


Temario

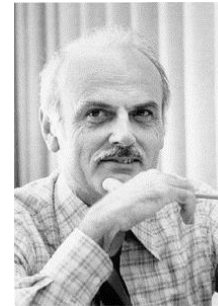
- ♦ Introducción y fundamentos
- ♦ Introducción a SQL
- ♦ Modelo Entidad / Relación
- ♦ Modelo relacional
- ♦ Diseño relacional: formas normales
- ♦ Consultas
 - Cálculo relacional
 - Álgebra relacional
- ♦ Implementación de bases de datos
 - Estructura física: campos y registros
 - Indexación
 - Índices simples
 - Árboles B
 - Hashing

Diseño de esquemas relacionales

- ♦ Igual que un programa C, un diseño BD puede ser sintácticamente correcto, pero de baja calidad
- ♦ Hay muchos criterios para valorar y procurar la calidad de un diseño
 - Claridad, facilidad de lectura, reflejo de las estructuras naturales del dominio
 - Eficiencia de consulta
 - Aprovechamiento del espacio en disco
 - Buena selección de claves, detalle en las restricciones
 - Facilidad de actualización y evolución del diseño
 - ...
 - Ciertas propiedades formales definibles en el modelo relacional 
- ♦ Propiedades formales
 - Evitar NULLs
 - Formas normales

Normalización de esquemas relacionales

- ♦ Criterios formales para valorar y mejorar el diseño de una base de datos
 - Reducir redundancias (consumen espacio y pueden generar problemas de consistencia)
 - Facilitar la actualización de datos de forma más modular (evitar anomalías)
 - Facilitar la evolución de los esquemas
 - Neutralidad respecto de las consultas
- ♦ Formas normales 1ª, 2ª y 3ª
 - E. Codd en 1970/71
- ♦ Forma normal Boyce-Codd (BCNF)
 - R. F. Boyce en 1974
- ♦ Formas normales 4ª, 5ª, 6ª
 - 1977/79/2002
 - Menos utilizadas



Edgard F. Codd

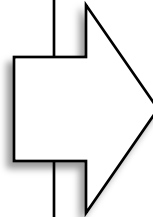
Anomalías de diseño

- ♦ No son un error en sí mismas
 - Si se hace bien la actualización no habría problema
- ♦ Pero... son un factor de error
 - Dan la ocasión de generar inconsistencias semánticas y otros problemas que el modelo relacional en sí no detectaría
- ♦ La normalización las evita y hace el diseño más robusto
 - Ahorra precauciones externas al diseño

Anomalías de actualización

1. Modificación del valor de un campo

<u>NIE</u>	Nombre	Teléfono	<u>Asignatura</u>
12345	Isabel	123456789	17824
12345	Isabel	123456789	17825
12345	Isabel	123456789	17826
12345	Isabel	123456789	17827
12345	Isabel	123456789	17828
67890	David	578234890	17824
67890	David	321654987	17825
67890	David	321654987	17826
67890	David	321654987	17827
⋮	⋮	⋮	⋮



Mejor diseño...

<u>NIE</u>	Nombre	Teléfono
12345	Isabel	123456789
67890	David	321654987
⋮	⋮	⋮

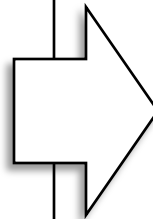
<u>NIE</u>	<u>Asignatura</u>
12345	17824
12345	17825
12345	17826
12345	17827
12345	17828
67890	17824
67890	17825
67890	17826
67890	17827
⋮	⋮

- ♦ Redundancia
- ♦ Problemas al modificar: estado inconsistente si no se actualizase el campo en todas las filas

Anomalías de actualización (cont)

2. Inserción

<u>NIE</u>	Nombre	Teléfono	<u>Asignatura</u>
12345	Isabel	123456789	17824
12345	Isabel	123456789	17825
12345	Isabel	123456789	17826
12345	Isabel	123456789	17827
12345	Isabel	123456789	17828
67890	David	321654987	17824
67890	David	321654987	17825
67890	David	321654987	17826
67890	David	321654987	17827
32154	María	987654321	NULL
⋮	⋮	⋮	⋮



Mejor diseño...

