DFyN (Diseño de BDs)

(Cap 10 - Elmasri 5ª edición y parte del Cap 11)

Diseño de BD

- Hasta ahora solo se ha utilizado el **sentido común** para diseñar BDs.
- Necesitamos algún tipo de **medida formal** que nos que nos indique porque el agrupamiento de atributos en el esquema de relación puede ser mejor o no.
- Para ello esta lo que se denomina "**Bondad**" de los esquemas de relación que se puede explicar a 2 niveles:
  - **Nivel lógico**: se refiere a la forma en que los usuarios interpretan el esquema y significado de los atributos y se aplica a esquemas de relaciones base y vistas (tablas virtuales, ie. una tabla que deriva de otras tablas, CREATE VIEW).
  - **Nivel de manipulación o almacenamiento**: se refiere a como se almacenan y se actualizan la tuplas de una relación. Se aplica a esquemas de relaciones base que son los que se almacenan físicamente como archivos.

Diseño de BD

- El **diseño** de una BD puede seguir dos metodologías:
  - **Ascendente** o diseño por sintesis (Bottom-up): parte de las relaciones básicas entre atributos individuales hacía el esquema de relación de la BD.
  - **Descendente** o diseño por análisis (Top-down): empieza con varios agrupamientos de atributos de una relación que están juntos de forma natural y luego viene la posible descomposición.
- La teoría descrita aquí se aplica a los dos tipos de diseño, pero en general a la descendente.
- En general vamos a ver:
  - Criterios para distinguir esquemas buenos de esquemas malos.
  - Dependencias Funcionales (DFs): son la principal herramienta para medir formalmente la idoneidad de las agrupaciones de atributos para formar esquemas de relación.
  - El uso de DFs para agrupar atributos en esquemas que estén en una determinada Forma Normal (FN).
  - Veremos que cuando un esquema está en su FN tiene ciertas características deseables.

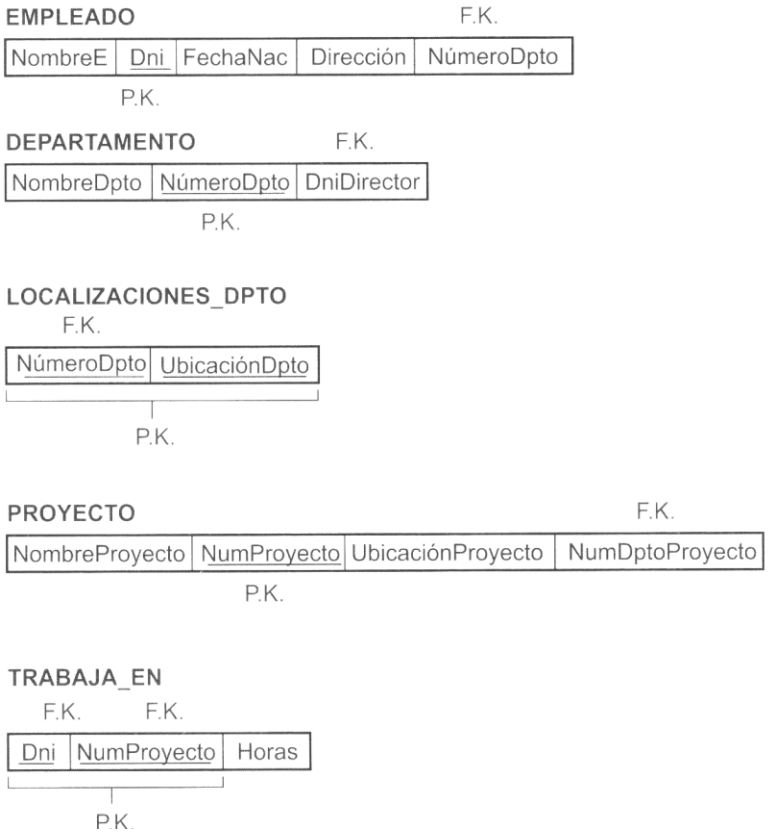
Medidas de Calidad Informales de BDs

- Semántica de atributos
- Reducción de valores redundantes en tuplas
- Reducción de valores nulos en tuplas
- Prohibición de tuplas espurias

Estas 4 medidas de calidad sobre el diseño de BDs no son independientes entre si.

Semántica de los Atributos de una Relación

- Especifica que relaciones hay entre los atributos de una tupla
- Cuando más fácil sea especificar la semántica de la tupla, más fácil será el diseño del esquema.
- Las recomendaciones son las siguientes:
  - El diseño del esquema se hará de modo que sea fácil explicar su significado.
  - Por regla general no se combinarán atributos de varios tipos de entidades en una sola relación (excepto los extrictamente necesarios, PKs)



- Este es un esquema fácil de explicar semánticamente

Semántica de los Atributos de una Relación

- Ejemplo del estado de la BD del esquema relacional anterior:

EMPLEADO

NombreE	Dni	FechaNac	Dirección	NúmeroDpto
Pérez Pérez, José	123456789	09-01-1965	Eloy I, 98	5
Campos Sastre, Alberto	333445555	08-12-1955	Avda. Ríos, 9	5
Jiménez Celaya, Alicia	999887777	19-07-1968	Gran Vía, 38	4
Sainz Oreja, Juana	987654321	20-06-1941	Cerquillas, 67	4
Ojeda Ordóñez, Fernando.	666884444	15-09-1962	Portillo, s/n	5
Oliva Avezuela, Aurora	453453453	31-07-1972	Antón, 6	5
Pajares Morera, Luis	987987987	29-03-1969	Enebros, 90	4
Ochoa Paredes, Eduardo	888665555	10-11-1937	Las Peñas, 1	1

DEPARTAMENTO

NombreDpto	NúmeroDpto	DniDirector
Investigación	5	333445555
Administración	4	987654321
Sede central	1	888665555

LOCALIZACIONES_DPTO

NúmeroDpto	UbicaciónDpto
1	Madrid
4	Gijón
5	Valencia
5	Sevilla
5	Madrid

PROYECTO

NombreProyecto	NumProyecto	UbicaciónProyecto	NumDptoProyecto
ProductoX	1	Valencia	5
ProductoY	2	Sevilla	5
ProductoZ	3	Madrid	5
Computación	10	Gijón	4
Reorganización	20	Madrid	1
Comunicaciones	30	Gijón	4

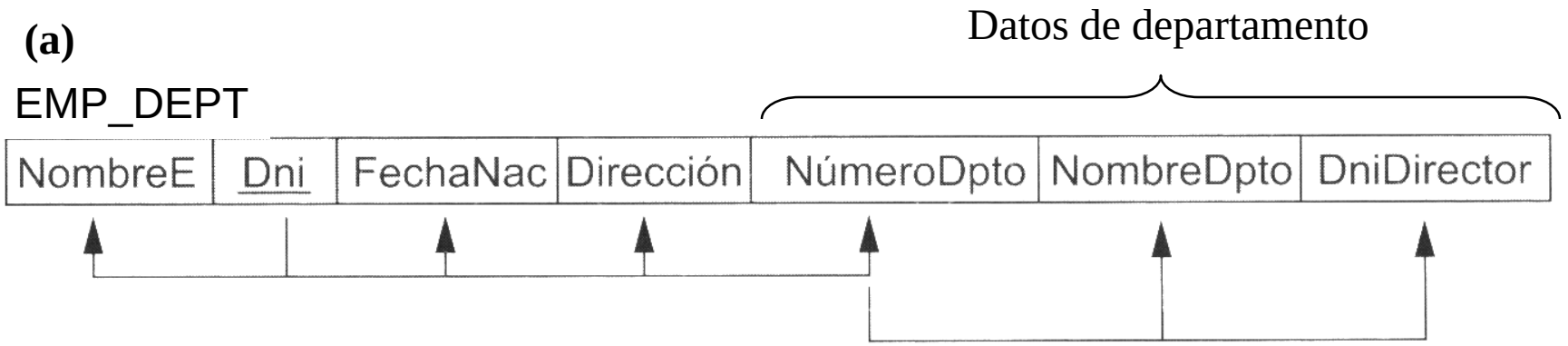
TRABAJA_EN

Dni	NumProyecto	Horas
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

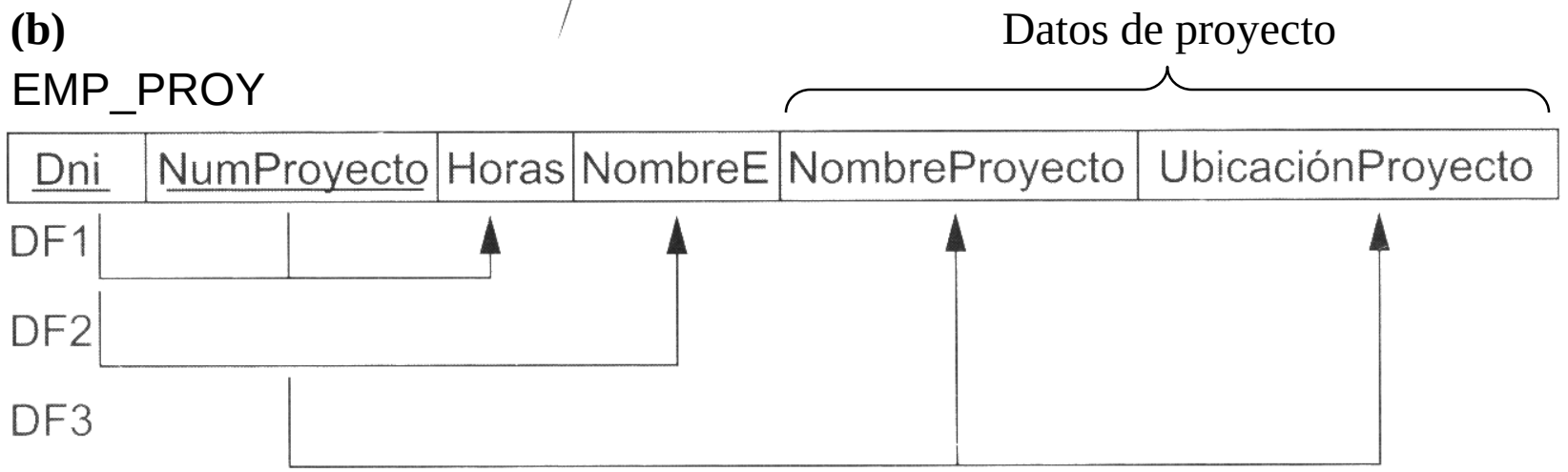
Semántica de los Atributos de una Relación

- En este caso, aunque aquí la semántica es buena, se mezclan atributos de diferentes entidades:
  - (a) Mezcla de atributos de EMPLEADO y DEPARTAMENTO
  - (b) Mezcla de atributos de EMPLEADO, PROYECTO y TRABAJA_EN (hemos quitado los atributos Dirección, FechaNac y NumeroDpto para que se pueda visualizar la tabla bien en la pantalla)

(a)



(b)



Información Redundante en Tuplas

- Uno de los objetivos de las BDs (su diseño) es minimizar el espacio de almacenamiento BDs que ocupan las relaciones base.
- La agrupación de atributos en relaciones tiene efecto significativo sobre el espacio de almacenamiento.
- Si aplicamos un NJ al estado de la BD que tenemos como ejemplo a las tablas EMPLEADO >< DEPARTAMENTO obtenemos una tabla EMP_DEPT
- Aparece una tabla con redundancia en las tuplas.

