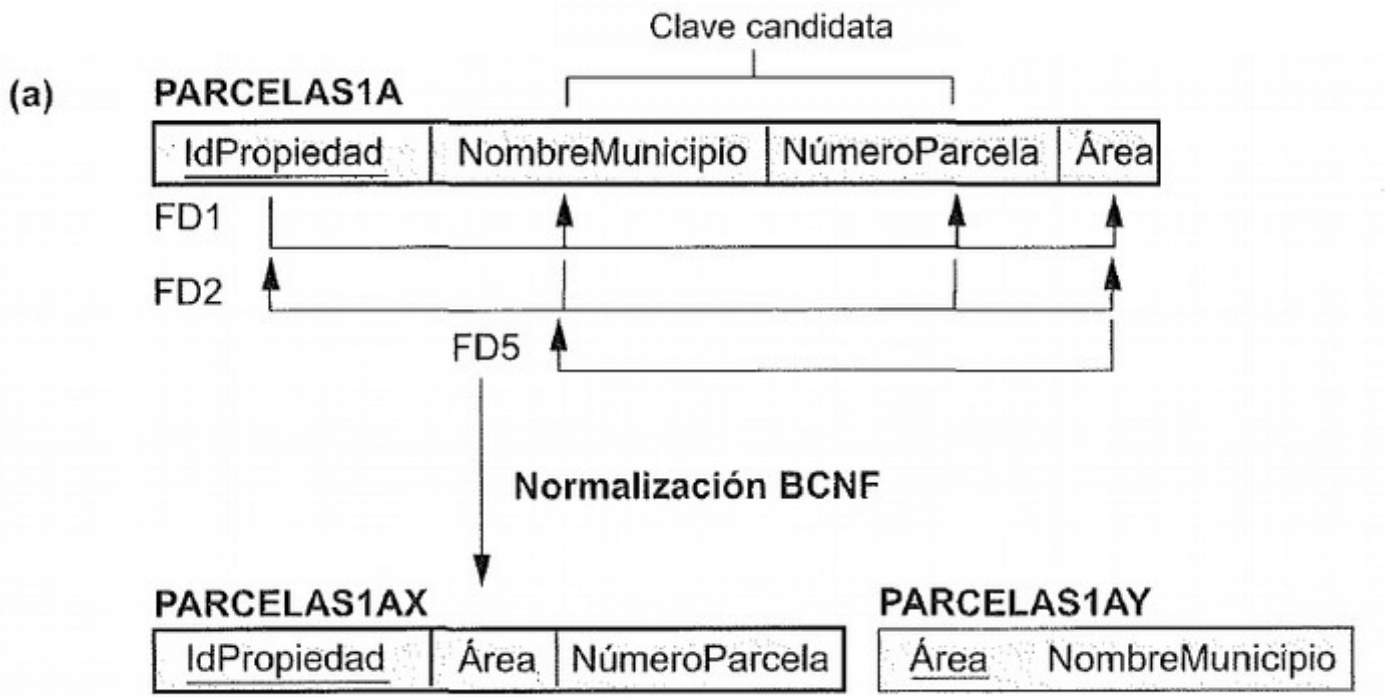
- **Ejemplo:**
  - Sea *R(nombre, teléfono, afición, dirección)*
  - *F={nombre, teléfono, afición → dirección*

    dirección → teléfono}

    Viola BCNF porque dirección no es superclave de R
  - *D = { R1(nombre, afición, dirección), R2(dirección, teléfono) }*