<u>NIE</u>	Nombre	Teléfono
12345	Isabel	123456789
67890	David	321654987
⋮	⋮	⋮

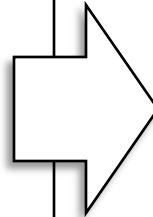
<u>NIE</u>	<u>Asignatura</u>
12345	17824
12345	17825
12345	17826
12345	17827
12345	17828
67890	17824
67890	17825
67890	17826
67890	17827
⋮	⋮

- ♦ Problema al insertar tuplas cuando aún no se saben todos sus campos –especialmente parte de la clave primaria

Anomalías de actualización (cont)

3. Eliminación

<u>NIE</u>	Nombre	Teléfono	<u>Asignatura</u>
12345	Isabel	123456789	17824
12345	Isabel	123456789	17825
12345	Isabel	123456789	17826
12345	Isabel	123456789	17827
12345	Isabel	123456789	17828
67890	David	321654987	17824
67890	David	321654987	17825
67890	David	321654987	17826
67890	David	321654987	17827
⋮	⋮	⋮	⋮



Mejor diseño...

<u>NIE</u>	Nombre	Teléfono
12345	Isabel	123456789
67890	David	321654987
⋮	⋮	⋮

<u>NIE</u>	<u>Asignatura</u>
12345	17824
12345	17825
12345	17826
12345	17827
12345	17828
67890	17824
67890	17825
67890	17826
67890	17827
⋮	⋮

- ♦ Al eliminar toda la matriculación de un estudiante se podrían perder todos los datos de éste

Anomalías de actualización (cont)

- ♦ Las inconsistencias típicamente afloran al hacer consultas
 - Un estudiante ya no aparece
 - Un estudiante con dos nºs de teléfono, o dos direcciones, dos nombres...
- ♦ Las anomalías de actualización se depuran generalmente descomponiendo esquemas
 - Pero no de cualquier manera: una descomposición inadecuada puede dar lugar a tuplas espurias

Matrícula

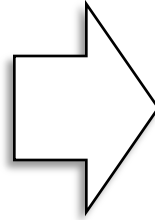
<u>NIE</u>	Nombre	Teléfono	<u>Asignatura</u>
12345	Isabel	123456789	17824
12345	Isabel	123456789	17825
67890	David	321654987	17826
67890	David	321654987	17827
89456	Isabel	755326284	17835
89456	Isabel	755326284	17838
⋮	⋮	⋮	⋮

Estudiante

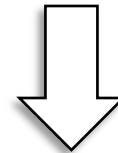
<u>NIE</u>	Nombre	Teléfono
12345	Isabel	123456789
67890	David	321654987
89456	Isabel	755326284
⋮	⋮	⋮

Matrícula

<u>Nombre</u>	<u>Asignatura</u>
Isabel	17824
Isabel	17825
David	17826
David	17827
Isabel	17835
Isabel	17838
⋮	⋮



Tuplas
espurias



```
SELECT NIE, Nombre, Asignatura
FROM Estudiante
NATURAL JOIN Matrícula
WHERE NIE = '12345'
```

Tuplas espurias



<u>NIE</u>	Nombre	<u>Asignatura</u>
12345	Isabel	17824
12345	Isabel	17825
12345	Isabel	17835
12345	Isabel	17838

Descomposición de esquemas relacionales

Una descomposición de un esquema R es un mapping $R \rightarrow \{R_1, \dots, R_n\}$

Es la base de la normalización (que ahora veremos...)

Es deseable que una descomposición cumpla las siguientes propiedades:

- ◆ Preservación de atributos

- Es decir $\bigcup_{i=1}^n R_i = R$

¿Qué son?

- ◆ Preservación de dependencias

- Sea F el conjunto de dependencias funcionales de R

- Sea $\pi_{R_i}(F)$ el subconjunto de dependencias de F que implican sólo atributos de R_i

- Se debe cumplir que F es equivalente a $\bigcup_{i=1}^n \pi_{R_i}(F)$ } *En breve...*

Veámoslo...

