

AMPLIACIÓN DE BASES DE DATOS

TEMA: Diseño Avanzado de BD

Calidad en el Diseño:

Normalización:

Formas Normales

Desnormalización:

Análisis de redundancias

Diseño orientado a prestaciones

y re-Organización Física de Datos

Fases del Diseño de BD relacionales

Requisitos (escritos en lenguaje natural)

↓ Diseño conceptual (requisitos de I-S)

Esquema conceptual (Modelo entidad-relación)

↓ Diseño lógico (Análisis y Diseño de I-S)

Diseño de tablas (Modelo Relacional)

↓ Diseño físico (uso SQL) (Implementar de I-S)

Organización de las tablas en archivos e índices
(Interno al SGBD)

Cómo verificar la calidad: Normalización

- ◆ Falta de calidad de un *Diseño Deficiente* provoca problemas:
 - Redundancias de Datos
 - Anomalías de actualización
 - Filas incorrectas
 - Exceso de espacio ocupado
- ◆ La *Forma Normal (FN)* de un esquema relacional determina
 - . . . su grado de calidad respecto a esos problemas
 - Cuanto más alta es la *FN* en la que está: mejor calidad
 - Una *FN* se define con unas normas que debe cumplir el esquema
 - Basadas en *Dependencias Funcionales (DFs)* y *Multivaloradas (DMs)*
- ◆ La *Normalización* mejora esos problemas:
 - Descomponiendo el esquema relacional en otros que . . .
 - . . . cumplan *FN* más exigentes (aquellas con numeración más alta)

Problema: Redundancia de datos

<i>Empleados_Centros</i>							
<u>Id empleado</u>	NombreE	DirecciónE	Puesto	Salario	Centro	DirecciónC	Teléfono
123A	Ana Almansa	c/ Argentaes	Profesor	20.000	Informática	Complutense	123
456B	Bernardo Botín	c/ Barcelona	Administrativo	15.000	Matemáticas	Complutense	456
789C	Carlos Crespo	c/ Cruz	Catedrático	30.000	CC. Empresariales	Somosaguas	789
012D	David Díaz	c/ Daroca	Ayudante	10.000	Informática	Complutense	123

- ◆ *Redundancia*: DirecciónC y Teléfono repetido en varias filas (*a veces necesaria*)
 - Se produce una anomalía si
 - se modifica el teléfono o la DirecciónC *solo en algunas* filas
 - ♦ donde está ese teléfono o DirecciónC
 - ◆ Dos problemas como consecuencia de la Redundancia de Datos:
 - Si no se modifican todas las filas provoca *inconsistencias* en la BD
 - Además hay un espacio ocupado excesivo con la redundancia.
 - ◆ ¿Porqué? DirecciónC y Teléfono no “**dependen**” sólo de la PK: Id_empleado
- Veremos que es eso de “**dependen**” más adelante en Dependencias Funcionales

Ese Problema genera: Anomalías de actualización

1. Anomalías de inserción (dos casos)

- a) Al añadir una fila de un empleado adscrito a Informática y con un teléfono distinto de 123
 - Explicación Técnica: (se ve más adelante)
 - ◆ Se inserta una nueva fila sin respetar las DFs

- b) No se puede dar de alta un centro sin empleados destinados en él
 - Porque queda valor nulo en la clave (Id_empleado),
 - ◆ lo cual es imposible.
 - Explicación Técnica: (se ve más adelante)
 - ◆ No puedes añadir filas con valores del consecuente de la DF
 - sin que exista un antecedente para ella.

Ese Problema genera: Anomalías de actualización

2. Anomalías de modificación

- Se modifica el teléfono de Informática
 - sólo en la primera fila donde aparece (caso de la figura)
- Explicación Técnica: (se ve más adelante)
 - se modifica una columna con dato redundante de sólo
 - un subconjunto de las filas con el mismo dato.

3. Anomalías de eliminación

- Si se elimina la segunda fila porque el empleado se da de baja
 - se pierden también los datos del centro
- Explicación Técnica: (se ve más adelante)
 - se eliminan todas las filas en las que aparecen los datos redundantes por lo que se pierde los datos de la DF

Problema: Obtención de datos incorrectos generado por división inadecuada de una relación

Ejemplo 1

- ◆ Para mejorar el diseño se divide una relación en dos, pero al
- ◆ dividir la tabla Empleados_Centros anterior de la siguiente forma
 - y hacer el join posteriormente, se obtienen datos incorrectos
 - ya que se crean filas falsas que no existían originalmente: *F. espurias*

<i>Datos_Empleados</i>						
<u>Id empleado</u>	DirecciónE	Puesto	Salario	Centro	DirecciónC	Teléfono
123A	c/ Argentales	Profesor	20.000	Informática	Complutense	123
456B	c/ Barcelona	Administrativo	15.000	Matemáticas	Complutense	456
789C	c/ Cruz	Catedrático	30.000	CC. Empresariales	Somosaguas	789
012D	c/ Daroca	Ayudante	10.000	Informática	Complutense	123

<i>Ubicaciones_Empleados</i>	
<u>NombreE</u>	DirecciónC
Ana Almansa	Complutense
Bernardo Botín	Complutense
Carlos Crespo	Somosaguas
David Díaz	Complutense

Unión por DirecciónC



Problema: Obtención de datos incorrectos

<u>Id empleado</u>	<u>Nombre E</u>	<u>DirecciónE</u>	<u>Puesto</u>	<u>Salario</u>	<u>Centro</u>	<u>DirecciónC</u>	<u>Teléfono</u>
123A	Ana Almansa	c/ Argentaes	Profesor	20.000	Informática	Complutense	123
456B	Bernardo Botín	c/ Barcelona	Administrativo	15.000	Matemáticas	Complutense	456
012D	David Díaz	c/ Daroca	Ayudante	10.000	Informática	Complutense	123
789C	Carlos Crespo	c/ Cruz	Catedrático	30.000	CC. Empresariales	Somosaguas	789
123A	Bernardo Botín	c/ Argentaes	Profesor	20.000	Informática	Complutense	123
123A	David Díaz	c/ Argentaes	Profesor	20.000	Informática	Complutense	123
456B	Ana Almansa	c/ Barcelona	Administrativo	15.000	Matemáticas	Complutense	456
456B	David Díaz	c/ Barcelona	Administrativo	15.000	Matemáticas	Complutense	456
012D	Ana Almansa	c/ Daroca	Ayudante	10.000	Informática	Complutense	123
012D	Bernardo Botín	c/ Daroca	Ayudante	10.000	Informática	Complutense	123

Explicación de la Obtención de datos incorrectos

- ◆ El atributo que relaciona las tablas originales es DirecciónC
 - que **no** es clave de ninguna de las relaciones.
- ◆ *Conclusión*
 - se deben dividir las tablas de forma que
 - queden relacionadas por atributos clave
- ◆ Es una *conclusión informal* que se formaliza al estudiar
 - la propiedad de reunión no aditiva (no añadir filas falsas)
 - que asegura recuperar exactamente la información original
 - al realizar consultas uniendo las tablas divididas

Descomposición Correcta del Ejemplo 1

<i>Empleados</i>					
<u>Id empleado</u>	NombreE	DirecciónE	Puesto	Salario	Centro
123A	Ana Almansa	c/ Argentaes	Profesor	20.000	Informática
456B	Bernardo Botín	c/ Barcelona	Administrativo	15.000	Matemáticas
789C	Carlos Crespo	c/ Cruz	Catedrático	30.000	CC. Empresariales
012D	David Díaz	c/ Daroca	Ayudante	10.000	Informática

<i>Centros</i>		
<u>NombreC</u>	DirecciónC	Teléfono
Informática	Complutense	123
Matemáticas	Complutense	456
CC. Empresariales	Somosaguas	789

- ◆ cada atributo depende sólo de la clave primaria, y
- ◆ no hay redundancias (cada atributo se encuentra en un único lugar)
- ◆ Se dice que este diseño está en *3ª forma normal* (ya veremos porqué)

Propiedades Deseables de descomposiciones Correctas

- ◆ La propiedad de conservación de dependencias,
 - que asegura que las DFs originales se mantienen en
 - algún esquema de relación después de la descomposición

- ◆ La propiedad de reunión (join) no aditiva (o sin pérdida)
 - que evita el problema de la generación de tuplas incorrectas

- ◆ ¿Cómo se consiguen? Aplicando los conceptos de
 - Dependencias Funcionales (DFs) y Multivaloradas (DMs)
 - Formas Normales y
 - Normalización

- ◆ ¿Porqué Descomponer? : para arreglar Anomalías (Normalizar)

Definición de Dependencias Funcionales (DFs)