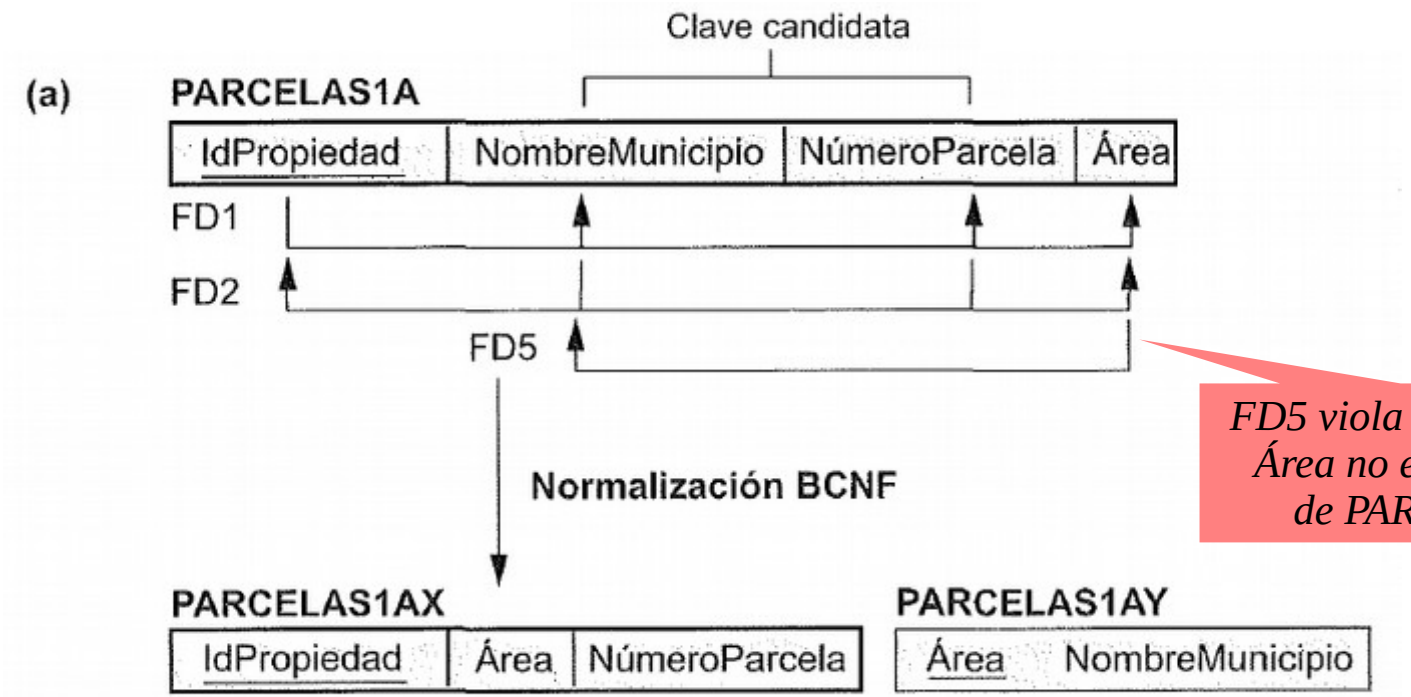
Normalización

Descomposición en BCNF con JSP



Normalización

Descomposición en BCNF con JSP



FD5 viola BCNF porque Área no es superclave de PARCELAS1A

Se pierde FD2

Descomposición en 3NF con JSP y pres. de dfs

- Si queremos que una descomposición tenga la propiedad de *join* sin pérdida y que conserve las dependencias, tenemos que **ajustar** los esquemas de relación **a *3FN*** en lugar de a *BCNF*

- Se modifica el algoritmo de *3NF* con pres. de dfs, para lograr una descomposición que cumpla:

  - Preservación de dependencias
  - Tener *join* sin pérdida (*JSP*)
  - Que cada ER resultante de la descomposición esté en *3NF*

Normalización
Descomposición en 3NF con JSP y pres. de dfs

- **Algoritmo:**

1. Encontrar un **cubrimiento minimal** *G* para *F*

2. Para cada miembro izq. *X* de una df que aparezca en *G*,

   crear un ER $\{X \cup A1 \cup A2 \cup ... \cup Am\}$ en *D*, donde $X \rightarrow A1$, $X \rightarrow A2$, ... , $X \rightarrow Am$ sean las únicas dfs en *G* con *X* como miembro izquierdo (*X* es la clave de esta relación)

3. Si ninguno de los esquemas de relación en *D* contienen una **clave de *R***, **crear un ER** más en *D* que **contenga** los atributos que forman una clave de *R*

4. **Eliminar** las relaciones **redundantes** del conjunto de relaciones resultante. Una relación *R* se considera como redundante si es una proyección de otra relación *S* en el esquema; alternativamente, *R* está abarcada por *S*.

Descomposición en 3NF con JSP y pres. de dfs

- **Ejemplo 1:**
  - ***U*** *(DniEmpleado, NumProy, SalarioEmpleado, TlfEmpleado, NúmeroDpto, NombreProy, UbicaciónProy)*
  - ***DF1****: DniEmpleado → SalarioEmpleado, TlfEmpleado, NúmeroDpto*
  - ***DF2****: NumProy → NombreProy, Ubicación Proy*
  - ***DF3****: DniEmpleado, NumProy → SalarioEmpleado, TlfEmpleado, NúmeroDpto, NombreProy, Ubicación Proy*

Normalización
Descomposición en 3NF con JSP y pres. de dfs

- ***U** (<u>DniEmpleado, NumProy</u>, SalarioEmpleado, TlfEmpleado, NúmeroDpto, NombreProy, UbicaciónProy)*

- **Paso 1:** cubrimiento minimal
  - *G: {DniEmpleado → SalarioEmpleado, TlfEmpleado, NúmeroDpto;*
    NumProy → NombreProy, UbicaciónProy}

- **Paso 2:** crear ERs
  - *R1 (<u>DniEmpleado</u>, SalarioEmpleado, TlfEmpleado, NúmeroDpto)*
  - *R2 (<u>NumProy</u>, NombreProy, Ubicación Proy)*

