


Temario

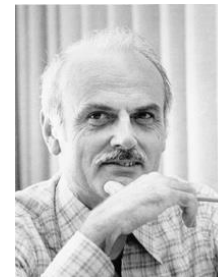
- ♦ Introducción y fundamentos
- ♦ Introducción a SQL
- ♦ Modelo Entidad / Relación
- ♦ Modelo relacional
- ♦ Diseño relacional: formas normales Elmasri cap. 10 y 11 parcial
- ♦ Consultas
 - Cálculo relacional
 - Álgebra relacional
- ♦ Implementación de bases de datos
 - Estructura física: campos y registros
 - Indexación
 - Índices simples
 - Árboles B, B* y B+

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



Raymond F.
Boyce

Directrices informales de diseño

- ♦ **Semántica de los atributos.** Los atributos deben de explicar claramente la relación con la entidad (tabla) a la que pertenecen (autoexplicativos).
- ♦ **Reducción de información redundante en tuplas.** La información redundante ocasiona un potencial riesgo de inconsistencias
- ♦ **Reducción de NULL en las tuplas.** Además de ocupar espacio innecesario, NULL tiene una interpretación de su significado ambigua.
- ♦ **No generar tuplas falsas.** Ocasionado por el uso de atributos para la relación entre tablas que no son claves.

Directrices informales de diseño (cont)

Semántica de los atributos. Tiene que ser sencillo explicar el significado de un esquema de relación. No debe mezclar atributos de distintas entidades pues un esquema representa una entidad y producirá una ambigüedad semántica.

Se mezcla la entidad “estudiante” con la entidad “matricula” de las asignaturas del alumno

<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
⋮	⋮	⋮	⋮

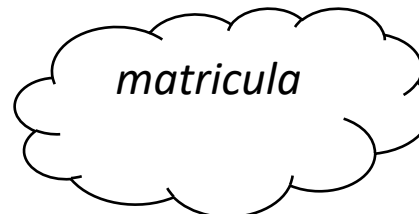
Directrices informales de diseño (cont)

Semántica de los atributos.

<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
⋮	⋮



Mejor diseño...



Directrices informales de diseño (cont)

Reducción de información redundante en tuplas.

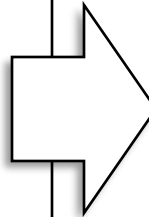
<u>NIE</u>	Nombre	Teléfono	<u>Asignatura</u>
12345	Isabel	123456789	17824
12345	Isabel	123456789	17825
12345	Isabel	123456789	17826
12345	Isabel	123456789	17827
12345	Isabel	123456789	17828
67890	David	321654987	17824
67890	David	321654987	17825
67890	David	321654987	17826
67890	David	321654987	17827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

NIE, Nombre y Teléfono se repiten
pues corresponden a distintos valores
de asignaturas. Da lugar a:

- ♦ Anomalía de inserción
- ♦ Anomalía de borrado
- ♦ Anomalía de modificación

Anomalía de 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...

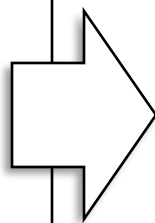
<u>NIE</u>	Nombre	Teléfono
12345	Isabel	123456789
67890	David	321654987
⋮	⋮	⋮

<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. Estudiante sin asignatura o viceversa

Anomalías de borrado

<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
⋮	⋮	⋮	⋮



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

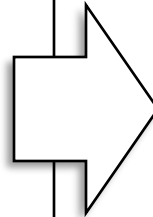
Mejor diseño...