- ◆ Las DFs son restricciones al diseño:
 - *normas a cumplir por los datos de la BD*
 - ◆ Las DF's definen qué tuplas (filas) son legales respecto a una rel. R.
 - y, por lo tanto, dice las tuplas o filas que están prohibidas
 - ◆ Siendo tanto J como M: uno o más atributos, tenemos la . . .
 - ◆ *Definición Informal de **DF***: M depende funcionalmente de J, $J \rightarrow M$
 - Si para todas las filas que J tiene un valor concreto, todas esas filas, debe tener en M un único valor.
 - También M depende funcionalmente de J cuando: (**J será PK**)
 - Solo puede haber una fila con cada valor distinto en J
 - ◆ *Aunque J igual tiene un valor único de M en varias filas, en otras filas diferentes, mismo M puede tener varios valores de J*
- “debe tener” se determina por el **significado** de esos atributos en la BD
- Es decir, la relación semántica de los conceptos del dominio de la BD

Ejemplo de Dependencias Funcionales (DFs)

- ◆ EJ: Esquema **R (A, B, C,D)**

- ◆ Tenemos las siguientes filas válidas:

- 1. (a2, b1, c2, d1)

- 2. (a2, b2, c2, d2)

- 3. (a3, b3, c2, d3)

- C depende funcionalmente de A, $A \rightarrow C$:

- Se cumple porque todas las filas con valor a2 tienen mismo valor c2

- Aunque en 3. tengamos un valor nuevo a3 , solo hay una fila, así que vale c2

- pero **NO** se cumple $C \rightarrow A$ por la 3:

- no todas las filas con valor c2 tienen el mismo valor a2 (la 3. tiene a3)

- Si añadido 4. (a3, b3, c3, d4) dejaría de cumplirse $A \rightarrow C$, (4. está prohibida)

- porque hay dos filas con valor a3 que tienen distinto valor C : c2 y c3

- Por otra parte D tiene valores diferentes en cada fila, luego: $D \rightarrow A,B,C$

Concepto Necesarios Basados en DFs

- ◆ Un cjto de atributos es **Clave Candidata(CC)** si el resto de atrib de la tabla
 - “dependen funcionalmente” de ellos:
 - Es decir que existe una DF así: $CC \rightarrow \text{Resto Atrib}$
 - y, si se quita un atributo de la CC, deja de cumplir la condición
- ◆ La **Clave Primaria (CP o PK)** es cualquiera de las CC
- ◆ Una **superclave (SC)** es una CC con atributos añadidos
 - La CC es una SC que tiene el mínimo num. de atrib para ser CC
- ◆ El diseñador *decide* las DF semánticamente obvias pero
 - hay otras que se cumplen también:
 - las **DFs inferidas** a partir de las otras
- ◆ **EJ:** ¿Que DF's hay?
 - COCHES(matricula,Nbastidor, DNIComprador, Color)

Dependencias Multivaloradas (DMs) en Normalización

- ◆ Una **DM** obliga a que haya unas filas en la tabla que repiten información:
 - La **DF** prohíbe unas filas en la tabla (excluye filas ilegales)
- ◆ *Definición Informal de DM:*
 - Dado esquema $R(X Y \text{ Resto})$, donde X , Y y Resto son uno o más atributos
 - Existe la DM, $X \twoheadrightarrow Y$ si para un valor fijo de X ,
 - el conjunto de valores que corresponden a Y
 - ◆ son independientes de los valores que corresponden para *Resto*.
 - Deben existir los atributos *Resto*
- ◆ EJ: Esquema Biblioteca $R(\text{Tit}, \text{Aut}, \text{NumEd})$
 - Qué pasa a lo largo del tiempo?

Dependencias Multivaloradas (DMs) para Normalización

- ◆ EJ: Esquema **Biblioteca R** (Tit, Aut, NumEd)

- ◆ Un libro tiene título, varios autores y, con el tiempo, va teniendo más ediciones

- ◆ Qué pasa si se necesita hacer a lo largo del tiempo lo siguiente:

- a - Libro 'Titx' es escrito por Aut1, Aut2, Aut3, es la 1ª edición : se crean filas

- Titx, Aut1, 1

- Titx, Aut2, 1

- Titx, Aut3, 1

- b - Se publica el mismo libro en la 2ª edición, NumEd = 2 : se crean las filas

- problema: obliga a añadir tres filas (y repetir valores)

- Titx, Aut1, 2

- Titx, Aut2, 2

- Titx, Aut3, 2

- c - y Si necesitamos poner un nuevo Autor4 ?

→ Para crear unas tuplas nos obliga a crear otras

Dependencias Multivaloradas (DMs) para Normalización

(CONT.) EJ: Esquema Biblioteca R (Tit, Aut, NumEd)

◆ COMO ENCONTRAR LAS DM?

- 1) Encuentra que atributos son multivalorados? Aut y NumEd
- 2) Determina si esos atributos son independientes entre sí?
 - SI, Aut y NumEd no dependen entre ellos ... entonces si hay DM.

- ## ◆ Cual es la DM? Tit \twoheadrightarrow Aut donde: X es Tit, Y es Aut, Resto es NumEd
- por los corolarios también se cumple Tit \twoheadrightarrow NumEd

Cierres de DFs y de Atributos

- ◆ DEF - Cierre de un cnjto de dfs S : es el cnjto de las dfs y todas las posibles DFs que se puedan inferir de ellas, denominado S^+ .
 - Se hace con el cierre de atributos, así:
- ◆ DEF - Cierre de cnjtos de atributos. Dado un cnjto de atrib. X, y un cnjto de dfs S en R, el cierre de X en S, es X^+ :

X^+ es el cnjto de atrib. que son Dependientes. Func. de X.

¿Para qué vale?:

- descubrir claves primarias y candidatas
 - Aplicado a cada lado izq de las DFs nos da todas las DFs posibles
- Comprobar Propiedades de Descomposición de Tablas
- Descubrir en qué Forma Normal está una BD.

Primer paso: Algoritmo para obtener el cierre X^+

◆ Algoritmo: Cierre de un cnjto X de atributos

→ para cada df: se añade el lado dcho al resultado, solo
 si el izq ya estaba en el resultado

```
result  := X;                (! son dos bucles !!!)
while (hay cambios en result) do
    for each dep.func.  $Y \rightarrow Z$  in F do
        begin
            if Y está en result then
                result := result  $\cup$  Z;
            end
```

◆ **EJ:** Dada la rel $R = (A, B, C, G, H, I, J)$

 y las dfs : $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, AI \rightarrow J, B \rightarrow HG\}$

Se pide encontrar una CC:

 → busco un atributo (o más, juntos) del que dependan el resto de atrib.:

 → haciendo cierre del lado izq de las DFs

- para A, tiene los atrib, $(A)^+ = A B C G H I J$

Pasos para Mejorar el Diseño de la BD

- 1) Diagnosticar lo que va bien y mal:
 - I. Descubrir *todas* las DFs y DMs
 - II. Descubrir *qué* DFs y DMs pueden causar problemas
 - III. Qué nivel de calidad tiene la BD: en qué Forma Normal está
 - Los diseños prácticos exigen al menos 3FN
- 2) Reparar lo que no va bien
 - Eliminar las DFs problemáticas con **Normalización**
 - Dividiendo las tablas con DFs o DMs problemáticas
- 3) Refinar para mejorar Rendimiento con **Desnormalización**
 - Unir ciertas tablas y controlar por programa las DFs problemáticas
 - . . . para agilizar ciertas consultas frecuentes

Calidad basada en Formas normales, Normalización

- ◆ La forma normal de una TABLA se refiere a
 - La forma normal más exigente que satisface dicha tabla
 - Representa el grado o nivel hasta donde se ha **normalizado**
- ◆ La forma normal de una BD se refiere a
 - La forma normal más exigente que satisfacen
 - todas sus tablas
- ◆ Las formas normales más habituales, por orden ascendente de exigencia de las propiedades deseadas, son:
 - Primera (1FN)
 - Segunda (2FN)
 - Tercera (3FN)
 - Boyce/Codd (FNBC)
 - Cuarta (4FN)
- ◆ En general, los diseños **prácticos exigen al menos 3FN**

Primera forma normal

- ◆ Un Tabla está en 1ª FN si cumple estos requisitos:
 - Los dominios de sus atributos sólo pueden ser atómicos
 - Así se evitan atributos multivalorados y compuestos

- ◆ Actualmente se considera como
 - parte de la definición formal del Modelo Relacional y de SQL
 - Así que los ejemplos son en el paso de D. E/R al M. Relacional

Primera forma normal

Ejemplo 2: No está en 1ª FN

- ◆ Si se asume que en la entidad Centros,
 - un centro puede tener más de un teléfono,
 - podríamos tener una representación de la figura
 - esto supondría el uso del atributo **multivalorado** Teléfonos