Redundancia

EMP_DEPT

NombreE	Dni	FechaNac	Dirección	NúmeroDpto	NombreDpto	DniDirector
Pérez Pérez, José	123456789	09-01-1965	Eloy I, 98	5	Investigación	333445555
Campos Sastre, Alberto	333445555	08-12-1955	Avda. Ríos, 9	5	Investigación	333445555
Jiménez Celaya, Alicia	999887777	19-07-1968	Gran Vía, 38	4	Administración	987654321
Sainz Oreja, Juana	987654321	20-06-1941	Cerquillas, 67	4	Administración	987654321
Ojeda Ordóñez, Fernando.	666884444	15-09-1962	Portillo, s/n	5	Investigación	333445555
Oliva Avezuela, Aurora	453453453	31-07-1972	Antón, 6	5	Investigación	333445555
Pajares Morera, Luis	987987987	29-03-1969	Enebros, 90	4	Administración	987654321
Ochoa Paredes, Eduardo	888665555	10-11-1937	Las Peñas, 1	1	Sede central	888665555

Información Redundante en Tuplas

- En teoría de información la **redundancia** es una propiedad de los mensajes, consiste en tener partes predictibles a partir del resto del mensaje y que por tanto en si mismo no aportan nueva información o repiten parte de la información ya existente.
- En BD's la redundancia hace referencia al almacenamiento de los mismos datos varias veces.
- La redundancia puede provocar varios problemas:
  - **Incremento de trabajo** (cuando se almacena o se borra un dato hay que almacenarlo o borrarlo de varios sitios).
  - **Derroche** de espacio de almacenamiento.
  - **Inconsistencia** (cuando los datos redundantes no coinciden por una modificación parcial de los mismos).
- La redundancia debe eliminarse en una BD, excepto la controlada (para seguridad y no perdida de datos, para mejorar el rendimiento respecto la consulta de BDs, etc.).

Información Redundante en Tuplas

- Si ahora combinamos información de EMPLEADO, PROYECTO y TRABAJA_EN obtenemos la tabla EMP_PROY.
- Aparece una tabla con redundancia en las tuplas.

Redundancia

Redundancia

EMP_PROY

Dni	NumProyecto	Horas	NombreE	NombreProyecto_	UbicaciónProyecto
123456789	1	32.5	Pérez Pérez, José	ProductoX	Valencia
123456789	2	7.5	Pérez Pérez, José	ProductoY	Sevilla
666884444	3	40.0	Ojeda Ordóñez, Fernando.	ProductoZ	Madrid
453453453	1	20.0	Oliva Avezuela, Aurora	ProductoX	Valencia
453453453	2	20.0	Oliva Avezuela, Aurora	ProductoY	Sevilla
333445555	2	10.0	Campos Sastre, Alberto	ProductoY	Sevilla
333445555	3	10.0	Campos Sastre, Alberto	ProductoZ	Madrid
333445555	10	10.0	Campos Sastre, Alberto	Computación	Gijón
333445555	20	10.0	Campos Sastre, Alberto	Reorganización	Madrid
999887777	30	30.0	Jiménez Celaya, Alicia	Comunicaciones	Gijón
999887777	10	10.0	Jiménez Celaya, Alicia	Computación	Gijón
987987987	10	35.0	Pajares Morera, Luís	Computación	Gijón
987987987	30	5.0	Pajares Morera, Luís	Comunicaciones	Gijón
987654321	30	20.0	Sainz Oreja, Juana	Comunicaciones	Gijón
987654321	20	15.0	Sainz Oreja, Juana	Reorganización	Madrid
888665555	20	Null	Ochoa Paredes, Eduardo	Reorganización	Madrid

Información Redundante en Tuplas

- De la tabla anterior hemos quitado los atributos Dirección, FechaNac y NumeroDpto para que se pueda visualizar la tabla bien en la pantalla.

Información Redundante en Tuplas: Anomalías de Actualización

- Obviamente uno de los problemas que surgen cuando se usan las relaciones anteriores formados por los NJs, como relaciones bases, son la **anomalías de actualización** que comentamos anteriormente:
  - **Inserción**:
    - para insertar un nuevo empleado en la tabla EMP-DEPT debemos asegurar que los datos del departamento al que pertenece son congruentes con los datos del departamento en otras tuplas.
    - Es difícil insertar un nuevo departamento que aún no tiene empleados (se puede hacer poniendo NULLs, pero esto tiene problemas (DNI es C-1º ya que cada tupla representa un empleado, además cuando se introduce el primer empleado del departamento hay que ponerlo en los NULLs)
  - **Eliminación**:
    - Si eliminamos la tupla correspondiente al último empelado de un departamento se pierde la información de ese departamento.
  - **Modificación** :
    - Si cambiamos el valor de uno de los atributos de un departamento, debemos actualizar todas la tuplas de todos los empleados que pertenecían a ese departamento. Por ejemplo investigación se cambia a I+D: hay que hacerlo en todas.

Información Redundante en Tuplas

- Si nos basamos en las tres anomalías anteriores, las recomendaciones serían:
  - Diseñar esquemas de forma que podamos evitar las anomalía anteriores.
  - Si hubiese anomalías indicarlo para que los programas de actualización del SGDB operen correctamente.
  - OJO: en ciertas ocasiones es preciso saltarse la recomendaciones para mejorar el rendimiento de ciertas consultas :
    - por ejemplo cuando se quiere consultar datos de departamento y empleado a la vez de manera muy intensa.
    - Pero para esto están las VISTAS.
  - **En general es altamente aconsejable utilizar relaciones base LIBRES de anomalías y especificar VISTAS que incluyan atributos de distintas relaciones.**

Valores Nulos en Tuplas

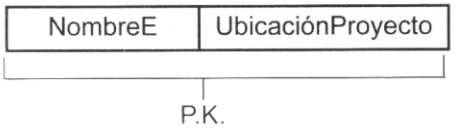
- Tenemos que ser conscientes que si agrupamos muchos atributos en una relación, puede que algunos de ellos que no apliquen a todas la tuplas de la relación (NULL).
- Los NULLs siempre tiene problemas:
  - Desperdicio de espacio de almacenamiento.
  - Como se manejan en funciones agregadas (operaciones relacionales), por ejemplo: COUNT, SUM, MAX, MIN, AVG
  - SELEC SUM(sueldo), MAX(sueldo), MIN(sueldo), AVG(sueldo) FROM EMPLEADO;
  - SELEC COUNT(*) FROM EMPLEADO;
- Además los nulos pueden tener múltiples interpretaciones:
  - El atributo no se aplica a esa tupla en cuestión.
  - Valor desconocidos del atributo para esa tupla.
  - Valor no registrado aunque se conoce.
- Por lo tanto tienen la misma representación pero el significado puede ser diferente.
- Se recomienda:
  - Evitar incluir atributos cuyos valores puedan ser nulos.
  - Si es imposible evitar nulos asegurarse que son una excepción en cierta tuplas.

Tuplas Espurias

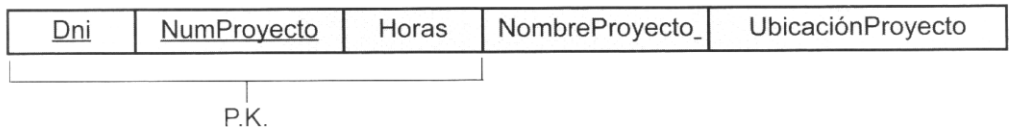
- Supongamos que empleamos EMP_LOCS y EMP_PROJ1 como relaciones base:

(a)

EMP_LOCS



EMP_PROJ1



(b)

EMP_LOCS

NombreE	Ubicación-Proyecto
Pérez Pérez, José	Valencia
Pérez Pérez, José	Surgarland
Ojeda Ordóñez, Fernando.	Madrid
Oliva Avezuela, Aurora	Valencia
Oliva Avezuela, Aurora	Surgarland
Campos Sastre, Alberto	Surgarland
Campos Sastre, Alberto	Madrid
Campos Sastre, Alberto	Gijón
Jiménez Celaya, Alicia	Gijón
Pajares Morera, Luis	Gijón
Sainz Oreja, Juana	Gijón
Sainz Oreja, Juana	Madrid
Ochoa Paredes, Eduardo	Madrid

EMP_PROJ1

Dni	NumProyecto	Horas	NombreProyecto	Ubicación-Proyecto
123456789	1	32.5	ProductoX	Valencia
123456789	2	7.5	ProductoY	Sevilla
666884444	3	40.0	ProductoZ	Madrid
453453453	1	20.0	ProductoX	Valencia
453453453	2	20.0	ProductoY	Sevilla
333445555	2	10.0	ProductoY	Sevilla
333445555	3	10.0	ProductoZ	Madrid
333445555	10	10.0	Computación	Gijón
333445555	20	10.0	Reorganización	Madrid
999887777	30	30.0	Comunicaciones	Gijón
999887777	10	10.0	Computación	Gijón
987987987	10	35.0	Computación	Gijón
987987987	30	5.0	Comunicaciones	Gijón
987654321	30	20.0	Comunicaciones	Gijón
987654321	20	15.0	Reorganización	Madrid
888665555	20	NULL	Reorganización	Madrid

- Esto nos hace no poder recuperar la información que había en la tabla EMPO_PROY desde esas 2 nuevas tablas .
- Para ello hacemos un NJ de la estas dos tablas (**a través de UbicacionProyecto**)

Tuplas Espurias

- Se obtienen tuplas erróneas (*) porque le NJ se hace sobre atributos que no son C-1ºs (UbicaciónProyecto no es ni PK ni FK).

	Dni	NumProyecto	Horas	NombreProyecto	UbicaciónProyecto	NombreE
	123456789	1	32.5	ProductoX	Valencia	Pérez Pérez, José
*	123456789	1	32.5	ProductoX	Valencia	Oliva Avezuela, Aurora
	123456789	2	7.5	ProductoY	Sevilla	Pérez Pérez, José
*	123456789	2	7.5	ProductoY	Sevilla	Oliva Avezuela, Aurora
*	123456789	2	7.5	ProductoY	Sevilla	Campos Sastre, Alberto
	666884444	3	40.0	ProductoZ	Madrid	Ojeda Ordóñez, Fernando.
*	666884444	3	40.0	ProductoZ	Madrid	Campos Sastre, Alberto
*	453453453	1	20.0	ProductoX	Valencia	Pérez Pérez, José
	453453453	1	20.0	ProductoX	Valencia	Oliva Avezuela, Aurora
*	453453453	2	20.0	ProductoY	Sevilla	Pérez Pérez, José
	453453453	2	20.0	ProductoY	Sevilla	Oliva Avezuela, Aurora
*	453453453	2	20.0	ProductoY	Sevilla	Campos Sastre, Alberto
*	333445555	2	10.0	ProductoY	Sevilla	Pérez Pérez, José
*	333445555	2	10.0	ProductoY	Sevilla	Oliva Avezuela, Aurora
	333445555	2	10.0	ProductoY	Sevilla	Campos Sastre, Alberto
*	333445555	3	10.0	ProductoZ	Madrid	Ojeda Ordóñez, Fernando.
	333445555	3	10.0	ProductoZ	Madrid	Campos Sastre, Alberto
	333445555	10	10.0	Computación	Gijón	Campos Sastre, Alberto
*	333445555	20	10.0	Reorganización	Madrid	Ojeda Ordóñez, Fernando.
	333445555	20	10.0	Reorganización	Madrid	Campos Sastre, Alberto