- **Paso 3:** crear ERs con clave
  - *R3 (<u>DniEmpleado</u>, NumProy)*

Normalización
Determinación de claves

- Se deben buscar 3 grupos de atributos:
  - **Los que aparecen *sólo* a la derecha** → *No* forman parte de ninguna clave
    - No me permiten determinar a ningún otro atributo
  - **Los que *no* aparecen a la derecha** → *Seguro* forman parte de todas las claves
    - No puedo determinar su valor a partir de otro atributo
    - Si un atributo no figura en las dfs, pero sí está en el ER, va en este conjunto
  - **Los que aparecen a la izquierda *y* derecha** → *Tal vez* formen parte de una clave
    - Debo verificar si forman parte o no

Normalización
Determinación de claves

- **Ejemplo**:
  - $R(A, B, C, D, E, G)$
  - $F = \{ AB \rightarrow C, D \rightarrow E, C \rightarrow D, EA \rightarrow B \}$

- Entonces:
  - Sólo a la der. = $\{ \}$ → no estarán en ninguna clave
  - Nunca a la der. = $\{A, G\}$ → estarán en todas las claves
  - A la izq. y der. = $\{B, C, D, E\}$ → tal vez estén en alguna clave

$\{AG\}+ = \{A, G\}$ → *no es clave*

$\{\mathbf{GAB}\}+ = \{G, A, B, C, D, E\}$ → *es clave*

$\{\mathbf{GAC}\}+ = \{G, A, C, D, E, B\}$ → *es clave*

$\{\mathbf{GAD}\}+ = \{G, A, D, E, B, C\}$ → *es clave*

$\{\mathbf{GAE}\}+ = \{G, A, E, B, C, D\}$ → *es clave*

Normalización
Verificar preservación de dfs

- **Algoritmo:** Para toda df $X \rightarrow Y$ en el conjunto de dfs hacer:

1. **Si todos** los atributos de df pertenecen a una de las relaciones R_i, esa dependencia se conserva **trivialmente**.
2. **Si no** pasa lo anterior, **verificar** si df se conserva mediante el siguiente método:

$Z := X$

Repetir

Para cada relación R_i:

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

Hasta que Z **no cambie** luego de una iteración **o** hasta que Z **incluya** a Y.

→ **Si Z incluye a Y, la dependencia se conserva.**

Normalización
Verificar preservación de dfs

- **Ejemplo:**
  - ***R1***$=(A, B, C, D)$, ***R2***$=(A, E, F)$
  - ***FD1***$: \{A \rightarrow BD, B \rightarrow CD, AC \rightarrow E\}$

La descomposición, preserva las dependencias?

- **Paso 1:** ver dfs triviales
  - $A \rightarrow BD$ y $B \rightarrow CD$ se preservan trivialmente en $R1$, ya sus los atributos están incluidos
- **Paso 2:** verifico las dfs restantes ($AC \rightarrow E$)
  - Para cada relación R_i: $Z := Z \cup ((Z \cap R_i)^+ \cap R_i)$

    Para $R1$:
    - $Z = AC \cup ((AC \cap ABCD)^+ \cap ABCD)$
    - $Z = AC \cup (AC^+ \cap ABCD)$
    - $Z = AC \cup (ACEBD \cap ABCD)$
    - $Z = AC \cup ABCD = ABCD$

Z no incluye a E

Normalización
Verificar preservación de dfs

- **Ejemplo:**
  - ***R1***$=(A, B, C, D)$, ***R2***$=(A, E, F)$
  - ***FD1***$: \{A \rightarrow BD, B \rightarrow CD, AC \rightarrow E\}$

- **Paso 1:** ver dfs triviales
  - $A \rightarrow BD$ y $B \rightarrow CD$ se preservan trivialmente en *R1*, ya sus los atributos están incluidos
- **Paso 2:** verifico las dfs restantes ($AC \rightarrow E$)
  - Para cada relación R_i: $Z := Z \ U \ ((Z \cap R_i)+ \cap R_i)$

Z en R1

Para *R1*:

- $Z = AC \cup ((AC \cap ABCD)^+ \cap ABCD)$
- $Z = AC \cup (AC^+ \cap ABCD)$
- $Z = AC \cup (ACEBD \cap ABCD)$
- $Z = AC \ U \ ABCD = ABCD$

Para *R2*:

- $Z = ABCD \cup ((ABCD \cap AEF)^+ \cap AEF)$
- $Z = ABCD \cup (A^+ \cap AEF)$
- $Z = ABCD \cup (ABDCE \cap AEF)$
- $Z = ABCD \cup AE = ABCD\mathbf{E}$

Z incluye a E → se preservan las dfs