En otras palabras, la unión de las dependencias de las tablas R_i equivale a las dependencias de la tabla original

- ◆ Join sin pérdida (o join no aditivo –pérdida de información, no de tuplas)

- Para todo estado r de R se cumple $\pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \dots \bowtie \pi_{R_n}(r) = r$, donde:

$\pi_{R_i}(r)$ es la proyección de las tuplas de r a los atributos de R_i

\bowtie es el natural join

Operaciones de álgebra relacional (más adelante...)

En otras palabras: el join de las tablas R_i es la tabla original R

- Evita las tuplas espurias al hacer natural join

Dependencias funcionales

- ◆ Def: Dados dos conjuntos X e Y de atributos de un esquema R, Y depende funcionalmente de X si $t_1[X] = t_2[X] \Rightarrow t_1[Y] = t_2[Y], \forall t_1, t_2 \in r(R)$
Es decir, los atributos de Y están unívocamente determinados por los de X
- ◆ Notación: $X \rightarrow Y$
- ◆ Ejemplos
 - X es una superclave de R $\Rightarrow X \rightarrow Y$ para cualquier conjunto de atributos Y de R
 - Reserva (Nombre, Dni, Nvuelo, Origen, Destino, Hora, Fecha, Precio)
 - Dni \rightarrow Nombre /* No ponemos { } para abreviar */
 - Nvuelo \rightarrow { Origen, Destino, Hora }
 - { Dni, Nvuelo } \rightarrow { Nombre, Origen, Destino, Hora }
 - { Dni, Nvuelo, Fecha } \rightarrow { Nombre, Origen, Destino, Hora, Precio }
 - { Dni, Fecha, Hora } \rightarrow { Nombre, Nvuelo, Origen, Destino, Precio }

Formas normales

- ♦ Su cumplimiento reduce anomalías de actualización y mejora las propiedades del diseño (robustez, actualización, etc.)
- ♦ Son incrementales
 - Si se cumple la forma normal n-ésima se cumple la (n-1)-ésima
- ♦ Formas normales 1ª, 2ª, 3ª, BCNF
 - 2ª, 3ª y BCNF se definen en términos de dependencias funcionales
 - 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

1ª forma normal (1NF)

- ♦ Def: en un esquema 1NF...
 - 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
- ♦ Se considera parte inherente del modelo relacional
- ♦ Aunque...
 - 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

2ª forma normal (2NF)

- ♦ Def: Un esquema R es 2NF si todo atributo no primario de R tiene una dependencia funcional plena con las claves de R
- ♦ Def: Atributos primarios son los que forman parte de alguna clave
- ♦ Def: Una dependencia funcional $X \rightarrow Y$ es plena si no le sobra ningún atributo a X, es decir $X - \{A\} \not\rightarrow Y, \forall A \in X$
- ♦ Dicho de otro modo, los atributos dependen de la clave completa; sólo los atributos de una clave pueden depender de partes de una clave

2ª forma normal: ejemplo 1

Reserva (Nombre, Dni, Nvuelo, Origen, Destino, Hora, Fecha, Precio)

♦ Ejemplos

- Reserva ('Ana', 165467, 123, 'MAD', 'LAX', '16:25:00', '2011-10-24', 620)
- Reserva ('Luis', 165467, 123, 'MAD', 'LAX', '16:25:00', '2011-10-24', 620)
- Reserva ('Luis', 165467, 123, 'MAD', 'LAX', '16:25:00', '2011-11-18', 620)

♦ Claves...

- { Dni, Nvuelo, Fecha } /* Suponiendo que un pasajero sólo reserva */
- { Dni, Fecha, Hora } /* billetes que va a usar personalmente */

♦ Dependencias...