<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

<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
⋮	⋮

- ◆ Problemas al actualizar los atributos de un estudiante: estado inconsistente si no se actualizase el campo en todas las filas

Directrices informales de diseño (cont)

Reducción de valores NULL en tuplas

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

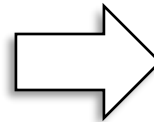
- ♦ NULL ocupa espacio.
- ♦ Tiene distintas interpretaciones:
 - Atributo no aplica a tupla
 - Valor desconocido
 - Conocido pero ausente
- ♦ Interpretación al hacer JOIN
- ♦ Ambigüedad en COUNT, SUM

Directrices informales de diseño (cont)

No generar tuplas falsas

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

Nombre	<u>Asignatura</u>
Isabel	17824
Isabel	17825
Isabel	17826
Isabel	17827
Isabel	17828
Isabel	17824
Isabel	17825
Isabel	17826
Isabel	17827
⋮	⋮



JOIN

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

Falso

- ♦ Si dos tablas están relacionadas por un atributo que no es clave primaria da lugar a tuplas falsas al hacer el join.

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 diseño (cont)

- ♦ Las inconsistencias típicamente afloran al hacer consultas
 - Un estudiante ya no aparece
 - Un estudiante con dos números de teléfono, o dos direcciones, dos nombres...
- ♦ Las anomalías de diseño se solucionan normalmente descomponiendo esquemas, de forma que:
 - Los esquemas resultantes cumplan unas ciertas propiedades: formas normales
 - Se preserve el join
- ♦ Vamos a ver primero las formas normales, después los algoritmos de descomposición
- ♦ Las formas normales se basan en la noción de dependencia funcional

Superclave, clave, clave candidata, clave principal, atributo primo

- ♦ Una **Superclave** X de un esquema de relación $R=\{A_1, \dots, A_n\}$ 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 $t_1[X]=t_2[X]$.
- ♦ Una **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 mas de una clave:
 - Cada una de ellas se llama clave candidata.
 - Una de ellas se elige arbitrariamente como **clave primaria**
 - 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

- ◆ Def: Dados dos conjuntos X e Y de atributos de un esquema R, Y depende funcionalmente de X si $t_i[X] = t_j[X] \Rightarrow t_i[Y] = t_j[Y], \forall t_i, t_j \in r(R)$
Es decir, los atributos de Y están unívocamente determinados por los de X.
Esto se representa por $X \rightarrow Y$

- ◆ Ejemplos

- {Dni,Nvuelo,Fecha} es una clave en la siguiente relación:
- Reserva (Dni, Nvuelo , Fecha, Nombre, Origen, Destino, Hora, Precio)
 - DF1: {Dni} \rightarrow Nombre
 - DF2: {Nvuelo} \rightarrow {Origen, Destino, Hora}
 - DF3: {Nvuelo, Fecha} \rightarrow {Origen, Destino, Hora, Precio}
 - DF4: {Dni, Nvuelo} \rightarrow {Nombre, Origen, Destino, Hora}
 - DF5: {Dni, Nvuelo, Fecha} \rightarrow {Nombre, Origen, Destino, Hora, Precio}

Dependencias funcionales (cont)

- ♦ Las dependencias funcionales:
 - Obtenidas a partir de la documentación de las especificaciones del problema (MAS IMPORTANTE)
 - Aquellas que son semánticamente obvias (POCAS)
 - Existen otras que no son las anteriores pero que se pueden deducir a partir de éstas.
 - Para determinar/deducir las DF hay que tener unas reglas de inferencia

Dependencias funcionales (cont)

F es un conjunto de dependencias funcionales en un esquema de relación R. El conjunto de todas las dependencias que incluye F junto con las que pueden inferirse a partir de ellas se llama clausura de F, F^+

Reglas de inferencia:

RI1 (reglas reflexiva): Si $X \supseteq Y$, entonces $X \rightarrow Y$

RI2 (regla aumento): $(X \rightarrow Y) \models XZ \rightarrow YZ$

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

RI4 (regla proyectiva): $\{X \rightarrow YZ\} \models X \rightarrow Y$

RI5 (regla aditiva): $\{X \rightarrow Y, X \rightarrow Z\} \models X \rightarrow YZ$

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

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

Dependencias funcionales (cont)