<i>Centros</i>		
<u>NombreC</u>	DirecciónC	Teléfonos
Informática	Complutense	{123, 321, 213}
Matemáticas	Complutense	{456}
CC. Empresariales	Somosaguas	{789, 987}

Primera forma normal

- ◆ Hay tres posibilidades de representar la entidad para
 - satisfacer la primera forma normal: ¿Cuál es la mejor?
- ◆ **Solución 1:** Eliminar el atributo Teléfonos y
 - crear una nueva relación que asocie en cada fila un centro con un teléfono
- ◆ *Consecuencias:*
 - La clave de la 1ª relación debe formar parte de clave de la 2ª relación
 - Inconveniente: añadir una nueva relación a la BD y redundancia
 - Suceden anomalías cuando
 - se borra un centro y olvidamos borrar los teléfonos asociados
 - La integridad referencial (FK) asegura evitar estas anomalías
 - El NombreC de Teléfonos se hace FK con apunte a NombreC de Centros

Primera forma normal

Solución 1. Diseño en primera forma normal

<i>Centros</i>	
<u>NombreC</u>	DirecciónC
Informática	Complutense
Matemáticas	Complutense
CC. Empresariales	Somosaguas

<i>Teléfonos</i>	
<u>NombreC</u>	<u>Teléfono</u>
Informática	123
Informática	321
Informática	213
Matemáticas	456
CC. Empresariales	789
CC. Empresariales	987

Primera forma normal

Solución 2: Ampliar la clave de la relación de manera que incluya al atributo multivalorado.

◆ *Consecuencias:*

- Inconveniente: añade redundancia que provoca anomalías

<i>Centros</i>		
<u>NombreC</u>	<u>DirecciónC</u>	<u>Teléfono</u>
Informática	Complutense	123
Informática	Complutense	321
Informática	Complutense	213
Matemáticas	Complutense	456
CC. Empresariales	Somosaguas	789
CC. Empresariales	Somosaguas	987

Primera forma normal

Solución 3:

- ◆ Si se conoce la cardinalidad máxima del atributo multivalorado
 - se pueden crear tantas columnas como la cardinalidad máxima
- ◆ *Consecuencias:*
 - Inconveniente : uso de valores Null

<u>NombreC</u>	DirecciónC	Teléfono1	Teléfono2	Teléfono3
Informática	Complutense	123	321	213
Matemáticas	Complutense	456	Null	Null
CC. Empresariales	Somosaguas	789	987	Null

- ¿Qué solución es la mejor?: La 1ª , porque controla las redundancias

Primera forma normal

- ◆ Si el atributo multivalorado es compuesto,
 - representar varias direcciones para un empleado:
 - Empleados(Id_empleado, NombreP,
 {Direcciones(Calle, Ciudad, CódigoPostal)}))
- ◆ Esta relación se puede descomponer en dos:
 - Empleados(Id_empleado, NombreP)
 - DireccionesP(Id_empleado, Calle, Ciudad, CódigoPostal)

Primera forma normal

- ◆ Este procedimiento de desanidamiento se puede aplicar
 - recursivamente a cualquier relación con atributos multivalorados

- ◆ teniendo en cuenta que es necesario propagar
 - la clave de la relación original a la clave de la nueva relación
 - que contiene además la clave
 - que identifica unívocamente al atributo multivalorado.

Segunda forma normal

Ejemplo 3 *Personal_Proyectos* **NO** está en 2ª FN pero sí en 1ªFN

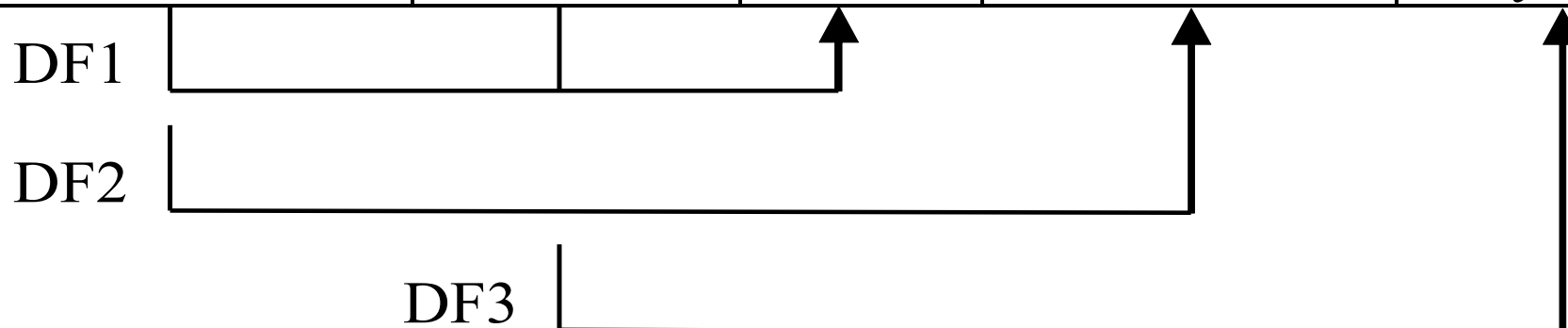
◆ **Problema:** Todos los atributos no dependen de la PK completa

- alguno solo de parte de ella (de algún atributo, no de todos)

◆ Existen *anomalías de actualización* causadas por DF2 y DF3

- Porque sus antecedentes no son clave completa de la tabla

<i>Personal_Proyectos</i>				
<u>Id empleado</u>	<u>NúmeroP</u>	Horas	NombreE	NombreP
123A	P-1	16	Ana Almansa	Proyecto 1
012D	P-1	8	David Díaz	Proyecto 1
012D	P-2	4	David Díaz	Proyecto 2



Segunda forma normal

- ◆ La 2ª FN evita este tipo de anomalías
- ◆ La 2ª FN se basa en el concepto de “*DF completa*”

Definición: *Dependencia funcional completa*

- ◆ La DF tipo $X \rightarrow Y$ es una *DF completa* si
 - no se cumple $X - \{A_i\} \rightarrow Y, A_i \in X$
 - Es decir, si quito algún atributo del antecedente
 - Deja de ser DF
- ◆ Si se cumple $X - \{A_i\} \rightarrow Y, A_i \in X$
 - se trata de una *DF parcial*

Segunda forma normal

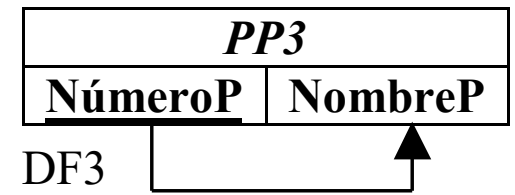
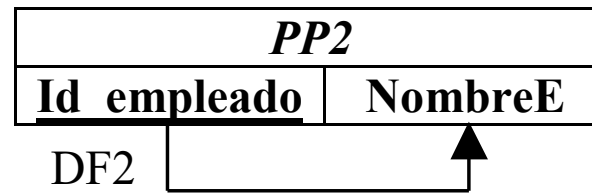
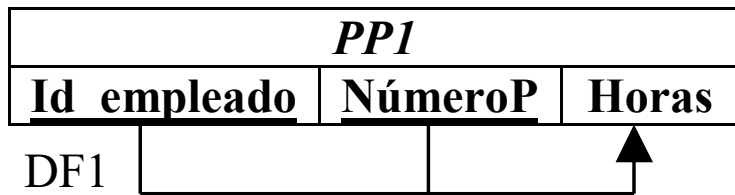
Definición: *Segunda forma normal*

- ◆ Una relación/tabla está en 2ª FN
 - si cada atributo que no forme parte de ninguna clave candidata (CC)
 - depende funcional y completamente de cada clave candidata
- ◆ Una relación/tabla que NO está en 2ª FN
 - Puede transformarse en varias tablas que sí lo estén
- ◆ **Solución** El procedimiento es dividir la tabla en
 - tantas nuevas tablas como DFs que *no* sean completas

Segunda forma normal

Ejemplo 3 , solución

◆ Así, el ejemplo anterior se traduce en:



Segunda forma normal

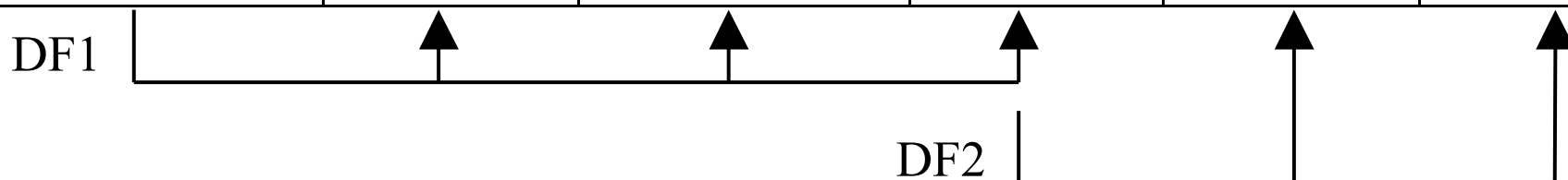
- ◆ este procedimiento asegura que el resultado está
 - al menos, en segunda forma normal
- ◆ En este caso concreto, por casualidad,
 - el resultado conseguido en este ejemplo se encuentra en 4FN (como se podrá comprobar más adelante)