- Dni → Nombre
- Nvuelo → { Origen, Destino }
- Nvuelo → Hora /* Hora es un atributo primario */
- { Dni, Nvuelo, Fecha } → Precio
- { Dni, Fecha, Hora } → { Nvuelo, Origen, Destino, Precio }

← **No 2NF**

} **Ok**

2ª forma normal: ejemplo 2

Vuelo (Nvuelo, Origen, Destino, Hora)

- ♦ Claves: Nvuelo
- ♦ Dependencias: Nvuelo \rightarrow { Origen, Destino, Hora } } *Ok 2NF*

Pasajero (Dni, Nombre)

- ♦ Claves: Dni
- ♦ Dependencias: Dni \rightarrow Nombre } *Ok 2NF*

Reserva (Dni, Nvuelo, Fecha)

- ♦ Claves: { Dni, Nvuelo, Fecha }
- ♦ Dependencias: \emptyset } *Ok 2NF*

2ª forma normal: ejemplo 3

Vuelo (Nvuelo, Origen, Destino, Ciudad_origen, Ciudad_destino, Hora)

- ◆ Ejemplos

- Vuelo (123, 'CDG', 'LHR', 'París', 'Londres', '11:35:00')
- Vuelo (456, 'ORY', 'LGW', 'París', 'Londres', '15:20:00')

- ◆ Claves

- Nvuelo

- ◆ Dependencias

- Nvuelo → { Origen, Destino, Ciudad_origen, Ciudad_destino, Hora }
- Origen → Ciudad_origen
- Destino → Ciudad_destino

} Ok 2NF

2ª forma normal: ejemplo 4

Si todas las claves tienen un solo atributo, ¿el esquema es 2NF?

2ª forma normal: ejemplo 5

Dirección (Calle, Número, Piso, Municipio, Provincia, País, CP)

♦ Ejemplos

- Dirección ('Pza. Mayor', 2, 2, 'Oropesa', 'Castellón', 'España', 12594)
- Dirección ('Pza. Mayor', 2, 1, 'Oropesa', 'Castellón', 'España', 12594)
- Dirección ('Pza. Mayor', 2, 1, 'Oropesa', 'Toledo', 'España', 45687)

♦ Claves

- { Calle, Número, Piso, CP }
- { Calle, Número, Piso, Municipio, Provincia, País }

♦ Dependencias

- { Calle, Número, Municipio, Provincia, País } → CP
- CP → { Municipio, Provincia, País }

← Ok
← Ok? } Ok 2NF

CP, Municipio, Provincia, País son primarios

3ª forma normal (3NF)

$X \rightarrow A$ es trivial si $A \subset X$

- ♦ Def: Un esquema es 3NF si para toda dependencia $X \rightarrow A$ no trivial,
o bien X es una superclave, o bien A es un atributo primario
- ♦ 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: ejemplo 1

Dirección (Calle, Número, Piso, Municipio, Provincia, País, CP)

♦ Ejemplos

- Dirección ('Pza. Mayor', 2, 2, 'Oropesa', 'Castellón', 'España', 12594)
- Dirección ('Pza. Mayor', 2, 1, 'Oropesa', 'Castellón', 'España', 12594)
- Dirección ('Pza. Mayor', 2, 1, 'Oropesa', 'Toledo', 'España', 45687)

♦ Claves

- { Calle, Número, Piso, CP }
- { Calle, Número, Piso, Municipio, Provincia, País }

♦ Dependencias

- { Calle, Número, Municipio, Provincia, País } → CP
- CP → { Municipio, Provincia, País }

CP, Municipio, Provincia, País son primarios

} Ok 2NF } Ok 3NF

3ª forma normal: ejemplo 2

Vuelo (Nvuelo, Origen, Destino, Ciudad_origen, Ciudad_destino, Hora)

♦ Ejemplos

- Vuelo (123, 'CDG', 'LHR', 'París', 'Londres', '11:35:00')
- Vuelo (456, 'ORY', 'LGW', 'París', 'Londres', '15:20:00')

♦ Claves

- Nvuelo

♦ Dependencias

- Nvuelo → { Origen, Destino, Ciudad_origen, Ciudad_destino, Hora }

– Origen → Ciudad_origen

– Destino → Ciudad_destino

← **No 3NF**

} *Ok 2NF*

No superclaves

No primarios

3ª forma normal: ejemplo 3

Si todas las claves tienen un solo atributo, ¿el esquema es 3NF?