*

*

Tuplas Espurias

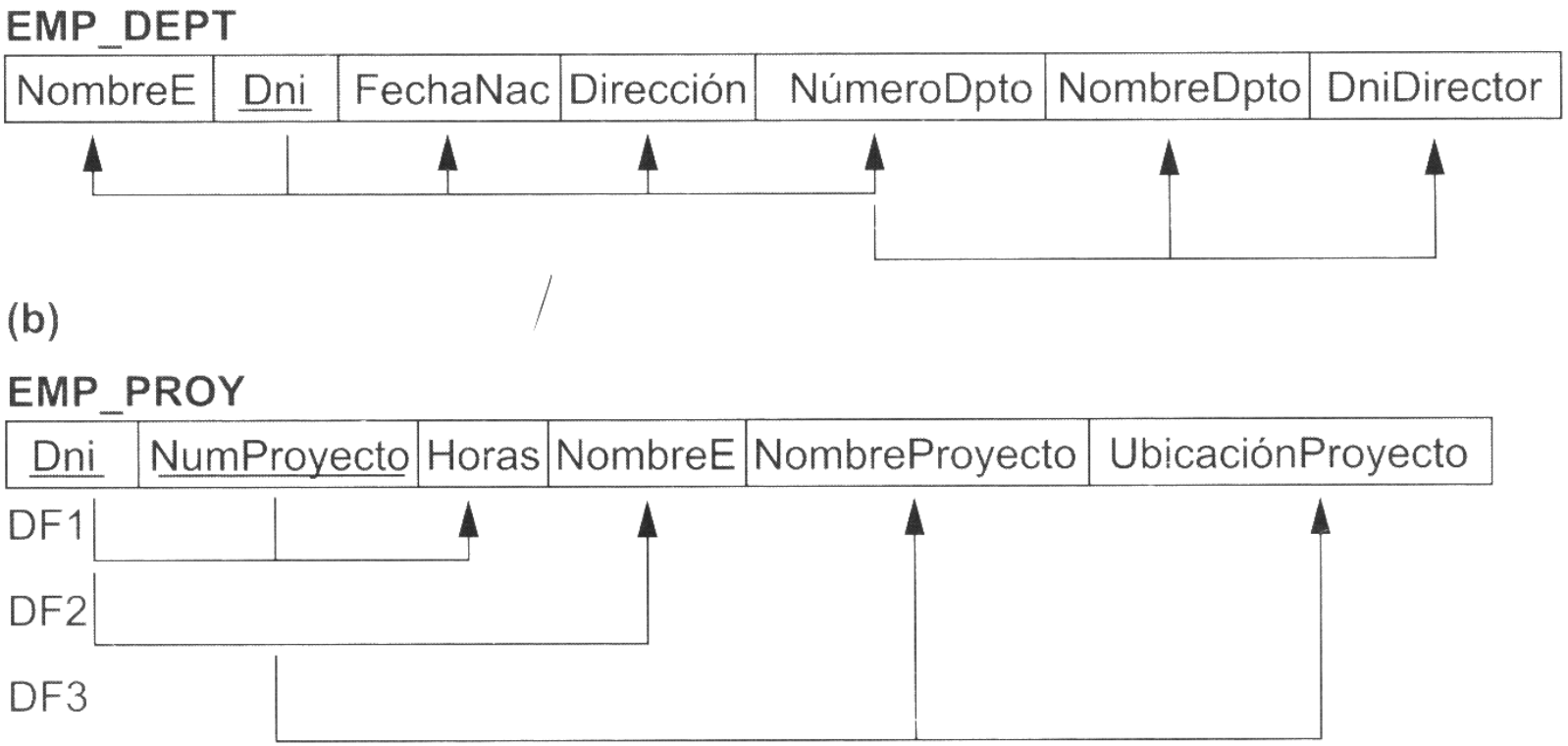
- **La recomendación es que se diseñen esquemas sobre los cuales se pueden hacer NJ sin producir tuplas espurias, es decir que los atributos del NJ sean PK y FK.**

Superclaves, Clave Candidata, Clave Principal, Atributos Primos

- Una **superclave** X de un esquema de la relación R={A_1,…,A_n} es un conjunto de atributos S $\subseteq$ R con la propiedad de que no habrá un par de tuplas t_1 y t_2 en ningún estado de la relación permitido r de R tal que t_1[X] = t_2[X].
- Un **clave** K es una superclave con la propiedad adicional de que la eliminación de cualquier atributo de K provocará que K deje de ser una superclave. Es la superclave **mínima**.
- Si un esquema de una relación tiene más de una clave cada una de ellas se llama **clave candidata**.
- Una de ellas se elige arbitrariamente como **clave principal**.
- El resto de las claves candidatas son **claves secundarias**.
- Cualquier atributo del esquema de la relación R que pertenece a una clave candidata o es una clave candidata se denomina **atributo primo** o primario.
- Un **atributo no primo** no es miembro de una clave candidata o no es una clave candidata.

Dependencias Funcionales

- Una dependencia funcional (DF) es una **restricción implícita**.
- Si X e Y son dos conjuntos de atributos, decimos que hay un DF de X a Y o que Y depende funcionalmente de X, si y solo si siempre que dos tuplas que coinciden en su valor X, necesariamente deben coincidir en su valor Y.
- Es decir, los atributos de Y están unívocamente determinados por los de X
- Notación: X → Y (dni → FechaNac, pero no FechaNac → dni)



Dependencias Funcionales

- De manera más estricta: Dados dos conjuntos X e Y de atributos de un esquema de relación R, supongamos dos tuplas t_1 y t_2 entonces el conjunto de atributos Y depende funcionalmente del conjunto de atributos X si $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y]$, $\forall\ t_1, t_2 \in r(R)$, donde $r(R)$ es un estado de la relación particular del esquema de relación.
- **Ojo que el tener la DF X → Y no supone tener Y → X en R.**
- En el ejemplo de la figura anterior las posibles DFs que tenemos son:
  - DF1: {Dni, NumProyecto}→ Horas
  - DF2: Dni → NombreE
  - DF3: NumProyecto → {NombreProyecto, UbicaciónProyecto}
- **En general si X es una superclave de R ⇒ X → Y para cualquier conjunto de atributos Y de R.**
- Observar que si X es una clave candidata de la relación entonces tenemos X → Y para cualquier agrupación de atributos Y (ejem: en EMP_DEPT existe la DF Dni → Número Dpto, pero no Número Dpto → Dni , ver en la tabla).
- **Realmente un DF la podríamos ver en la tabla como una restricción entre atributos.**

Dependencias Funcionales: Reglas de Inferencia

- Las reglas de inferencia sirven para deducir o inferir nuevas dependencias funcionales a partir de un conjunto dado de dependencias.

(RI1) Reflexividad: $Y \subset X$ $\models$ $X \rightarrow Y$ (dependencia trivial)

(RI2) Aumento: $\{ X \rightarrow Y \}$ $\models$ $XZ \rightarrow YZ$

(RI3) Transitividad: $\{ X \rightarrow Y, Y \rightarrow Z \}$ $\models$ $X \rightarrow Z$

(RI4) Proyección: $\{ X \rightarrow YZ \}$ $\models$ $X \rightarrow Y$ (descomposición)

(RI5) Aditividad: $\{ X \rightarrow Y, X \rightarrow Z \}$ $\models$ $X \rightarrow YZ$ (unión)

(RI6) Pseudotransitividad: $\{ X \rightarrow Y, WY \rightarrow Z \}$ $\models$ $WX \rightarrow Z$

Obsérvese que: $\{ X \rightarrow Y, Z \rightarrow W \}$ NO$\models$ $XZ \rightarrow YW$

$XY \rightarrow Z$ NO$\models$ $X \rightarrow Z$

- **F $\models$ X $\rightarrow$ Y quiere decir que la dependencia funcional X $\rightarrow$ Y se infiere o se deduce del conjunto de dependencias funcionales F.**

Dependencias Funcionales: Reglas de Inferencia

- RI1 especifica que un conjunto de atributos siempre se determina a si mismo o cualquiera de sus subconjuntos (obvio). Ya que RI1 genera dependencias que siempre son verdaderas estas se llaman triviales, las no triviales son el resto.
- RI2 o regla de aumento dice que añadir el mismo conjunto de atributos a cada lado de la dependencia genera otra dependencia válida.
- RI3 o regla transitiva dice que una dependencia funcional $X \rightarrow Z$ en un esquema de relación R es una dependencia transitiva si existe un conjunto de atributos Y que ni es clave candidata ni es un subconjunto de una clave de R, y se cumple tanto $X \rightarrow Y$ como $Y \rightarrow Z$.
- RI4 (regla de eliminación de atributos del lado derecho), si aplicamos esta regla repetidamente podemos descomponer la DF $\{X \rightarrow A_1, \ldots, A_n\}$ en el conjunto de DFs $\{X \rightarrow A_1, \ldots, X \rightarrow A_n\}$.
- RI5(regla de unión de atributos), nos permite realizar lo contrario para combinar DFs $\{X \rightarrow A_1, \ldots, X \rightarrow A_n\}$ en la DF $\{X \rightarrow A_1, \ldots, A_n\}$.
- Se puede comprobar que RI4, RI5 y RI6 se infieren usando las reglas RI1, RI2 y RI3 (reglas de inferencia de Amstrong). (ver el cap 10 del libro).

Dependencias Funcionales: Reglas de Inferencia de Amstrong

- **Clausura de F** (F^+): el conjunto de todas las DFs que incluyen F, junto con todas la DFs que se pueden inferir desde F es lo que se llama clausura de F y se designa por **F^+**.
- Amstrong (1974) demostró que las reglas RI1, RI2 y RI3 (reglas de inferencia de Amstrong) son **sólidas** y **completas**.
- Sólida: Dado un conjunto de DFs F especificados en un esquema de una relación R, cualquier DF que podamos inferir de F usando solo RI1, RI2 y RI3, se cumple en cada estado de la relación r de R que satisfaga las DFs de F (**a partir de una DF que se cumple en R, aplicando las reglas de inferencia de Amstrong, obtenemos otra DF que se cumple en R**).
- Completa: Usando estas reglas para inferir DFs hasta que no se pueda determinar ninguna otra entonces se genera un conjunto completo de todas las dependencias posibles que se pueden inferir a partir de F. Así la clausura de F se puede obtener con la reglas de inferencia de Amstrong (**se puede obtener F^+ aplicando las reglas de inferencia de Amstrong**).

Dependencias Funcionales: Conjuntos Mínimos

- Dos conjuntos de dependencias F y E son **equivalentes** si toda dependencia de uno se puede inferir de las dependencias del otro y viceversa (es decir $F^+ = E^+$).
- Un conjunto de DFs de F es **mínimo** si:
  - La parte derecha de todas sus dependencias es un solo atributo.
  - Si eliminamos una dependencia, obtenemos un conjunto no equivalente a F.
  - Si eliminamos un atributo en la parte izquierda de una dependencia, obtenemos un conjunto no equivalente a F.

  En otras palabras: DFs en forma **canónica** y sin **redundancias.**
- Una **cobertura mínima** de un conjunto de DFs de F es un conjunto **mínimo** equivalente a F.

Dependencias Funcionales: Conjuntos Mínimos, Algoritmo

Cobertura mínima (E) **/* encuentra una cobertura mínima F de las DFs E */**

1. Establecer F := E

/* Descomponer en dependencias sobre atributos individuales en la parte derecha (atributos redundantes) */

2. Substituir todas las dependencias $X \rightarrow \{ A_1, \ldots, A_n \}$ en F
   por $X \rightarrow A_1, \ldots, X \rightarrow A_n$

/* Eliminar atributos que sobren en las partes izquierdas (atributos redundantes) */

3. for $X \rightarrow A \in F$ do
   for atributo $B \in X$ do
   if $F - \{ X \rightarrow A \} \cup \{ (X - \{B\}) \rightarrow A \}$ es equivalente a F
   then $F := F - \{ X \rightarrow A \} \cup \{ (X - \{B\}) \rightarrow A \}$ (reemplazar $X \rightarrow A$ por $(X - \{B\}) \rightarrow A$ en F)

/* Eliminar dependencias que se infieren de otras (inferencia redundantes) */

4. for $X \rightarrow A \in F$ do
   if $\{F - \{ X \rightarrow A\}\}$ es equivalente a F
   then $F := F - \{ X \rightarrow A\}$ (eliminamos $X \rightarrow A$)

return F

Ejemplo Cobertura Mínima

Supongamos las DFs de E: {DF1: B → A, DF2: D → A, DF3: AB → D}, la cobertura mínima sería: (paginas 511 y 512, Edi. 7)

1. Establecer F := E

/* Descomponer en dependencias sobre atributos individuales en la parte derecha */

2. Ya lo están.

/* Eliminar atributos que sobren en las partes izquierdas (atributos redundantes) */

3. De la única que podemos quitar es de AB → D ¿Podemos quitar A o B de la parte izquierda quedando B → D o A → D? Es decir ¿podríamos sustituir AB → D por B → D o A → D? Supongamos la primera DF1: B → A, aumentando con B (RI2) inferimos BB → AB o lo que es lo mismo B → AB. Así tenemos estas dos DFs: {B → AB, AB → D} |= B → D (RI3). Por lo tanto si sustituimos AB → D por B → D tenemos lo mismo, ya que B → D se infiere de las dos DFs.