Tercera forma normal

Ejemplo 4 existen anomalías de actualización causadas por la DF2 que queremos eliminar

- ◆ Sin embargo, este esquema está en 2ª FN porque
 - los dos últimos atributos, que son los que causan las anomalías,
 - dependen completa (*y transitivamente*)
 - ◆ del único atributo que forma la PK

<i>Empleados_Departamentos</i>					
<u>Id empleado</u>	NombreE	DirecciónE	CódigoD	NombreD	DirectorD
123A	Ana Almansa	<u>c/ Argentales</u>	DS	Sistemas	999Z
012D	David Díaz	<u>c/ Daroca</u>	DS	Sistemas	999Z



Tercera forma normal

- ◆ La 3ª FN se basa en el concepto de “*DF transitiva*”

Definición de *Dependencia funcional transitiva*

- ◆ Una DF del tipo de $X \rightarrow Y$ es una *DF transitiva* si
 - existe un conjunto de atributos Z que cumplen las tres:
 - juntos NO forman CC
 - no son subconjunto de ninguna clave (CC, PK)
 - y además se cumple $X \rightarrow Z$ y $Z \rightarrow Y$

Tercera forma normal

- ◆ En el ejemplo anterior, se cumple DF3
 - que es *transitiva* por DF1 y DF2

<i>Empleados_Departamentos</i>					
<u>Id empleado</u>	NombreE	DirecciónE	CódigoD	NombreD	DirectorD
123A	Ana Almansa	<u>c/ Argentaes</u>	DS	Sistemas	999Z
012D	David Díaz	<u>c/ Daroca</u>	DS	Sistemas	999Z

DF1: Id empleado → NombreE → DirecciónE → CódigoD → NombreD → DirectorD

DF2: CódigoD → NombreD → DirectorD

DF3: Id empleado → NombreE → DirecciónE → CódigoD → NombreD → DirectorD

Tercera forma normal

Definición: *Tercera forma normal (3ª FN)*

- ◆ Una tabla está en tercera forma normal si
 - satisface la segunda forma normal y
 - todos los atributos que **no** forman parte de una CC
 - **NO** dependen *transitivamente* de ninguna CC

Tercera forma normal

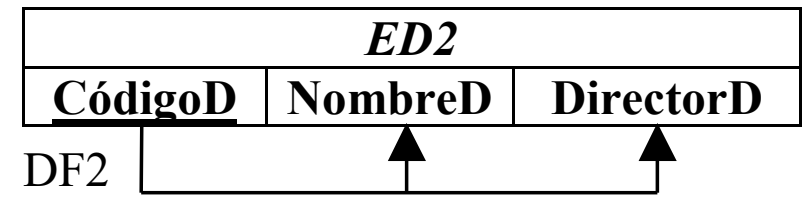
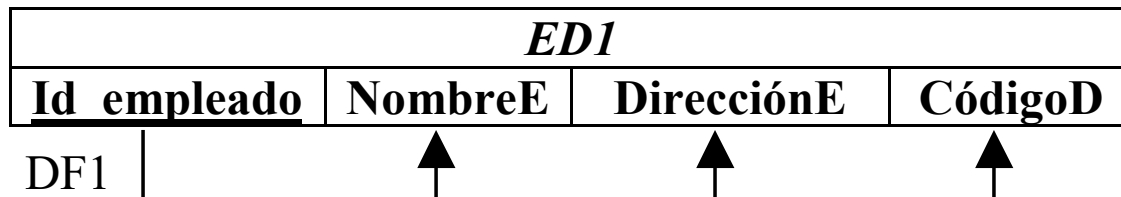
- ◆ Para facilitar la comprobación reformulamos la **definición**:
- ◆ Un esquema está en 3ª FN
 - con respecto a un conjunto de dependencias funcionales S
 - si para toda dependencia funcional no trivial $X \rightarrow Y$ de S^+
 - se cumple que
 - o bien X es superclave
 - o bien Y forma parte de una clave candidata (atributo primo)
- ◆ Así hay un test de 3ª FN más simple basado en el concepto
 - El **cierre** S^+ de un conjunto de dependencias funcionales S

Tercera forma normal

Para **normalizar** esta tabla:

- ◆ descomponerla en otra tabla con los atributos
- ◆ definidos por la DF responsable de la transitividad
- ◆ Dejando en la tabla original el antecedente de esa DF

Ejemplo 4 , solución:



Forma normal de Boyce-Codd (FNBC)

- ◆ La FNBC es más estricta que la reformulación de la 3FN
 - La FNBC evita otras redundancias que la 3FN no puede
 - Pero la FNBC no siempre es posible conseguirla

Definición: Forma normal de Boyce-Codd (FNBC)

- ◆ Un tabla está en la FNBC respecto a un conjunto **S** de DFs
 - si para toda DF no trivial $X \rightarrow Y$ de **S**⁺
 - se cumple que X es superclave

(quita la 2ª condición permitida en 3ªFN: prohíbe esas DFs)
- ◆ En particular, un esquema con dos atributos está en FNBC

Forma normal de Boyce-Codd

Ejemplo 5 Esta tabla está en 3FN, pero no en FNBC

- ◆ Contiene los investigadores participantes en proyectos,
 - que pueden ser codirigidos, (DF1)
 - pero los investigadores principales no dirigen más de un proy. (DF2)
- ◆ *Anomalía*: (que se debería poder evitar)
 - si no se vigila DF2 se podría añadir una fila de manera que
 - una persona fuese investigadora principal de dos proyectos
 - La vigilancia solo puede ser programando a mano

<i>Proyectos</i>		
<u>Investigador</u>	<u>Proyecto</u>	<u>IPrincipal</u>
111A	Proyecto 1	123A
222B	Proyecto 1	012D
333C	Proyecto 1	123A
444D	Proyecto 2	789C
444D	Proyecto 1	123A