- ♦ Obsérvense que:

$$\{XY \rightarrow Z\} \text{ NO } \models X \rightarrow Z$$

$$\{X \rightarrow Y, Z \rightarrow W\} \models XZ \rightarrow YW$$

$$X \rightarrow Z \models XY \rightarrow Z$$

- ♦ RI1, RI2 y RI3 son las reglas inferencia de Armstrong. Todas las reglas se derivan a partir de ellas.
- ♦ Dos conjuntos de DFs, F y E, son **equivalentes** si toda dependencia de uno se puede inferir de las dependencias del otro y viceversa ($F^+ = E^+$).
- ♦ Un conjunto de DFs, F, es **mínimo (forma canónica y sin redundancias)** 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.
- ♦ Una **cobertura mínima** de un conjunto de dependencias funcionales E es un conjunto mínimo de dependencias, F, equivalente a E.

Encuentra una cobertura mínima F de las DFs E

1. $F := E$
2. **Descomponer en dependencias sobre atributos individuales en la parte derecha**

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

3. **Eliminar atributos que sobren en las partes izquierdas**

for $X \rightarrow A \in F$ do

for $B \in X$ 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)

4. **Eliminar dependencias que se infieran de otras**

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

Cobertura mínima (cont)

En otras palabras, el algoritmo anterior:

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

Conjuntos mínimos: ejemplo

Supongamos las DFs E: {DF1: $B \rightarrow A$, DF2: $D \rightarrow A$, DF3: $AB \rightarrow D$ }.

1. $F := E$
2. **Descomponer en dependencias sobre atributos individuales en la parte derecha.**
 - Ya esta
3. **Eliminar atributos que sobra en las partes izquierdas (atributos redundantes)**
 - De la única que podemos quitar es de $AB \rightarrow D$. Se puede quitar A o B de la izquierda quedando $B \rightarrow D$ o $A \rightarrow D$
 - Es decir, ¿se podría sustituir $AB \rightarrow D$ por $B \rightarrow D$ o $A \rightarrow D$?
 - A partir de DF1: $B \rightarrow A$ aumentando con B (RI2) inferimos $BB \rightarrow AB$ que es lo mismo que $B \rightarrow AB$. De esta última y DF3, $\{B \rightarrow AB, AB \rightarrow D\} \models B \rightarrow D$.
 - Por tanto, de E se infiere $B \rightarrow D$ y por tanto se puede sustituir DF3 por $B \rightarrow D$, ya que $B \rightarrow D \models AB \rightarrow D$
4. **Eliminar dependencias que se infieran de otras (inferencias redundantes)**
 - Tenemos $\{B \rightarrow A, D \rightarrow A, B \rightarrow D\}$, así que $\{B \rightarrow D, D \rightarrow A\} \models B \rightarrow A$, y ésta ya está por lo que es redundante. Así $F := \{B \rightarrow D, D \rightarrow A\}$
 - return F (¡BC es clave!)

Ejemplo

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

1. $D := \emptyset$

2. $G :=$ cobertura mínima de R

- Pasamos de $B \rightarrow ACDE$ a $B \rightarrow A$, $B \rightarrow C$, $B \rightarrow D$, $B \rightarrow E$
- Pasamos de $E \rightarrow FG$ a $E \rightarrow F$, $E \rightarrow G$
- Las partes izquierdas ya están en su forma simple
- No hay dependencias redundantes (DFs que se infieran de otras DFs)
- $G := \{B \rightarrow A, B \rightarrow C, B \rightarrow D, B \rightarrow E, E \rightarrow F, E \rightarrow G\}$
- (¡B es clave!)

Formas normales

- ♦ Su cumplimiento reduce anomalías de inserción, borrado y actualización y mejora las propiedades del diseño.
- ♦ Esquemas que no cumplen se descomponen en más pequeños que si las cumplen
- ♦ Son incrementales
 - Si se cumple la forma normal n -ésima se cumple la $(n-1)$ -ésima