/* Eliminar dependencias que se infieren de otras (inferencia redundantes) */

4. Hacer notar que tenemos {B → A, D → A, B → D}, así que {B → D, D → A} |= B → A y esta ya está, y por lo tanto es redundante. Así F:={B → D, D → A}

return F (es una cobertura mínima)

Normalización de Datos (Basadas en PKs)

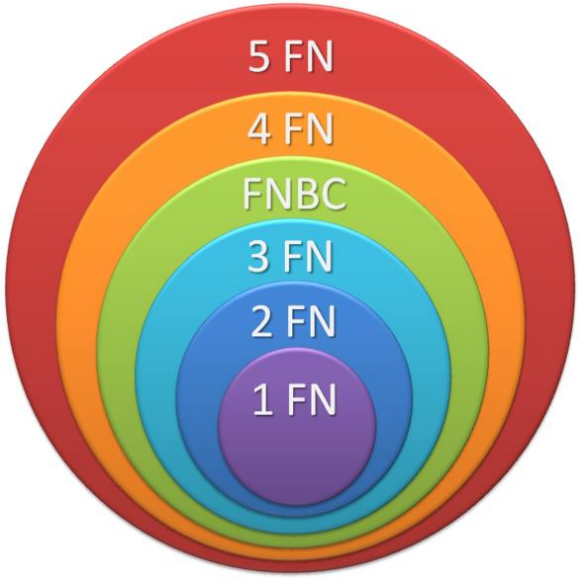
- En 1972 Codd hace pasar un esquema de relación por una serie de comprobaciones para certificar que satisface cierta forma normal (E. F. Codd. Further Normalization of the Data Base Relational Model. IBM Research Report, San Jose, California RJ909: (1971))
- Codd propone la 1ª, 2ª y 3ª Formas Normales (FNs), posteriormente Boyce y Codd proponen la BCNF (más estricta que la primera).
- La normalización de datos es el proceso por el cual los esquemas de relación insatisfactorios se descomponen en esquemas más pequeños con propiedades más deseables.
- **Deseables????** El objetivo es evitar las diversas anomalías que hemos comentado anteriormente.

Normalización de Datos (Basadas en PKs)

- El proceso consiste en efectuar una serie de pruebas sobre el esquema relacional propuesto:
  - Si fallan las pruebas, la relación se descompone en relaciones más pequeñas que si son capaces de satisfacer las pruebas.
  - Por si solo las FNs no garantizan de manera absoluta un buen diseño de la BD, se tienen que cumplir otras propiedades:
    - Preservación de atributos.
    - Reunión sin perdida o reunión no aditiva (se garantiza que no hay tuplas espurias), recordar que hay que diseñar tablas sobre las que se pueda hacer un NJ sobre atributos que sean PK o FK (ESTRICTA).
    - Conservación de dependencias: todas la DFs están representadas en alguna relación individual tras la descomposición (SE PUEDE SACRIFICAR).
- Así una FN de una relación hace referencia a la FN más alta que cumple e indica el grado en la que ha sido normalizada.

Formas Normales

- Es un marco formal para el análisis de los esquemas de relación en claves y en dependencias funcionales entre sus atributos.
- Se pasan una serie de pruebas a los esquemas hasta normalizar al grado deseado.
- Son incrementales
  - Si se cumple la forma normal n-ésima se cumple la (n-1)-ésima



Formas Normales

- Formas normales 1ª, 2ª, 3ª, BCNF
  - Involucran un solo esquema
  - No eliminan totalmente la posibilidad de anomalías de actualización, pero las reducen a casos muy excepcionales en la práctica
- Formas 4ª, 5ª y 6ª
  - Eliminan sucesivamente más anomalías de actualización
- **Se normaliza para evitar redundancia, mantener la integridad de los datos y mejorar el redimiendo del SGBD.**

1ª forma normal (1NF)

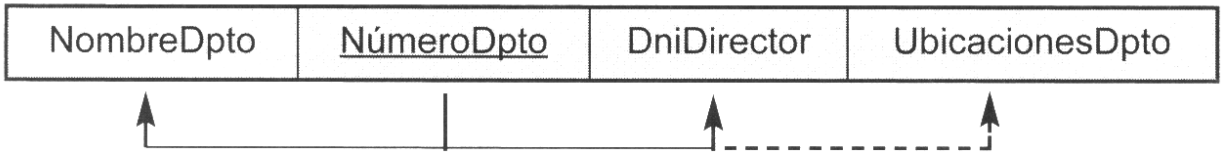
- Decimos que un esquema está 1NF si:
  - Los atributos son atómicos y univaluados.
  - Los nombres de atributo son únicos.
  - No hay tuplas duplicadas (consecuencia: todo esquema tiene alguna clave).
  - El orden de tuplas y atributos es arbitrario .
- Realmente estas propiedades se considera parte inherente del modelo relacional.
- Aunque tenemos que hacer notar que:
  - SQL sólo cumple la primera de estas condiciones (el tratamiento de NULL se sale también del modelo relacional)
  - Se estudian alternativas como el modelo relacional anidado, que admite relaciones como valores de atributos
- **Esta Forma Normal elimina los valores repetidos en una BD**

1ª forma normal (1NF)

- (a) **NO** esta en 1FN, por ejemplo mirar (b)UbicacionesDpto no es un atributo atómico, en cambio en (c) **SI** esta en 1FN (PK es {UbicacionesDpto,NumeroDpto})

(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

- La solución (c) démonos cuenta que introduce redundancia.
- Si seguimos posteriormente con el proceso de normalización quitaríamos esta redundancia y nos llevaría a **otra posible solución** que podríamos a haber tomado: **hacer una tabla independiente**.

1ª forma normal (1NF)

- ¿Como hacemos una tabla independiente?: sería quitar el atributo UbicacionesDpto (es el atributo que infringe 1NF) y se pone en una tabla aparte junto con el atributo NumeroDpto (FK). En la nueva tabla la PK es {UbicacionesDpto, NumeroDpto}.

LOCALIZACIONES_DPTO

NúmeroDpto	UbicaciónDpto
1	Madrid
4	Gijón
5	Valencia
5	Sevilla
5	Madrid

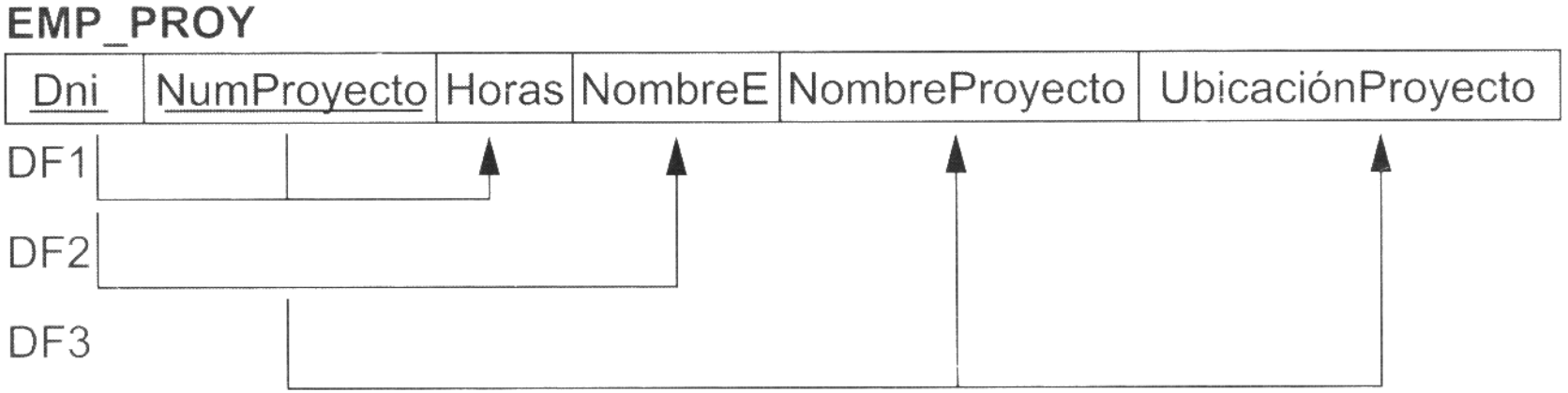
- Otra solución si se conoce el número máximo de valores para cada atributo, digamos por ejemplo tres, se sustituye ese atributo multievaluado por tres atributos atómicos UbicacionDpto1, UbicacionDpto2 y UbicacionDpto3.
- Esta solución es menos deseable.
- Esto tiene el consiguiente problema de NULLs y además la consulta de este atributo se hace más complicada (ejemplo de consulta: departamentos que tienen Valencia como ubicación).

2ª forma normal (2NF)

- Está basado en el concepto de DF total (si se elimina cualquier atributo de X se rompe la dependencia).
- Una **dependencia funcional X → Y es plena** si no le sobra ningún atributo a X, es decir X – {A} no determina funcionalmente a Y, $\forall A \in X$.
- Dicho de otro modo, los atributos dependen de la clave completa; sólo los atributos de una clave pueden depender de partes de éstas.
- **Atributo no primo** o no primario = atributo que no es parte de ninguna clave.
- **Un esquema R está en 2FN si todo atributo no primo forma una DF total de alguna clave candidata.**
- Un esquema R es 2NF si todo atributo no primario de R tiene una dependencia funcional plena con las claves de R.
- Recordar que atributos primarios son los que forman parte de alguna clave.

2ª forma normal (2NF)

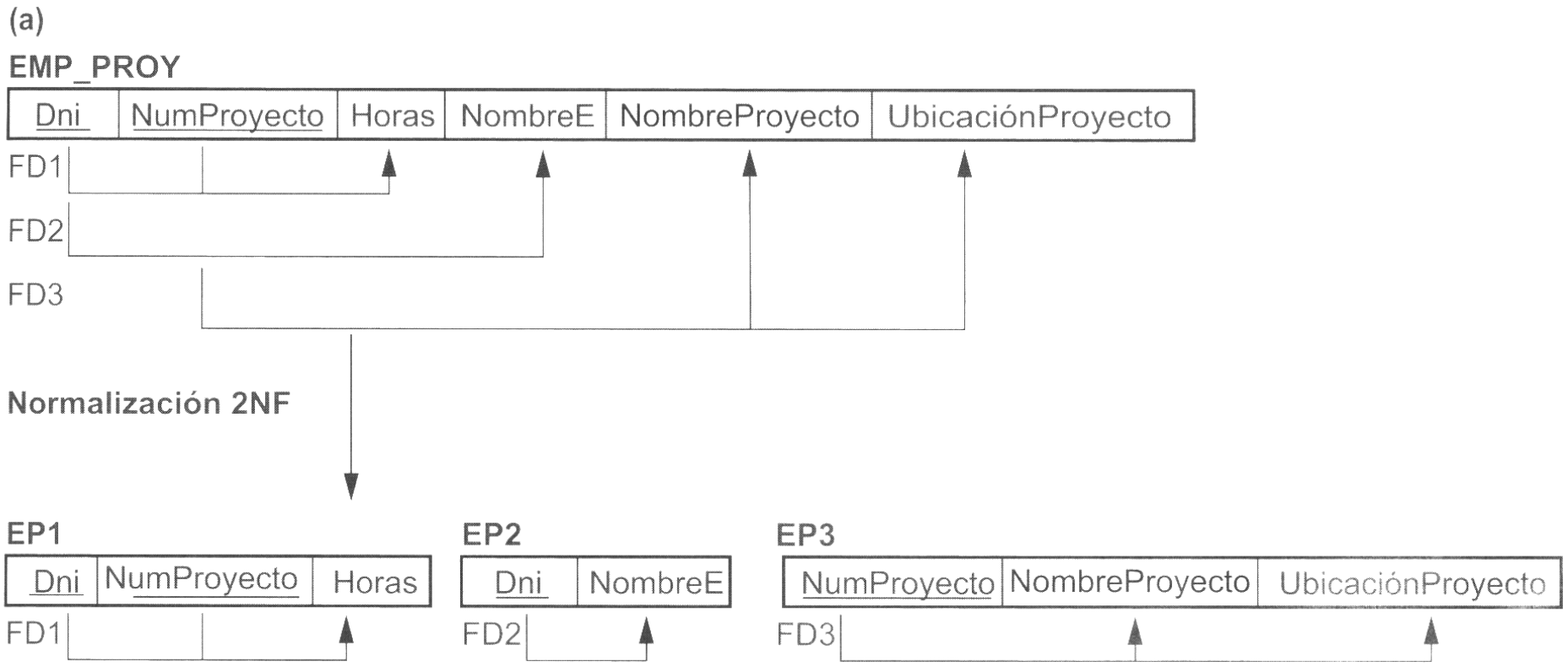
- Una **dependencia funcional X → Y es parcial** si X – {A} → Y, con A ∈ X.
  - La DF **{Dni, NumProyecto}→ Horas** es completa ya que ni **Dni→ Horas** ni **NumProyecto→ Horas** son dependencias válidas.
  - Sin embargo la DF **{Dni, NumProyecto}→ NombreE** es una dependencia parcial porque también se cumple que **Dni → NombreE** es una DF.