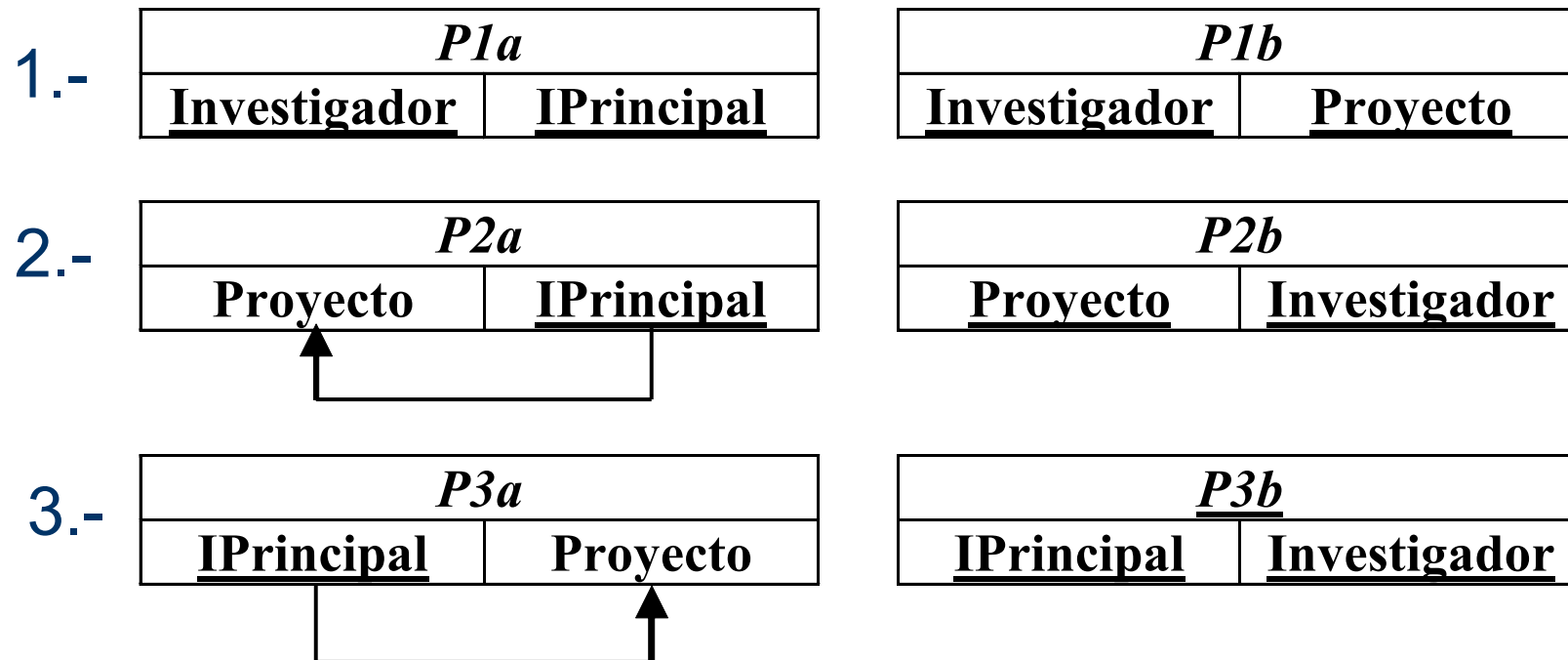
DF1

DF2

Forma normal de Boyce-Codd

Ejemplo 5 La descomposición no es inmediata, hay tres opciones

- 1.- **Problemas:** Pierde DF1 y DF2; y *Unión con pérdida (filas falsas)*
- 2.- **Problemas:** Pierde DF1 ; y *Unión con pérdida (filas falsas)*
- 3.- **Problemas:** Pierde DF1




→ Una DF es dentro de una sola tabla (contexto local)

Forma normal de Boyce-Codd

<i>P1a</i>	
<u>Investigador</u>	<u>IPrincipal</u>
111A	123A
222B	012D
333C	123A
444D	123A
444D	789C

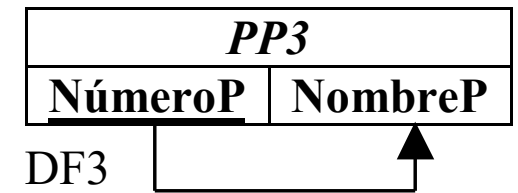
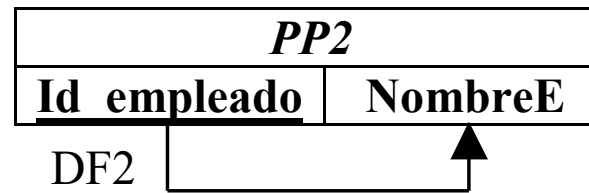
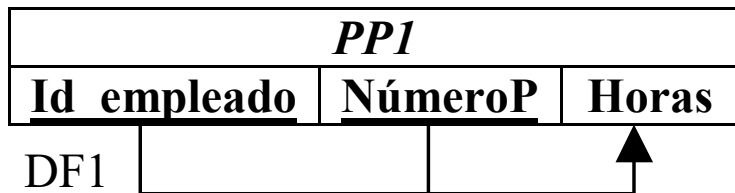
<i>P1b</i>	
<u>Investigador</u>	<u>Proyecto</u>
111A	Proyecto 1
222B	Proyecto 1
333C	Proyecto 1
444D	Proyecto 1
444D	Proyecto 2

<i>P1a</i>  <i>P1b</i>		
<u>Investigador</u>	<u>Proyecto</u>	<u>IPrincipal</u>
111A	Proyecto 1	123A
222B	Proyecto 1	012D
333C	Proyecto 1	123A
falsa → 444D	Proyecto 2	123A
444D	Proyecto 2	789C
444D	Proyecto 1	123A
falsa → 444D	Proyecto 1	789C

- ◆ Por ejemplo, en la primera descomposición se pierde la información de los investigadores principales de cada proyecto

Forma normal de Boyce-Codd

- ◆ Por tanto, ninguna de estas descomposiciones es aceptable.
- ◆ Pero hay situaciones donde la descomposición es correcta:
 - como la del ejemplo 3 de la 2ª FN
 - que queda en FNBC, y no pierde ninguna DF



- ◆ En la práctica, la mayoría de los esquemas
 - que están en 3NF lo están también en FNBC.

Dos propiedades a mantener en la descomposición

(como vimos) una descomposición debe asegurar dos propiedades:

- ◆ Que no se pierden la DFs
 - o bien, **asegurar que se cumplen** programando las validaciones
 - ◆ Re-uniones no aditivas o sin pérdida (**debe cumplirse sin excusa**)
 - *Sin pérdida* de información de integridad (mantenida por las claves)
 - *No aditividad*: no producir más tuplas (falsas) de las que estaban
 - en la relación antes de la descomposición
- Vemos a continuación la definición que caracteriza esta propiedad

Definición: Propiedad de reunión no aditiva

Dada una relación (o tabla) R, que tiene S (el cjto. de DFs):

- ◆ Una descomposición D de R , formada por las relaciones R_i
(representada por $D=\{R_1,...,R_m\}$)
- ◆ presenta reunión no aditiva respecto a S sobre R, si cumple que:
 - La reunión de todas las R_i obtiene las filas que había antes de hacer D

$$\bowtie (\pi_{R_1}(r), \dots, \pi_{R_m}(r)) = r$$

- Se debe cumplir
 - para todo estado r de la relación R que satisfaga S,
 - Donde “*todo estado r de la relación*” es conjunto de filas

→ *Para comprobar esto en la práctica necesitamos el concepto de CIERRE*

Comprobar reuniones no aditivas o sin pérdida

◆ RECORDAMOS

- DEF - Cierre de un cnjto de dfs S : es el cnjto de las DFs y todas las posibles DFs que se puedan inferir, denominado S^+ . El S^+ se hace con el cierre de cnjtos de atributos, así:
- DEF - Cierre de cnjtos de atributos. Dado un cnjto de atrib X, y un cnjto de dfs S en R, el cierre de X en S , es
 X^+ es el cnjto de atrib. que son Dependientes. Func. de X.

Comprobar si se cumple la *reunión no aditiva* (que no añade filas erróneas)

- ◆ una descomposición sencilla $D=\{R_1, R_2\}$ del esquema R, la cumple:
- ◆ . . . si y sólo si se cumple *alguna* de las dos condiciones siguientes:

$$\left((R_1 \cap R_2) \rightarrow (R_1 - R_2) \right) \in S^+ \quad , \text{ o bien}$$

$$\left((R_1 \cap R_2) \rightarrow (R_2 - R_1) \right) \in S^+$$

→ luego una de esas dos DFs se puedan deducir (pertenezcan al cierre S^+)

→ es decir: que los atributos comunes son PK en una de las relaciones

Conservación en la descomposición de: DFs y reuniones no aditivas

- ◆ En 3ªFN siempre se pueden conservar estas dos propiedades,
- ◆ En las FNBC y superiores, no se puede asegurar la conservación

Ejemplo 5: consigo la FNBC pero ¿conserva propiedades?

- ◆ El esquema Proyectos se encuentra en 3FN.
- ◆ Admite tres descomposiciones y ninguna de ellas conserva la DF1
- ◆ Sólo la última descomposición conserva reuniones no aditivas.
- ◆ Para comprobarlo verificamos que sí cumple la condición:
$$((R_1 \cap R_2) \rightarrow (R_1 - R_2)) = (\{\text{IPrincipal}\} \rightarrow \{\text{Proyecto}\}) \in S^+$$
- ◆ que de hecho se trata de la dependencia funcional DF2.

Otras formas normales

- ◆ La cuarta forma normal es una generalización de la forma normal de Boyce/Codd que se aplica a esquemas con DMs.
- ◆ Hay otras formas normales más exigentes que la 4FN.
- ◆ Las DMs representan restricciones que no se pueden con DFs
- ◆ Otros tipos de restricciones son las dependencias de reunión DRs
 - que generalizan las DMs y
 - que producen otra FN denominada forma normal proyección-reunión,
 - o también denominada quinta forma normal.

Normalización: Resumen

- ◆ Las formas normales buscan que las claves sean las protagonistas en el mantenimiento de la integridad, con el objetivo de que el diseño esté protegido por las claves.
- ◆ La idea es detectar las posibles fuentes de redundancias y anomalías y descomponer la relación.
- ◆ Hay dos propiedades que son deseables mantener en el proceso de normalización:
 - Preservar las dependencias funcionales.
 - Preservar las reuniones no aditivas.
- ◆ La segunda es de importancia fundamental, la primera no siempre se puede asegurar.
- ◆ El mejor diseño que siempre se puede alcanzar conservando ambas propiedades es la 3FN.
- ◆ En otros casos sólo a veces será posible alcanzar formas normales superiores conservando estas propiedades.