Forma normal Boyce-Codd (BCNF)

- ◆ Def: Un esquema R es BCNF si para toda dependencia $X \rightarrow Y$ no trivial X es una superclave de R
- ◆ Dicho de otro modo, no puede haber más dependencia que con las superclaves

Forma normal Boyce-Codd: ejemplo

Dirección (Calle, Número, Piso, Municipio, Provincia, País, CP)

♦ Ejemplos

- Dirección ('Pza. Mayor', 2, 2, 'Oropesa', 'Castellón', 'España', 12594)
- Dirección ('Pza. Mayor', 2, 1, 'Oropesa', 'Castellón', 'España', 12594)
- Dirección ('Pza. Mayor', 2, 1, 'Oropesa', 'Toledo', 'España', 45687)

♦ Claves

- { Calle, Número, Piso, CP }
- { Calle, Número, Piso, Municipio, Provincia, País }

♦ Dependencias

No superclave

– { Calle, Número, Municipio, Provincia, País } → CP

– CP → { Municipio, Provincia, País }

← **No BCNF**

CP, Municipio, Provincia, País son primarios

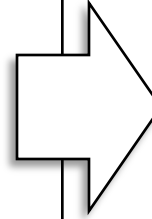
Ok 3NF

Forma normal Boyce-Codd (BCNF)

BCNF no elimina totalmente las anomalías de actualización → FN 4ª, 5ª, 6ª

<u>Marca</u>	<u>Medicamento</u>	<u>Indicaciones</u>
Gelocatil	Paracetamol	Fiebre
Gelocatil	Paracetamol	Resfriado
Gelocatil	Paracetamol	Cefalea
Tylenol	Paracetamol	Fiebre
Tylenol	Paracetamol	Resfriado
Tylenol	Paracetamol	Cefalea
Tylenol	Codeína	Tos
Tylenol	Codeína	Dolor

}
}
} Redundancias...



Mejor diseño...

<u>Marca</u>	<u>Medicamento</u>
Gelocatil	Paracetamol
Tylenol	Paracetamol
Tylenol	Codeína

<u>Medicamento</u>	<u>Indicaciones</u>
Paracetamol	Fiebre
Paracetamol	Resfriado
Paracetamol	Cefalea
Codeína	Tos
Codeína	Dolor

- El esquema es BCNF
- Pero contiene redundancias
- Anomalías de actualización: modificar / añadir / eliminar indicaciones de un medicamento, etc.

Un último ejemplo...

¿Cuál es la mínima forma normal de un esquema en el que la combinación de todos sus atributos forma una clave?

Resumen

Dada $X \rightarrow Y...$		X superclave		
		Sí	No	
			X primario	
			Sí	No
Y primario	Sí	BCNF	3NF	
	No		1NF	2NF

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

3NF (R, F)

¿Qué es?

$D := \emptyset$



$G :=$ cobertura mínima de F

for $X \rightarrow Y \in G$

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

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

Si ningún esquema de D contiene una clave de R, añadir a D un esquema con una clave de R

Eliminar los esquemas redundantes de D (esquemas incluidos en otros)

La descomposición tiene join sin pérdida,
y preserva las dependencias

Normalización 3NF (cont)

En otras palabras...

1. Partiendo de una cobertura mínima
2. Sacar a tablas aparte los atributos de todos los conjuntos de dependencias con la misma “parte izquierda”
3. Si no ha salido ninguna tabla con la clave original completa, crear esa tabla
4. Eliminar esquemas redundantes

Normalización BCNF

BCNF (R, F)

$D := \{R\}$

while D contiene una relación no BCNF

$Q :=$ elegir una relación no BCNF en D

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

 Substituir Q en D por $(Q - Y), (X \cup Y)$

El algoritmo genera una descomposición que tiene join sin pérdida, pero no asegura la preservación de todas las dependencias

Normalización BCNF (cont)

En otras palabras...