- Así DF1 cumple 2NF, pero DF2 y DF3 no cumplen 2NF ya que no son dependencias funcionales completas con la clave candidata {Dni, NumProyecto}.
- **OJO:** Lo atributos no primos tienen que depender completamente de la clave candidata. Solo se permite depender a atributos de una clave candidata (atributos primos) con partes de claves candidatas (esta permitido en 2NF).

2ª forma normal (2NF)

- El proceso de normalización de 1NF → 2NF se hace **dividiendo** la relación en varias relaciones en las que los **atributos no primos** estén **asociados solo a la parte de alguna clave candidata o claves candidatas de la que dependen.**



2ª forma normal (2NF)

- **La comprobación 2FN implica la comprobación de la DFs de la relación cuyos atributos del lado izquierdo forman parte de alguna clave candidata.**
- Si la clave candidata contiene un solo atributo no hace falta comprobar la verificación.
- En la tabla EMP_PROY la clave primaria esta formada por {Dni, NumProyecto} y no hay ninguna otra clave candidata.
- El atributo no primo NombreE que forma parte de DF2 no cumple 2FN ya no depende totalmente de la clave primaria o alguna clave candidata.
- Los atributos no primos NombreProyecto y UbicaciónProyecto que forman parte de DF3 incumplen 2FN por la misma razón.
- La normalización 2NF se realiza dividendo la relación en relaciones en las que los atributos no primos solo estén asociados a la parte de la clave principal de la que son completa y funcionalmente dependientes. Ver figura anterior.

Ejemplo de eliminación de redundancia con 2FN

EMPLEADO

NombreE	Dni
Pérez Pérez, José	123456789
Campos Sastre, Alberto	333445555
Jiménez Celaya, Alicia	999887777
Sainz Oreja, Juana	987654321
Ojeda Ordóñez, Fernando.	666884444
Oliva Avezuela, Aurora	453453453
Pajares Morera, Luis	987987987
Ochoa Paredes, Eduardo	888665555

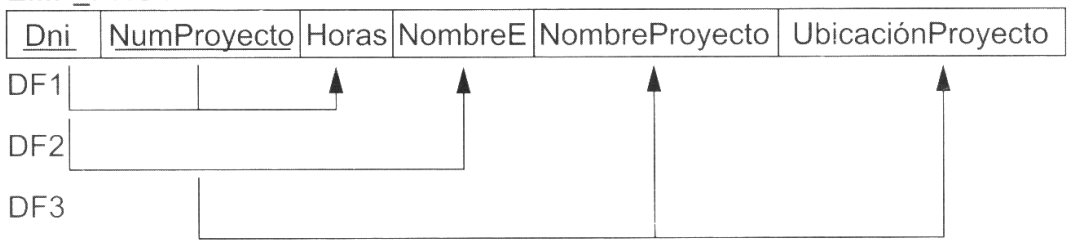
PROYECTO

NombreProyecto	NumProyecto	UbicaciónProyecto
ProductoX	1	Valencia
ProductoY	2	Sevilla
ProductoZ	3	Madrid
Computación	10	Gijón
Reorganización	20	Madrid
Comunicaciones	30	Gijón

TRABAJA_EN

Dni	NumProyecto	Horas
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

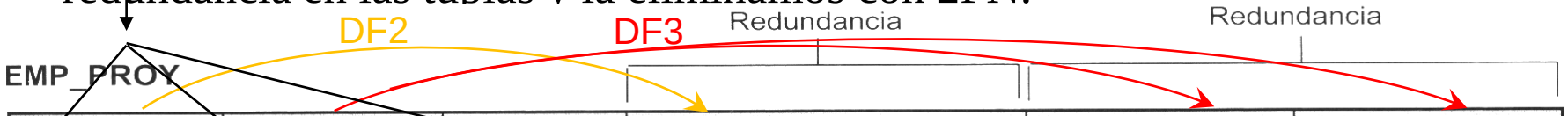
EMP_PROY



Ejemplo de eliminación de redundancia con 2FN

DF1 no genera redundancia

- Si ahora combinamos información de EMPLEADO, PROYECTO y TRABAJA_EN obtenemos la tabla EMP_PROY. Aparece una tabla con redundancia en las tuplas y la eliminamos con 2FN.



EMP_PROY

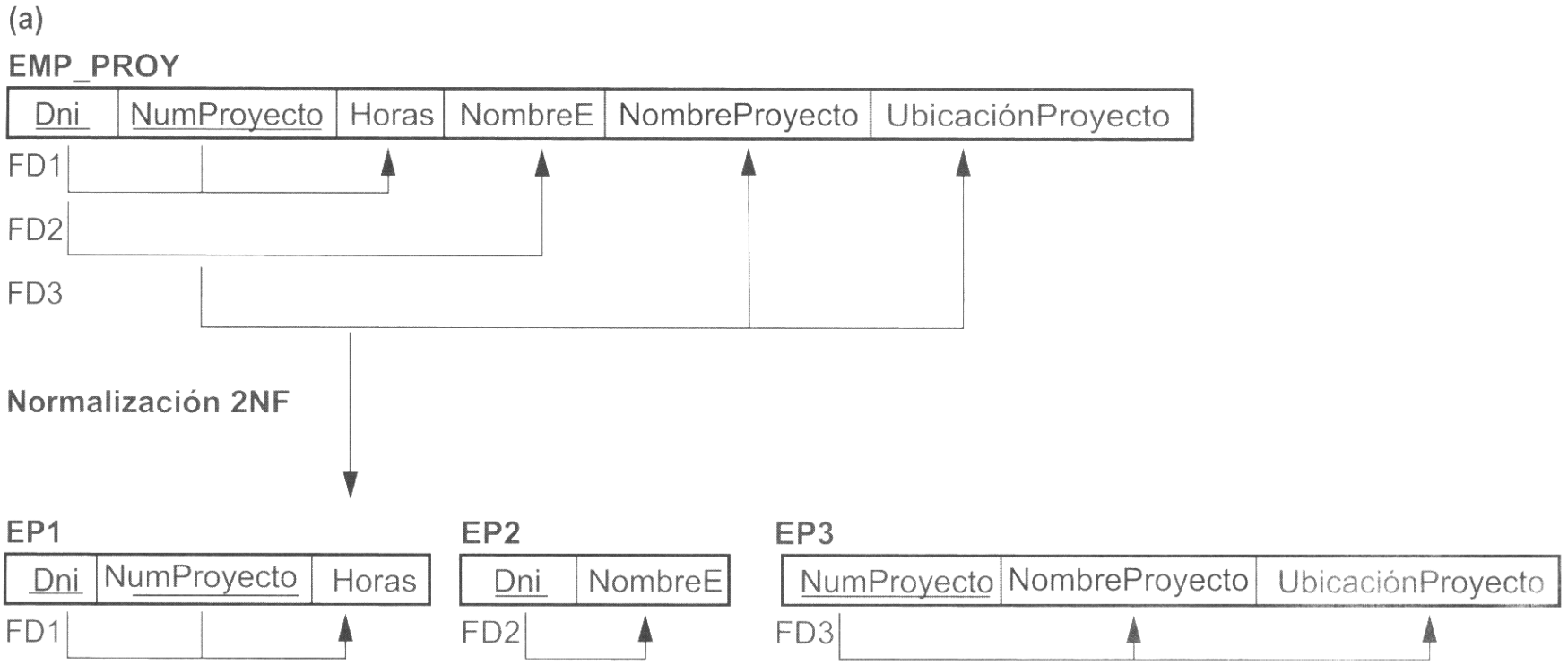
Dni	NumProyecto	Horas	NombreE	NombreProyecto_	UbicaciónProyecto
123456789	1	32.5	Pérez Pérez, José	ProductoX	Valencia
123456789	2	7.5	Pérez Pérez, José	ProductoY	Sevilla
666884444	3	40.0	Ojeda Ordóñez, Fernando.	ProductoZ	Madrid
453453453	1	20.0	Oliva Avezuela, Aurora	ProductoX	Valencia
453453453	2	20.0	Oliva Avezuela, Aurora	ProductoY	Sevilla
333445555	2	10.0	Campos Sastre, Alberto	ProductoY	Sevilla
333445555	3	10.0	Campos Sastre, Alberto	ProductoZ	Madrid
333445555	10	10.0	Campos Sastre, Alberto	Computación	Gijón
333445555	20	10.0	Campos Sastre, Alberto	Reorganización	Madrid
999887777	30	30.0	Jiménez Celaya, Alicia	Comunicaciones	Gijón
999887777	10	10.0	Jiménez Celaya, Alicia	Computación	Gijón
987987987	10	35.0	Pajares Morera, Luís	Computación	Gijón
987987987	30	5.0	Pajares Morera, Luís	Comunicaciones	Gijón
987654321	30	20.0	Sainz Oreja, Juana	Comunicaciones	Gijón
987654321	20	15.0	Sainz Oreja, Juana	Reorganización	Madrid
888665555	20	Null	Ochoa Paredes, Eduardo	Reorganización	Madrid

Ejemplo de eliminación de redundancia con 2FN

- **¿Porqué hay redundancia en tablas que no cumplen 2FN?**
- Supongamos la DF **Dni→ NombreE**, esta dependencia genera repeticiones:
  - Recordemos que la clave primaria es: {**Dni, NumProyecto}**
  - Por lo tanto puede haber tuplas con el mismo **Dni** (Dni aquí no es clave primaria).
  - Así como existe la DF **Dni→ NombreE**, hay todavía más repeticiones (por la definición de Dependencia Funcional: si encontramos tuplas con el mismo **Dni** a la fuerza **NombreE** es el mismo en esas tuplas.
- Para resolver este problema hacemos tablas base más pequeñas sin esas repeticiones mediante el proceso de normalización:
  - todos los atributos no primos dependen totalmente de alguna clave.

Ejemplo de eliminación de redundancia con 2FN

- El proceso de normalización de 1NF → 2NF se hace dividiendo la relación en varias relaciones en las que los atributos no primos estén asociado solo a la parte de alguna clave candidata o claves candidatas de la que dependen.