Cómo refinar el diseño orientándolo a las Prestaciones: Desnormalización

Una vez se tiene un modelo relacional de calidad en 3^aFN o mejor:

- ◆ Mejorar el rendimiento o prestaciones de Consultas importantes
 - asumiendo una pérdida de calidad de Diseño
- ◆ **Técnicas de Desnormalización** (también: *Denormalizar*)
 - Datos redundantes: atributos duplicados y derivados de otros
 - Crear otras relaciones que provocan ciclos
 - Fusión de tablas para evitar uniones
 - Partición de una tabla en varias:
 - horizontalmente o
 - verticalmente
 - Vectores de datos
 - Claves subrogadas

Cómo refinar el diseño orientándolo a las Prestaciones: Desnormalización

◆ Desnormalizar implica

- Dependencias entre las Consultas (procesos) y el diseño de los datos:
 - Si se cambia la consulta hay que cambiar el diseño
 - Por eso necesita un estudio previo serio.
- Mayor espacio ocupado por los datos redundantes
- Debe mantenerse la consistencia de los datos
 - por programa (aplicaciones, triggers, etc)
- Puede empeorar las actualizaciones

◆ Por eso debe justificarse el coste de dicho mantenimiento mediante una

- técnica de estimación del coste: Análisis de Redundancias
 - Sobre el Diagrama E/R por ser más gráfico

Técnicas de Desnormalización: Atributos Derivados

Atributos cuyos valores pueden ser derivados de

- otros de la misma entidad.

- Ejemplo: Entidad *Facturas* con atributos *Íntegro*, *Neto*, *IVA*.

◆ Atributos que se pueden derivar de

- otras entidades o relaciones

- generalmente con funciones de agregación.

◆ ¿Debo crear Atributos Derivados?

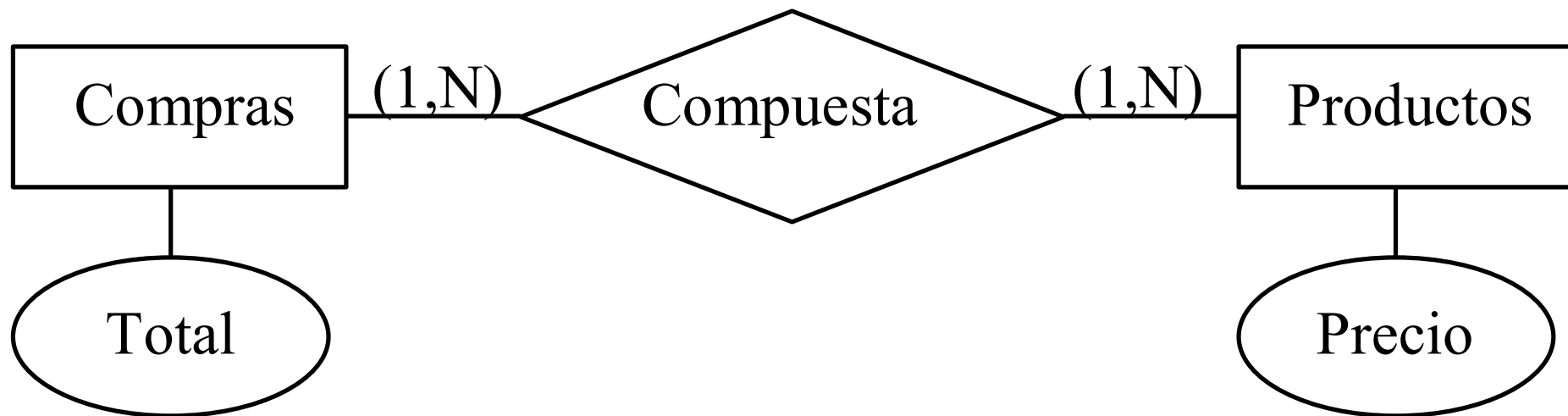
- Si son objetivo de alguna consulta frecuente

- Y la obtención de su valor es costosa porque
 - ◆ Se accede a muchas filas o tablas

◆ **Desventaja:** Hay que mantenerlo por programa, ej. un trigger

Técnicas de Desnormalización: atributo derivado

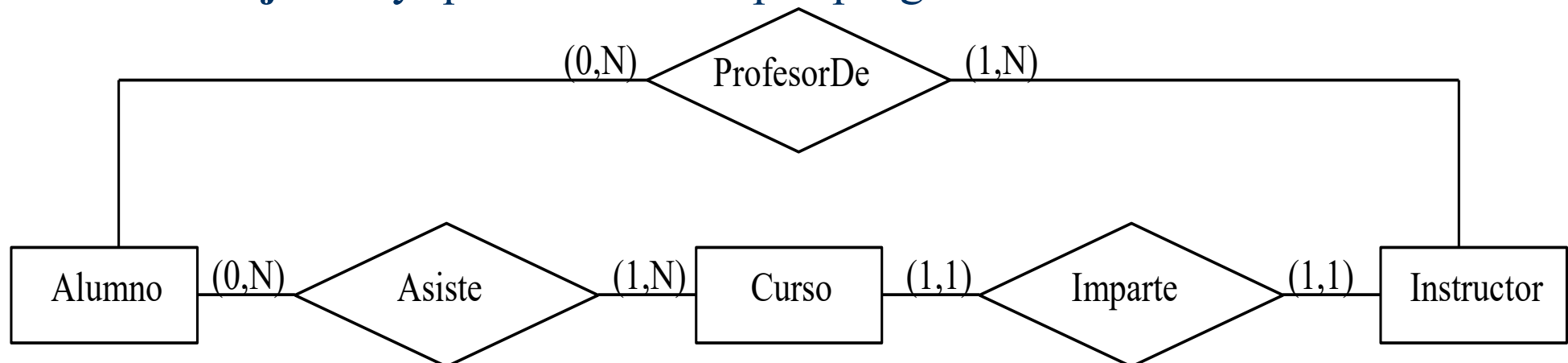
- ◆ El atributo Total se puede derivar del Precio de otra entidad



Técnicas de Desnormalización: Relaciones Derivadas

Relaciones derivadas de otras relaciones

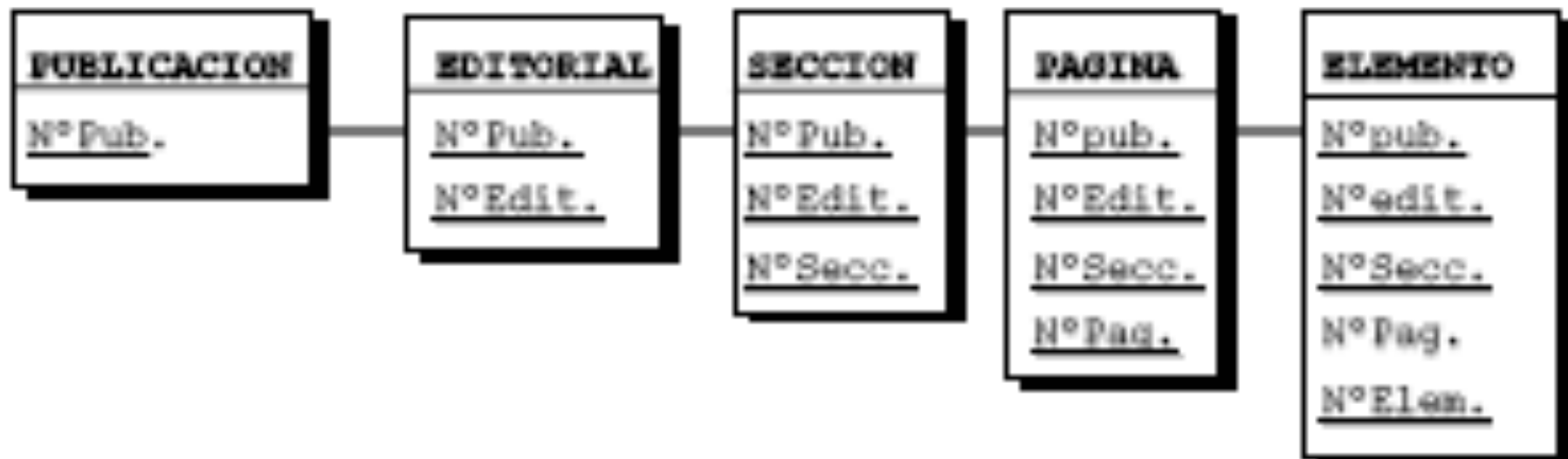
- Si hay ciclos
- ◆ Ejemplo: ProfesorDe se puede derivar del resto
 - Solo si la cardinalidad lo permite
- ◆ **¿Debo crear Relaciones Derivadas?**
 - Si agilizan alguna consulta frecuente porque sean un “atajo” en el E/R
 - Y la obtención camino original es costosa porque
 - ◆ Se accede a muchas tablas
- ◆ **Desventaja**: Hay que mantenerlo por programa: actualizaciones de tablas