Ahí se pueden perder dependencias

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

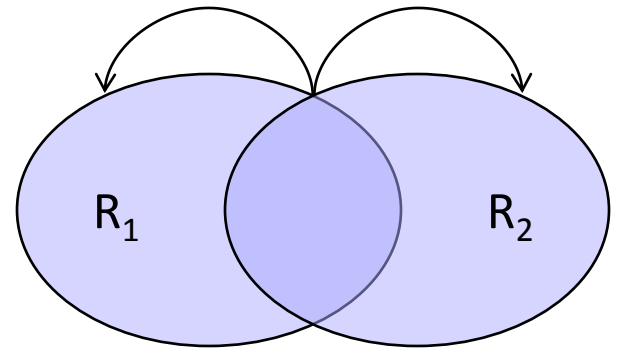
Join sin pérdida


Test sencillo:

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

O bien $(R_1 \cap R_2 \rightarrow R_1 - R_2)$ se infiere de F

O bien $(R_1 \cap R_2 \rightarrow R_2 - R_1)$ se infiere de F




¿Qué quiere decir?

Esto viene a decir que $R_1 \cap R_2$ actúa como clave foránea de R_1 a R_2 , o viceversa

Cobertura mínima

- ♦ Def: Una **cobertura mínima** de un conjunto de dependencias F es un conjunto mínimo equivalente a F
- ♦ Def: Un conjunto de dependencias 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 derecha de una dependencia, obtenemos un conjunto no equivalente a F

En otras palabras: dependencias en forma canónica y sin redundancias

- ♦ Def: Dos conjuntos de dependencias son equivalentes si toda dependencia de uno se puede **inferir** de las dependencias del otro y viceversa

Inferencia entre dependencias

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

Aumento: $X \rightarrow Y \Rightarrow XZ \rightarrow YZ$

Transitividad: $\{ X \rightarrow Y, Y \rightarrow Z \} \Rightarrow X \rightarrow Z$

Proyección: $X \rightarrow YZ \Rightarrow X \rightarrow Y$ (o $X \rightarrow Z$)

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

Pseudotransitividad: $\{ X \rightarrow Y, WY \rightarrow Z \} \Rightarrow WX \rightarrow Z$

Obsérvese que: $\{ X \rightarrow Y, Z \rightarrow W \} \Rightarrow XZ \rightarrow YW$

$X \rightarrow Z \Rightarrow XY \rightarrow Z$

Reglas (“axiomas”) de Armstrong, completas

Cobertura mínima (cont)

Cobertura mínima (E)

$F := E$

Substituir todas las dependencias $X \rightarrow \{Y_1, \dots, Y_n\}$ en F
por $X \rightarrow Y_1, \dots, X \rightarrow Y_n$

for $X \rightarrow Y \in F$ do

for $A \in X$ do

if $F - \{X \rightarrow Y\} \cup \{(X - \{A\}) \rightarrow Y\}$ es equivalente a F

then $F := F - \{X \rightarrow Y\} \cup \{(X - \{A\}) \rightarrow Y\}$

for $X \rightarrow Y \in F$ do

if $F - \{X \rightarrow Y\}$ es equivalente a F

then $F := F - \{X \rightarrow Y\}$

return F

Cobertura mínima (cont)

En otras palabras...

1. Descomponer en dependencias sobre atributos individuales en la parte derecha
2. Eliminar atributos que sobren en las partes izquierdas
3. Eliminar dependencias que se infieren de otras

Otros criterios de diseño

Además de la normalización... (motivada por las anomalías de actualización)

- ◆ Evitar valores NULL
 - Crean problemas en operaciones que implican comparaciones, conteos o sumas
 - La proporción de NULLs en un atributo puede ser un criterio para sacar el atributo a una relación aparte
- ◆ Semántica de los esquemas
 - La facilidad con que pueden explicarse es una medida informal de la calidad del diseño
 - P.e. un esquema que junta varias entidades del mundo real puede ser más confuso
- ◆ Eficiencia y denormalización
 - Tablas no normalizadas pueden ser más eficientes para algunas consultas
 - Por motivos de eficiencia en ocasiones compensa juntar o no descomponer ciertas tablas: ceder espacio y robustez a cambio de eficiencia (se ahorran joins)
 - Depende de la frecuencia de consulta, tamaño de las tablas y frecuencias de valores