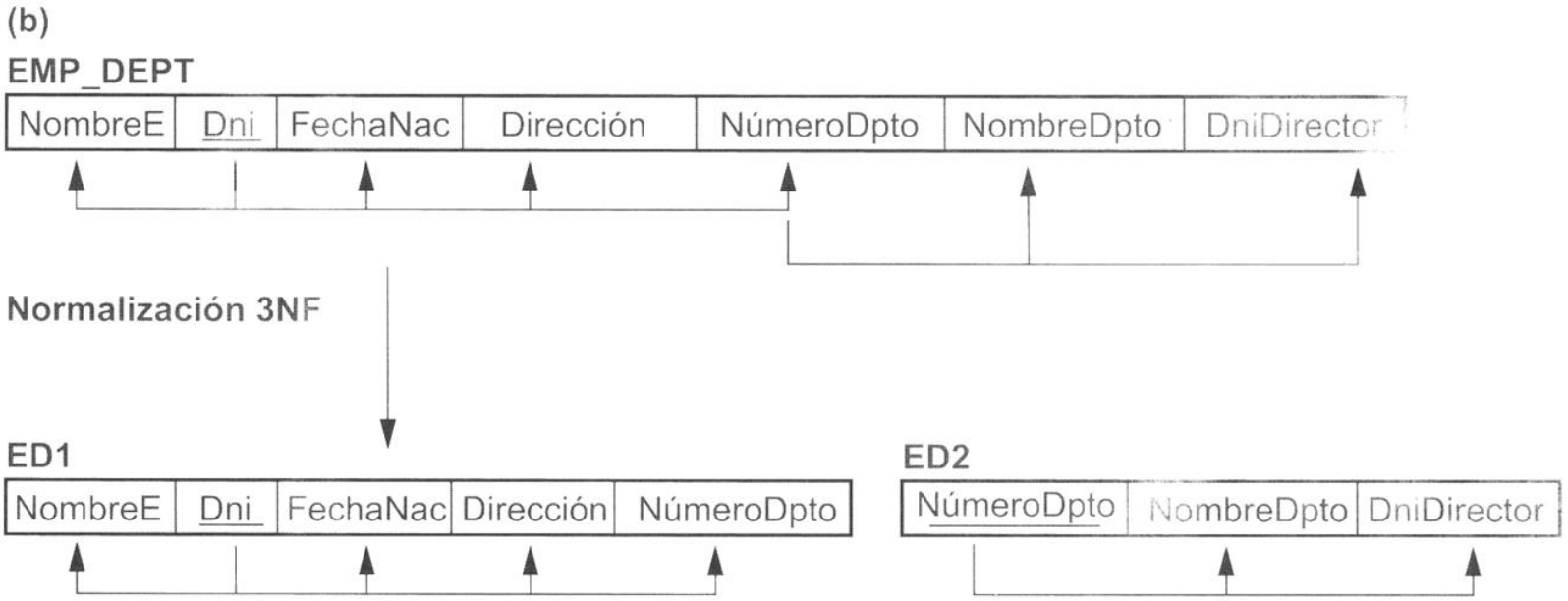


3ª forma normal (3NF)

- Se basa en el concepto de dependencia transitiva: una dependencia funcional X → Y en un esquema de relación R es una dependencia transitiva si existe un conjunto de atributos Z que ni es clave candidata ni es un subconjunto de una clave de R, y se cumple tanto X → Z como Z → Y.
- Ejemplo: en la relación EMP_DEPT la dependencia Dni → DniDirector es transitiva a través de NumeroDpto, ya que se cumplen las dependencias Dni → NumeroDpto y NumeroDpto → DniDirector (nota: NumeroDpto no es una clave por si misma ni un subconjunto de clave de EMP_DEPT).
- **3FN = 2FN+ningún atributo no primo depende transitivamente de la clave principal o primaria a través de un atributo no primo.**

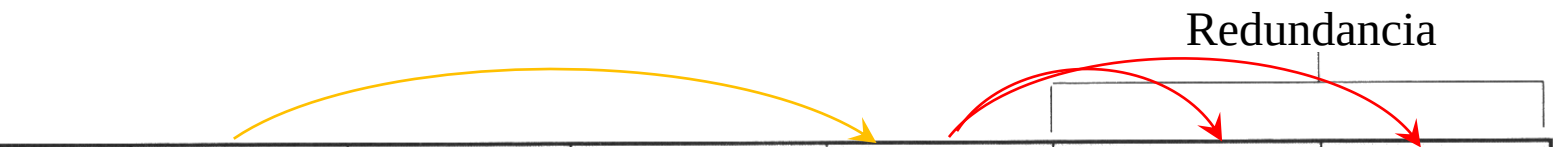
3ª forma normal (3NF)

- **Se puede demostrar que una definición equivalente es que un esquema es 3NF si para toda dependencia $X \rightarrow A$ no trivial ($X \rightarrow A$ es trivial si $A \subset X$), o bien X es una superclave, o bien A es un atributo primo.**
- Dicho de otro modo, no puede un atributo depender de algo que no sea una superclave, excepto acaso los atributos que forman parte de alguna clave.



3ª forma normal (3NF)

- **También las relaciones que están en 2FN pueden tener redundancia, por el mismo motivo de antes: hay dependencias funcionales que la parte Y depende de algo X que no es clave.**
- **<u>Dni</u> → NumeroDpto y NumeroDpto → {NombreDpto, DniDirector}, y NumeroDpto no es C-1ª (es un atributo no primo)**

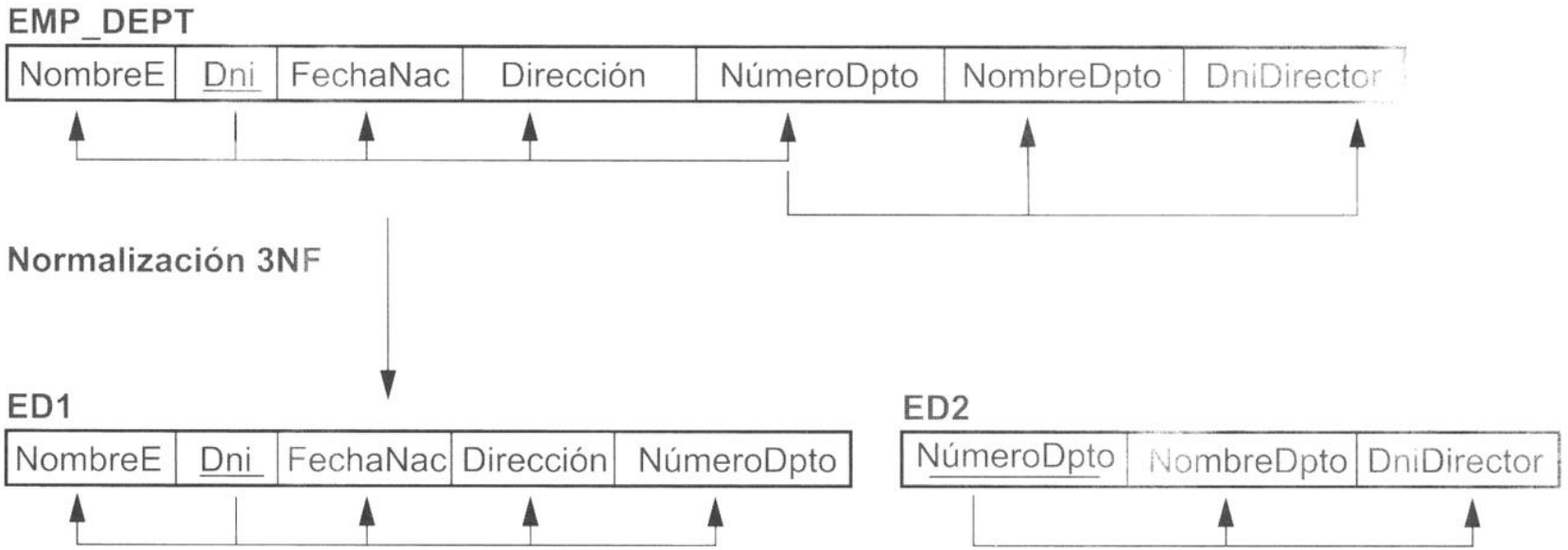


EMP_DEPT

NombreE	<u>Dni</u>	FechaNac	Dirección	NúmeroDpto	NombreDpto	DniDirector
Pérez Pérez, José	123456789	09-01-1965	Eloy I, 98	5	Investigación	333445555
Campos Sastre, Alberto	333445555	08-12-1955	Avda. Ríos, 9	5	Investigación	333445555
Jiménez Celaya, Alicia	999887777	19-07-1968	Gran Vía, 38	4	Administración	987654321
Sainz Oreja, Juana	987654321	20-06-1941	Cerquillas, 67	4	Administración	987654321
Ojeda Ordóñez, Fernando.	666884444	15-09-1962	Portillo, s/n	5	Investigación	333445555
Oliva Avezuela, Aurora	453453453	31-07-1972	Antón, 6	5	Investigación	333445555
Pajares Morera, Luis	987987987	29-03-1969	Enebros, 90	4	Administración	987654321
Ochoa Paredes, Eduardo	888665555	10-11-1937	Las Peñas, 1	1	Sede central	888665555

3ª forma normal (3NF)

- EMP_DEPT está en 2NF ya que no existen dependencias no parciales de una clave.
- No esta en 3NF ya que existen dependencias transitivas a través de NúmeroDpto:
  - **Dni → NombreDpto** y **Dni → DniDirector** son dependencias transitivas a través del atributo NúmeroDpto .
  - La DF **NúmeroDpto → {NombreDpto, DniDirector}** no esta en 3NF (la parte izquierda no es una superclave, la parte derecha no es un atributo primo).
- Podemos descomponerla en ED1 y ED2, de tal forma que si hacemos una NJ entre esas tablas recuperamos la tabla original EMP_DEPT sin tuplas espurias.



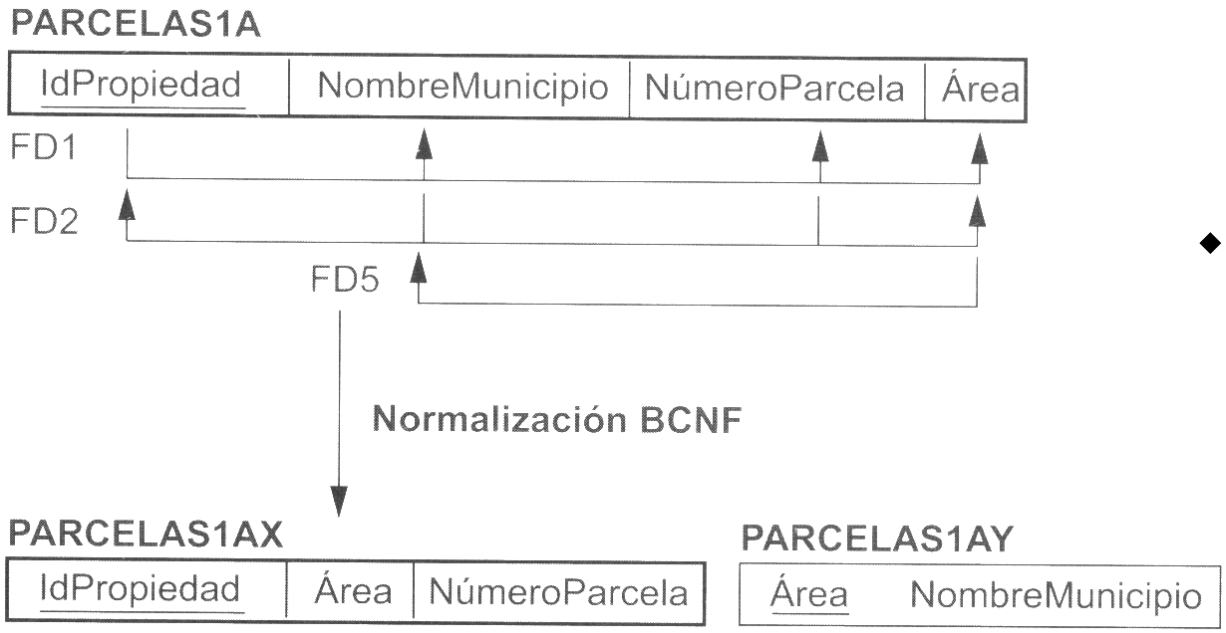
Resumen formas normales

- **Intuitivamente podemos ver que cualquier dependencia funcional en la que el lado izquierdo es parte de la clave principal (subconjunto propio) , o es un atributo no clave, implica una DF problemática.**
- La normalización 2FN y 3FN eliminan esas dependencias problemáticas.