Técnicas de Desnormalización: Claves Subrogadas evitan las claves ineficientes

Claves subrogadas:

- Son claves inventadas introducidas en el diseño, para sustituir la clave original en el caso de que ésta sea grande o ineficiente.
- ◆ Ejemplo: Inventamos una clave subrogada para sustituir
 - la clave de la tabla ELEMENTO muy larga, que viene dada por
 - la herencia de todas las claves de la jerarquía
- ◆ Se crean Si agilizan alguna consulta frecuente por esa clave
- ◆ **Desventaja:** se debe usar en las FKs que apunten a esa tabla



Técnicas de Desnormalización: Vector de datos sustituye atributo multivalorado

Vector de datos:

- Es un solo atributo o conjunto de atributos
- ... que se añaden a una tabla y
- que implementan un atributo multivalorado

Ejemplo:

◆ Dadas dos tablas:

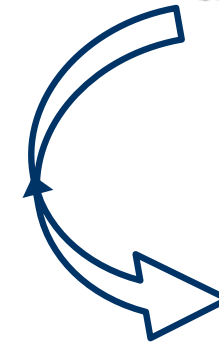
- socios y
- Pagos: cantidades pagadas mensualmente a lo largo de un año por los socios

Se crean si hay Consulta para optimizar:

- obtener la información de cada socio
- con las cantidades correspondientes a cada mes



Tablas normalizadas.



SOCIOS		
<u>N°socio</u>	nombre	
	Enero	
	Febrero	
	Marzo	
	Abril	
	Mayo	
	Junio	
	Julio	
	Agosto	
	Septiembre	
	Octubre	
	Noviembre	
	Diciembre	

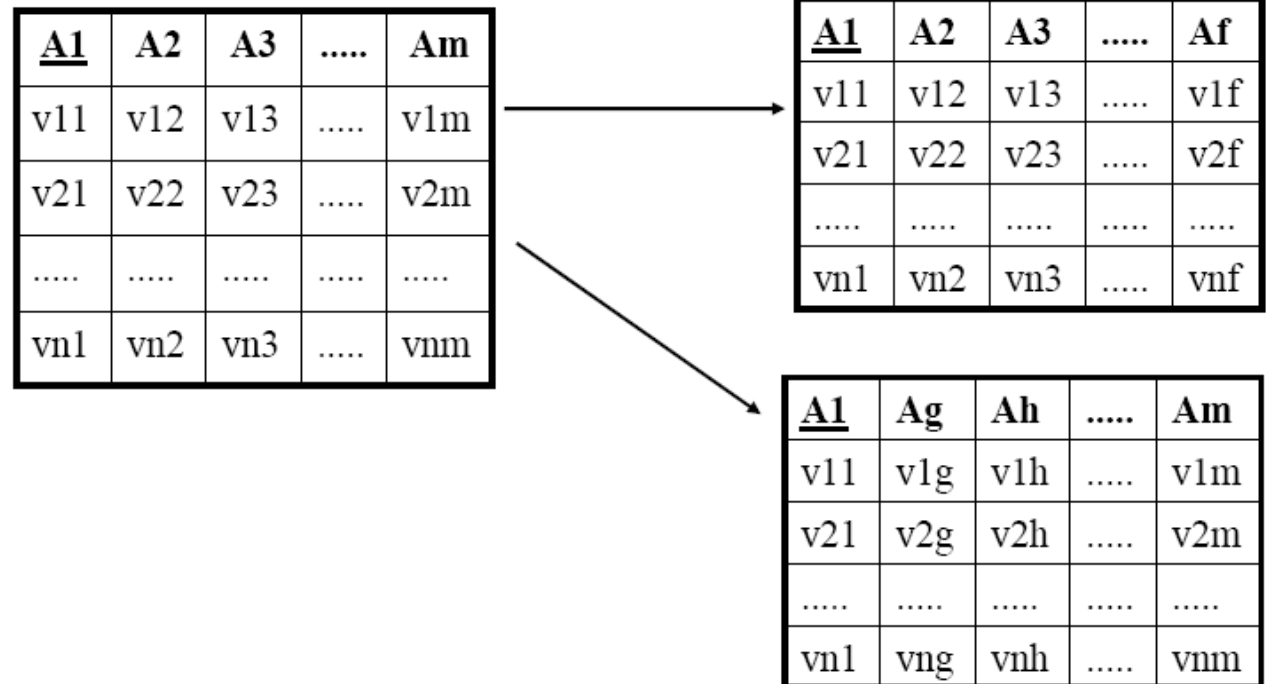
Técnicas de Desnormalización: Particionamiento Vertical

Particionamiento vertical de una Relación

- ◆ Se **crean** por eficiencia de consulta frecuente sobre *ciertos atributos*
- ◆ Reduce el número de atributos de la relación mediante
 - La división en dos relaciones: una con atributos que usa esa consulta
 - más la PK de la relación original
- ◆ El objetivo es delimitar conjuntos disjuntos de atributos incluidos en cada tipo de consulta

Ejemplo

- ◆ Mejorar dos consultas:
 - De A1 hasta Af
 - De Ag hasta Am



→ ORGANIZACIÓN
FÍSICA de DATOS

Técnicas de Desnormalización: Particionamiento Horizontal

Particionamiento horizontal de una relación

- ◆ Se **crean** por mejorar eficiencia de consulta frecuente sobre *ciertas filas*
- ◆ Divide en dos relaciones: una con las filas que usa esa consulta
- ◆ Se usan **técnicas de PLSQL dinámico** para la división y la consulta
- ◆ El objetivo es delimitar conjuntos disjuntos de tuplas afectados por cada tipo de consulta.

Ejemplo

- ◆ Mejorar dos consultas:
 - Filas de V1 hasta Vj
 - Filas de Vk hasta VN

a 1	a 2	a 3	...	a m
v 1 1	v 1 2	v 1 3	...	v 1 m
v 2 1	v 2 2	v 2 3	...	v 2 m
...
v n 1	v n 2	v n 3	...	v n m

<u>a 1</u>	a 2	a 3	...	a m
v 1 1	v 1 2	v 1 3	...	v 1 m
v 2 1	v 2 2	v 2 3	...	v 2 m
...
v j 1	v j 2	v j 3	...	v j m

<u>a 1</u>	a 2	a 3	...	a m
vk1	vk2	vk3	...	vk m
v1 1	v1 2	v1 3	...	v1 m
...
v n 1	v n 2	v n 3	...	v n m

→ **ORGANIZACIÓN**
FÍSICA de DATOS

Reestructuración de esquemas lógicos (E/R): Análisis de redundancias: ¿Porqué?

- ◆ *Ventajas* de la redundancia:
 - Reducción del número de accesos para completar las operaciones.
- ◆ *Inconvenientes*: Se necesitan
 - Mayor capacidad de almacenamiento y
 - operaciones adicionales para mantener la información extra actualizada.
- ◆ Decisión: *Qué Diseño conviene conservar?*
- ◆ Comparar el coste de las operaciones frecuentes en los diseños:
 - a) CASO 1 (sin redundancias)
 - Operación 1
 - Operación 2
 - b) CASO 2 (con redundancias)
 - Operación 1
 - Operación 2

Análisis de redundancias: Datos y Pasos

- ◆ Dando datos de Entrada específicos sobre los **criterios**:
 - 1.- Qué operaciones son problemáticas, y voy a estudiarlas.
 - 2.- Qué casos de diseño voy a estudiar
 - 3.- Estadísticas necesarias de cada operación
 - Tabla de *Volumen de datos* de cada relación/entidad que participa
 - Tabla de *Frecuencia al día* de cada operación
- ◆ Y otros datos a considerar, cómo que se decide que ...
 - Número de Escrituras de la operación (en qué tablas)
coste Escritura = 2 x coste Lectura
 - Número de Lecturas de la operación (en qué tablas)
- ◆ ...Podemos hacer el **cálculo del Análisis** para **cada Caso de Diseño**
- ◆ Vamos a aplicarlo a un ejemplo:

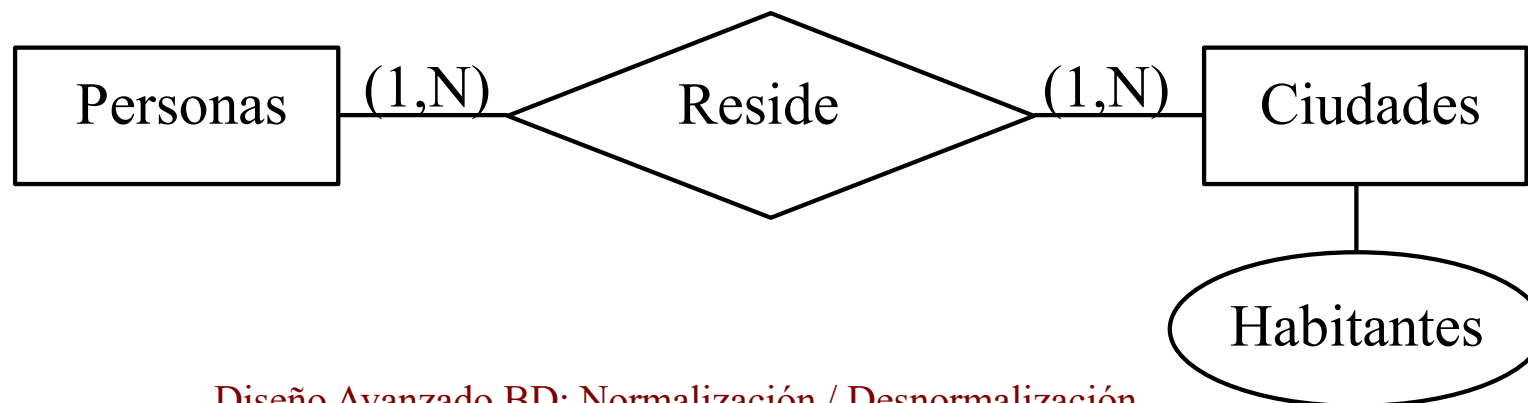
Análisis de redundancias

Ejemplo: dos posibilidades, dos Casos de Diseño de la Redundancia

- Redundancia a decidir: ¿Conservamos el atributo *Habitantes* o **no**?
- ¿Qué operación mejora? (operación problemática)
 - Consulta sobre el total de población por ciudad (Op2:imprimir)
- ¿Qué operación empeora? (operación problemática)
 - Actualizaciones de Personas (Op1: Crear una persona nueva)

◆ OBJETIVO: Determinar

¿Cuánto mejora la lectura frente a cuanto empeora la actualización?



Análisis de redundancias

1.- Con las operaciones problemáticas:

- ◆ Operación 1: Crear una nueva persona con su ciudad de residencia.
- ◆ Operación 2: Imprimir los datos de una ciudad.

2.- Qué casos de diseño voy a estudiar (los vemos después)

Análisis de redundancias

3.- Estadísticas necesarias de cada operación

◆ Tabla de volúmenes (filas) de datos

Concepto	Tipo	Volumen
Ciudades	E	200
Personas	E	1000000
Reside	R	1000000

Análisis de redundancias

3.- Estadísticas necesarias de cada operación

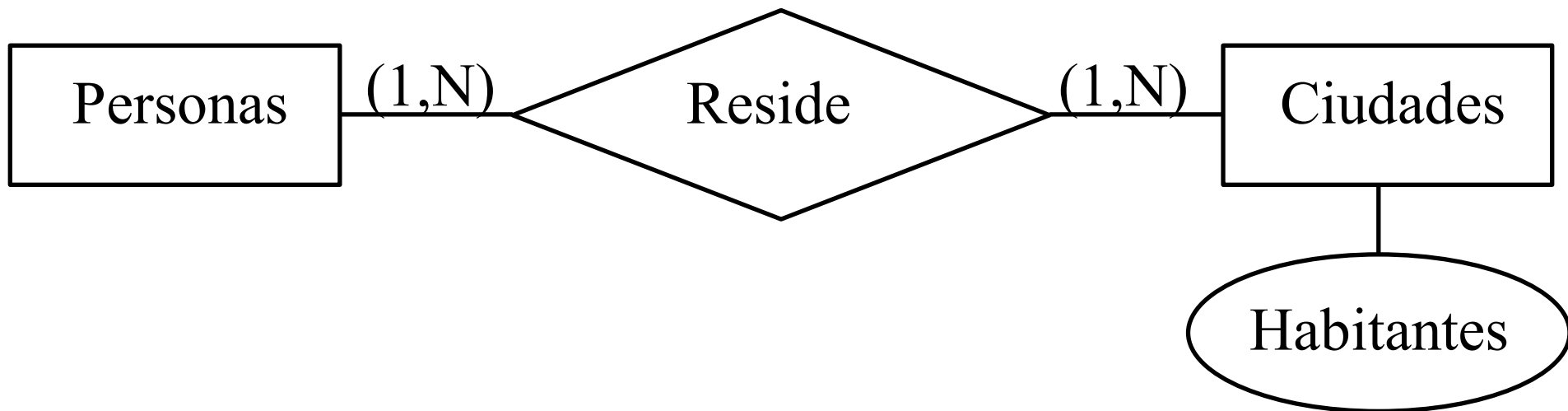
◆ Tabla de Frecuencias al día de las operaciones

Operación	Tipo	Frecuencia
Operación 1	I	500
Operación 2	I	2

◆ Leyenda: I=Interactivo, L=Procesamiento por lotes

Análisis de redundancias

A) Primer Caso de Diseño a Estudiar(2.-): con redundancia.



Análisis de redundancias

- ◆ (A) Determinar la Tabla de accesos para los cálculos
 - Siguiendo los accesos necesarios según la operación

Operación 1 crear persona		(ent/rela)		(lect/escr)	
Concepto	Tipo	Accesos		Tipo	
Personas	E	1		E	
Reside	R	1		E	
Ciudades	E	1		L	
Ciudades	E	1		E	
Operación 2 imprimir datos					
Ciudades	E	1		L	

Análisis de redundancias

◆ (A) Operación 1:

- Acceso de escritura a Personas,
- acceso de escritura a Reside,
- acceso de lectura a Ciudades y
- acceso en escritura a Ciudades (para actualizar el atributo).
- Total accesos por una operación: 3 de escritura + 1 lectura
- Frecuencia: 500 veces al día. Suma 1500 escrituras y 500 lecturas.

◆ (A) Operación 2:

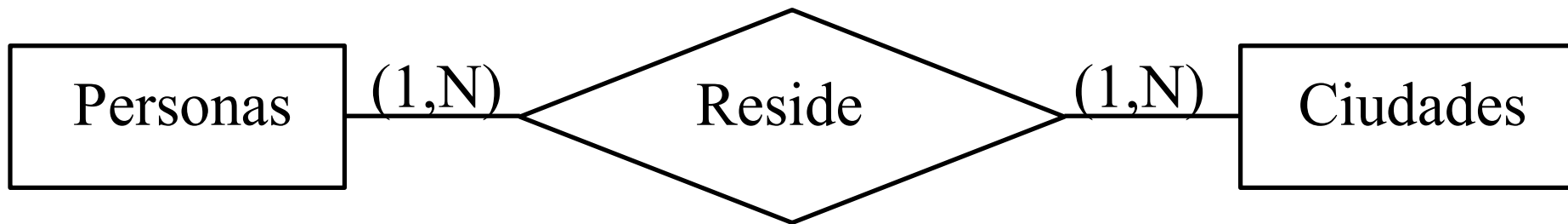
- Frecuencia: 1 veces/día
- El coste de esta operación es despreciable (200 lecturas = accesos).

◆ (A) Ambas operaciones: Si el coste de escritura es el doble que el de lectura

- ◆ $1500 \times 2 + 500 = 3500$ accesos.

Análisis de redundancias

B) Segundo Caso de Diseño a Estudiar (2.-) : sin redundancia.



Análisis de redundancias

◆ (B) Determina la Tabla de accesos según operaciones

◆ Operación 1			
		(enti/rela)	(Lectu/Escri)
Concepto	Tipo	Accesos	Tipo
Personas	E	1	E
Reside	R	1	E
Operación 2			
Ciudades	E	1	L
Reside	R	5000	L

◆ Leyenda: E=Entidad, R=Relación, L=Lectura, E=Escritura

Análisis de redundancias

◆ (B) Operación 1:

- Acceso de escritura a Personas,
- Acceso de escritura a Reside
- Total accesos : 2 escrituras
- Frecuencia: 500 veces al día
- Total accesos 1000 escrituras \Rightarrow 2.000 accesos

◆ (B) Operación 2:

- Acceso en lectura a Ciudades (despreciable)
- accesos de lectura a Reside
 - 5000 (aprox: número de personas/número de ciudades),
 - 2 veces al día=10.000 lecturas.

◆ (B) Ambas operaciones: Si el coste de escritura es el doble que el de lectura

- $1000 \times 2 + 10.000 = 12000$ accesos.

→ (CASO A : 3.500 , mejor con redundancia)

Análisis de redundancias

- ◆ Se puede decidir mantener la redundancia (denormalización),
- ◆ pese a sus problemas:
 - posibles inconsistencias (si no se mantiene)
 - mayor coste de las modificaciones
 - mayor dificultad de modificación del diseño en el futuro