Forma normal	Prueba	Remedio (normalización)
Primera (1FN)	La relación no debe tener atributos multivalor o relaciones anidadas.	Generar nuevas relaciones para cada atributo multivalor o relación anidada.
Segunda (2FN)	Para relaciones en las que la clave principal contiene varios atributos, un atributo no clave debe ser funcionalmente dependiente en una parte de la clave principal.	Descomponer y configurar una nueva relación por cada clave parcial con su(s) atributo(s) dependiente(s). Asegurarse de mantener una relación con la clave principal original y cualquier atributo que sea completa y funcionalmente dependiente de ella.
Tercera (3FN)	La relación no debe tener un atributo no clave que esté funcionalmente determinado por otro atributo no clave (o por un conjunto de atributos no clave). Esto es, debe ser una dependencia transitiva de un atributo no clave de la clave principal.	Descomponer y configurar una relación que incluya el(los) atributo(s) no clave que determine(n) funcionalmente otro(s) atributo(s) no clave.

Forma normal Boyce-Codd (BCNF)

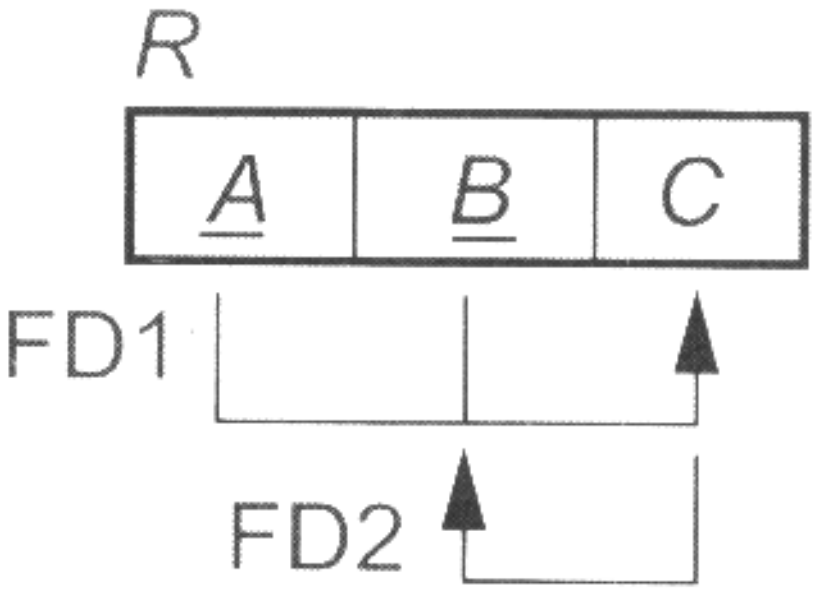
- **Un esquema R es BCNF si para toda dependencia X → Y no trivial**
  X es una superclave de R
- Dicho de otro modo, no puede haber más dependencia que con las superclaves.
- Este esquema tiene por claves candidatas: IdPropiedad y {NombreMunicipio, NúmeroParcela}, se elige como primaria IdPropiedad.



- La parte izquierda de DF5 no es superclave de la relación PARCELAS1A.
- Notar que si es 3FN ya que la parte derecha es un atributo primo (NombreMunicipio forma parte de una clave candidata).

Forma normal Boyce-Codd (BCNF)

- Para pasar a BCNF perdemos la dependencia DF2 ya que sus atributos no coexisten en una misma relación.
- Notar que en la normalización BCNF se pueden perder dependencias, al contrario de lo que pasa en 2FN y 3FN.
- Generalmente los esquemas que están en 3FN lo están BCNF. Solo si se cumple $X \rightarrow A$ en un esquema de relación R, no siendo X un superclave y siendo A un atributo primo estarán en 3FN pero no BCNF:



- FD1 está en 3FN ya que la parte izquierda es una superclave y FD2 también porque la parte derecha es un atributo primo.
- FD2 no está en BCNF ya que la parte izquierda no es superclave, pero FD1 si está en BCNF.

Forma normal Boyce-Codd (BCNF)

- Supongamos que tenemos la siguiente relación con las siguientes dependencias:
  - DF1: {Estudiante, Curso} → Profesor
  - DF2: Profesor → Curso
- {Estudiante, Curso} es una clave candidata y clave primaria en este caso.
- Esta relación esta en 3FN pero no en BCNF.

Estudiante	Curso	Profesor
Campos	Bases de datos	Marcos
Pérez	Bases de datos	María
Pérez	Sistemas operativos	Amanda
Pérez	Teoría	Sergio
Ochoa	Bases de datos	Marcos
Ochoa	Sistemas operativos	Aurora
Morera	Bases de datos	Eduardo
Celaya	Bases de datos	María
Campos	Sistemas operativos	Amanda

- DF2 es la razón de no estar en BCNF. 3 opciones para BCNF:

1. {Estudiante, Profesor} y {Estudiante Curso}
2. {Curso, Profesor} y {Curso, Estudiante}
3. {Profesor, Curso} y {Profesor, Estudiante}

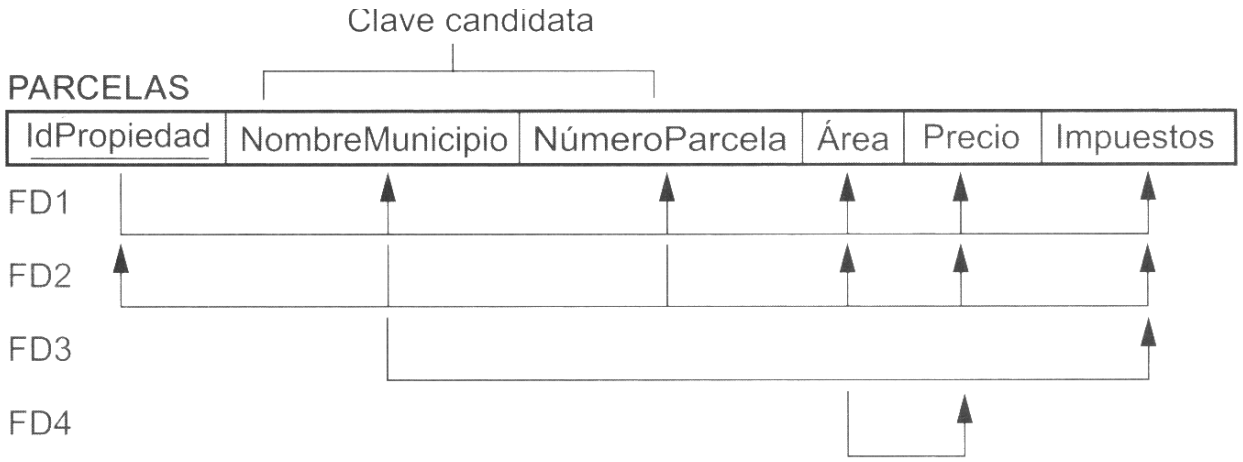
- Las 3 posibilidades rompen DF1, pero la apropiada es la (3) ya que no genera tuplas falsas tras una concatenación (se puede comprobar mediante el test del **join sin perdidas**).

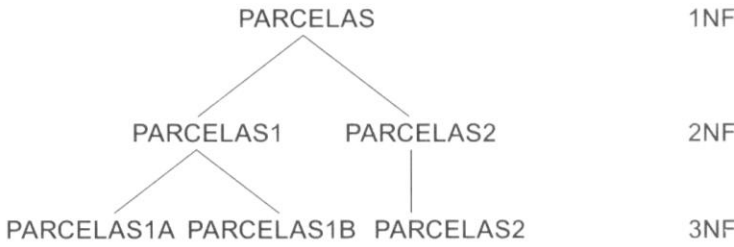
Forma normal Boyce-Codd (BCNF o 3.5NF)

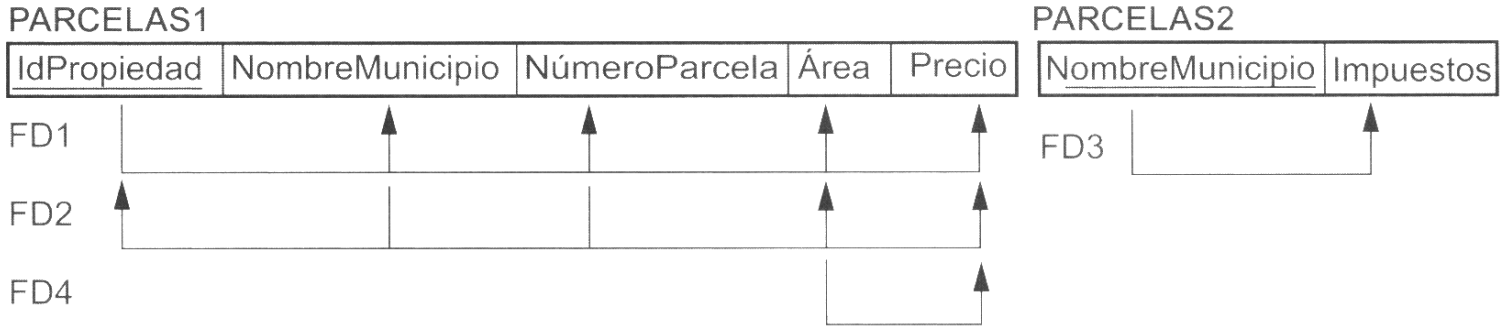
- Vemos que al normalizar a {Profesor, Curso} y {Profesor, Estudiante}, se elimina la redundancia de **Profesor → Curso** en la tabla (ya que en la tabla la clave primaria es {Estudiante, Curso}).
- Podemos ver por ejemplo que Marcos y Bases de datos esta varias veces repetida, y con la normalización se elimina esa redundancia.
- Observar la FK y PK de las nuevas relaciones base.

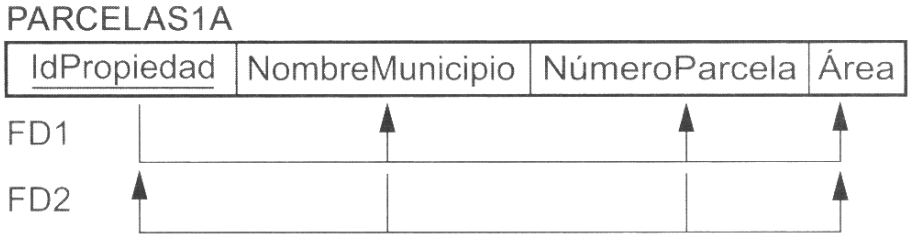
Estudiante	Curso	Profesor
Campos	Bases de datos	Marcos
Pérez	Bases de datos	María
Pérez	Sistemas operativos	Amanda
Pérez	Teoría	Sergio
Ochoa	Bases de datos	Marcos
Ochoa	Sistemas operativos	Aurora
Morera	Bases de datos	Eduardo
Celaya	Bases de datos	María
Campos	Sistemas operativos	Amanda

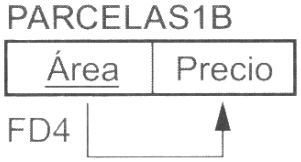
Mas Ejemplos











¿Normalizar o Desnormalizar?

- Hay veces que en una BD, por razones de optimización de rapidez en consultas, es conveniente desnormalizar (aumentar la redundancia de las relaciones base).
- Así mantener unas redundancias controladas en las tablas en algunos casos puede ser bueno.
- En algunos casos el tener la BD en niveles altos de normalización puede no significar más eficiencia: mientras más normalización algunas veces menor rendimiento (muchas concatenaciones para recuperar información).
- Es decisión del analista del SGDB encontrar un balance entre la normalización y desnormalización de la BD.
- Para desnormalizar se requiere que la BD este en su nivel optimo de normalización: 3FN o 3.5FN.
- Para desnormalizar por ejemplo se pueden utilizar vistas materializadas.

Algoritmos de normalización

- Comprobación de preservar dependencias en descomposiciones
- Comprobación de join sin pérdida en descomposiciones
- Comprobación de que un conjunto de atributos es una superclave
- Propiedad 3NF, BCNF de relaciones
- Descomposición de relaciones a 3NF, BCNF
  - Siempre es posible descomponer a 2NF y 3NF sin pérdida de dependencias
  - BCNF puede no ser posible sin perder alguna dependencia

Normalización 3NF (Cap 11)

3NF (R, F) **/* Esta descomposición tiene "join" sin pérdida, y preserva las dependencias */**

/* R es cualquier relación universal y F un conjunto de DFs de R */

1. D := ∅
2. G := **cobertura mínima** de F **/* se localiza la cobertura mínima G para F */**

/* Sacar a tablas aparte los atributos de todos los conjuntos de dependencias con la misma parte izquierda */

3. for $X \rightarrow Y \in G$ (cobertura mínima)

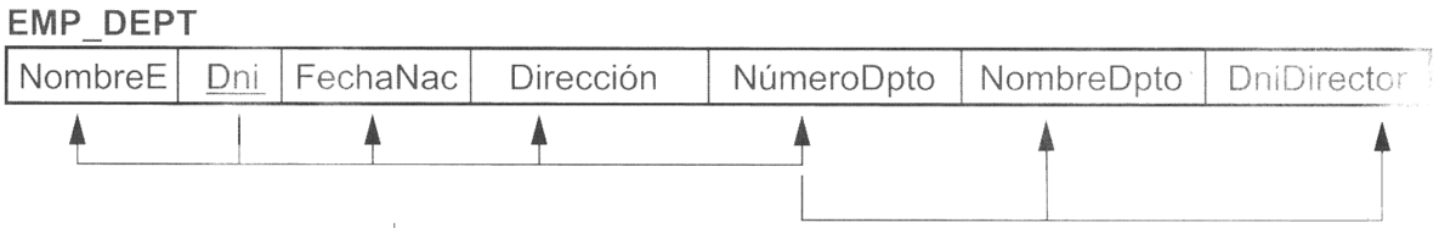
   Añadir a D el esquema $X \cup \{ A_1, A_2, \ldots, A_n \}$

   donde $X \rightarrow A_i$ son todas las dependencias sobre X en G

/* Si no ha salido ninguna tabla con la clave original completa, crear esa tabla */

4. Si ningún esquema de D contiene una clave de R, añadir a D un esquema con una clave de R
5. Eliminar los esquemas redundantes de D (esquemas incluidos en otros)

Ejemplo de Normalización 3NF (algoritmo)



R={A B C D E F G}, DF1: B → ACDE, DF2: E → FG

3NF (R, F) **/* Esta descomposición tiene "join" sin pérdida, y preserva las dependencias */**

1. D := ∅
2. G := **cobertura mínima** de R **/* se localiza la cobertura mínima G para R */**
   - Pasamos de B → ACDE a B → A, B → C, B → D, B → E
   - Pasamos de E → FG a E → F, E → G, estas 6 DFs ya tienen sus partes derechas en su forma simple
   - Las partes izquierdas también están en su forma simple y por tanto no hay redundancia de atributos
   - No hay dependencias redundantes (DFs que se infieren de otra DFs) y por tanto la cobertura mínima G
   - G:={B → A, B → C, B → D, B → E, E → F, E → G}

Ejemplo de Normalización 3NF (algoritmo)

/* Sacar a tablas aparte los atributos de todos los conjuntos de dependencias con la misma parte izquierda */

3. for $X \rightarrow Y \in G$ (cobertura mínima)

   Añadir a D el esquema $X \cup \{ A_1, A_2, \ldots, A_n \}$

   donde $X \rightarrow A_i$ son todas las dependencias sobre X en G

Así que tenemos:

D:={B ∪{A, C, D, E}, E ∪{F, G}}= {{B, A, C, D, E}, {E, F, G}}={R1, R2}

/* Si no ha salido ninguna tabla con la clave original completa, crear esa tabla */

4. Si ningún esquema de D contiene una clave de R, añadir a D un esquema con una clave de R (pero si hay una clave de R que es B, así que no es el caso).
5. Eliminar los esquemas redundantes de D (esquemas incluidos en otros) que no es el caso.
6. La clave primaria de R1 es B y la clave primaria de R2 es E. El atributo E en R1 es clave foránea (esta descomposición tiene “join” sin pérdida, y preserva las dependencias).

Normalización BCNF (Cap 11)

BCNF (R, F) **/* El algoritmo genera una descomposición que tiene join sin pérdida, pero no asegura la preservación de todas las dependencias */**

D := {R}

while D contiene una relación no BCNF

Q := elegir una dependencia funcional no BCNF en D

$X \rightarrow Y$:= elegir una dependencia de F en Q que no cumple BCNF

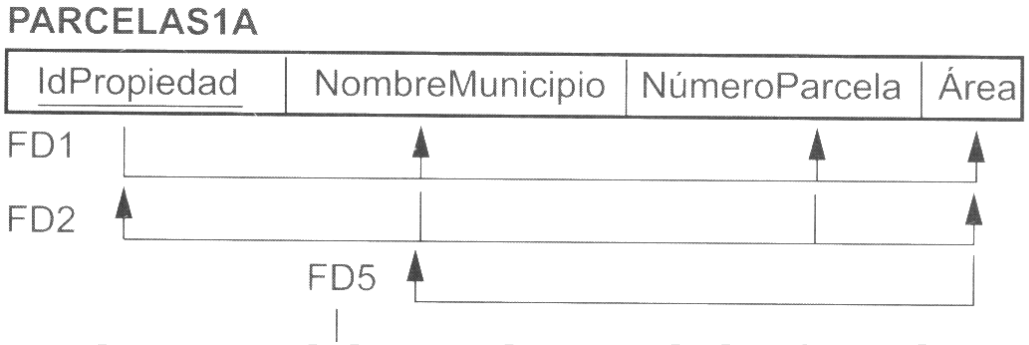
Substituir Q en D por dos esquemas $(Q - Y)$, $(X \cup Y)$ ***(es decir en cada pasada del while se descompone un esquema Q que no esta en BCNF en dos esquemas que si son BCNF)***

En otras palabras…

1. Sacar a tablas aparte todas las dependencias no BCNF de la relación original, pero **eliminando de ésta la parte derecha** de las dependencias
2. Repetir el proceso sobre las relaciones que van saliendo

- NOTA: En **eliminando de ésta la parte derecha** se pueden perder dependencias.

Ejemplo Normalización BCNF (algoritmo)



Este esquema tiene por claves candidatas: IdPropiedad y {NombreMunicipio, NúmeroParcela}, se elige como primaria IdPropiedad.

R={A B C D}, y clave candidata {BC}, la dependencias funcionales son

DF1: A $\rightarrow$ BCD, DF2: BC $\rightarrow$ AD, DF3: D $\rightarrow$ B

Solo hay una DF que no cumple BCNF que es DF3: D $\rightarrow$ B

Substituir el esquema R por dos esquemas (R – B) y (D $\cup$ B)

R={R1, R2}={{A C D}, {D B}}

La clave primaria de R1 es A y de R2 es D. El atributo D en R1 es clave foránea.

Este algoritmo genera una descomposición que tiene join sin pérdida, pero no asegura la preservación de todas las dependencias (por ejemplo la dependencia DF2 ya no existe).

Ejemplo Normalización BCNF (algoritmo)

- Supongamos la siguiente relación
- R={Estudiante, Curso, Profesor}
- con las siguientes dependencias:
  - DF1: {Estudiante, Curso} → Profesor
  - DF2: Profesor → Curso

R={A, B, C}, DF1: AB →C, DF2: C → B

Estudiante	Curso	Profesor
Campos	Bases de datos	Marcos
Pérez	Bases de datos	María
Pérez	Sistemas operativos	Amanda
Pérez	Teoría	Sergio
Ochoa	Bases de datos	Marcos
Ochoa	Sistemas operativos	Aurora
Morera	Bases de datos	Eduardo
Celaya	Bases de datos	María
Campos	Sistemas operativos	Amanda

Solo hay una DF que no cumple BCNF que es DF2: C → B

Substituir el esquema R por dos esquemas (R – B) y (C ∪ B)

R={R1, R2}={{A C}, {C B}}

La clave primaria de R1 es AC y de R2 es C. El atributo C en R1 es clave foránea.

Este algoritmo genera una descomposición que tiene join sin pérdida, pero no asegura la preservación de todas las dependencias (por ejemplo la dependencia DF1 ya no existe).

.

Test para Join sin pérdida (Cap 11)

Test sencillo para una descomposición binaria:

Una descomposición binaria $\{R_1, R_2\}$ de una relación R tiene join sin pérdida respecto a un conjunto de dependencias F si y solo si:

O bien $(R_1 \cap R_2 \rightarrow R_1 - R_2)$ se infiere de F (o está en F^+)

O bien $(R_1 \cap R_2 \rightarrow R_2 - R_1)$ se infiere de F (o está en F^+)

(*Esto quiere decir más o menos o que* $\{R_1 \cap R_2\}$ *es la clave foránea que me relaciona* R_1 *con* R_2 *o al contrario*)

Ejemplo de Test para Join sin pérdida

- R={Estudiante, Curso, Profesor}
- con las siguientes dependencias:
  - DF1: {Estudiante, Curso} → Profesor
  - DF2: Profesor → Curso

Estudiante	Curso	Profesor
Campos	Bases de datos	Marcos
Pérez	Bases de datos	María
Pérez	Sistemas operativos	Amanda
Pérez	Teoría	Sergio
Ochoa	Bases de datos	Marcos
Ochoa	Sistemas operativos	Aurora
Morera	Bases de datos	Eduardo
Celaya	Bases de datos	María
Campos	Sistemas operativos	Amanda

R={A, B, C}, DF1: AB → C, DF2: C → B

F={DF1, DF2}

Según el ejemplo anterior, la única descomposición sin perdidas es: R={R1, R2}=

{{A C}, {C B}}

O bien se infiere R1 ∩ R2 (= C) → R1 – R2 (= A) de F

O bien se infiere R1 ∩ R2 (= C) → R2 – R1 (= B) de F

En este caso la segunda C → B

Luego como sabíamos R1 y R2 producen un NJ sin perdida

DIFERENCIA: *R-S* *es una relación que incluye todas los aributos que están en R pero no están en S.*

Ejemplo de Test para Join sin pérdida

No obstante para la descomposición binaria : R={R1, R2}={{A C}, {A B}}

O bien se infiere R1 $\cap$ R2 (= A) $\rightarrow$ R1 – R2 (= C) de F

O bien se infiere R1 $\cap$ R2 (= A) $\rightarrow$ R2 – R1 (= B) de F

En este caso ni A $\rightarrow$ C, ni A $\rightarrow$ B se infieren de F, por lo tanto R1 y R2 **NO** producen un NJ sin perdida y se tendrían tuplas falsas como resultado.

Así para la descomposición binaria : R={R1, R2}={{B C}, {A B}}

O bien se infiere R1 $\cap$ R2 (= B) $\rightarrow$ R1 – R2 (= C) de F

O bien se infiere R1 $\cap$ R2 (= B) $\rightarrow$ R2 – R1 (= A) de F

En este caso ni B $\rightarrow$ C, ni B $\rightarrow$ A se infieren de F, por lo tanto R1 y R2 **NO** producen tampoco un NJ sin perdida y se tendrían tuplas falsas como resultado.