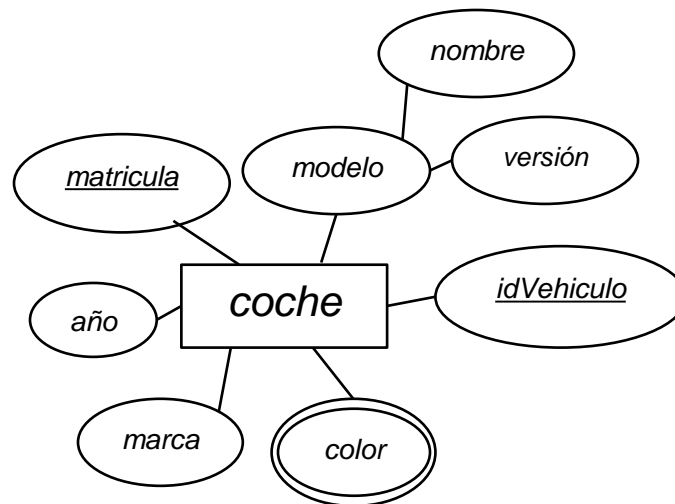
Formas normales (cont)

- ♦ Formas normales 1ª, 2ª, 3ª, BCNF
 - Involucran a un solo esquema
 - 1ª simplemente impide atributos multivalor y los compuestos
 - 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 que ocurren en situaciones muy excepcionales.
- ♦ Se normaliza para evitar la redundancia, mantener la integridad de los datos y evitar errores.

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.
- ♦ Ejemplo: coche, atributo multivalor (color) y compuesto (modelo)



1ª forma normal (1NF)

Formas de solucionarlo:

1. Segunda relación con clave primaria el conjunto los atributos de la primera y el multivalor

<u>matrícula</u>	año	marca
5654HKL	2005	Mercedes
9283MGF	2020	Tesla
⋮	⋮	⋮

<u>matrícula</u>	<u>color</u>
5654HKL	Rojo
5654HKL	blanco
⋮	⋮

2. Expandir la tupla original para cada color. Esta solución tiene el problema de la redundancia

<u>matrícula</u>	año	marca	<u>color</u>
5654HKL	2005	Mercedes	rojo
5654HKL	2005	Mercedes	blanco
9283MGF	2020	Tesla	negro
⋮	⋮	⋮	

1ª forma normal (1NF)

3. Aumentar el numero de atributos con el número máximo de valores del atributo multivalor, por ejemplo 3. Da lugar a muchos NULLs

<u>matrícula</u>	año	marca	color1	color2	color3
5654HKL	2005	Mercedes	rojo	Blanco	NULL
9283MGF	2020	Tesla	negro	NULL	NULL
⋮	⋮	⋮	⋮	⋮	⋮

De las tres soluciones posibles la primera es la óptima:

- pues es general
- no da lugar a NULL
- no tiene redundancia
- no está limitada a un número máximo de valores

2ª forma normal (2NF)

- ♦ Def: Un esquema R es 2NF si es 1NF y si todo atributo no primo de R tiene una dependencia funcional plena con las claves de R.
- ♦ 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 no primarios dependen de la clave completa.
- ♦ Dicho de otro modo: cualquier atributo que no forme parte de la clave primaria y tenga una dependencia con la clave, depende de toda la clave en vez de solo una parte de ella.

2ª forma normal: ejemplo

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

♦ Ejemplos

- Reserva (124, 165467, '2011-10-24', 'Ana', 'MAD', 'LAX', '16:25:00', 620)
- Reserva (123, 165467, '2011-10-24', 'Luis', 'MAD', 'LAX', '16:25:00', 620)
- Reserva (123, 165467, '2011-11-18', 'Luis', 'MAD', 'LAX', '16:25:00', 620)

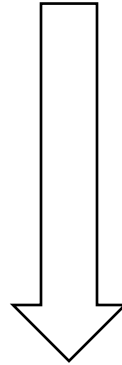
♦ Clave: {Dni, Nvuelo, Fecha}.

♦ Dependencias Funcionales:

- | | |
|---|--------|
| – Dni \rightarrow {Nombre} | No 2NF |
| – Nvuelo \rightarrow {Origen, Destino, Hora } . | No 2NF |
| – {Nvuelo, Fecha} \rightarrow {Precio} | No 2NF |

2ª forma normal: ejemplo (cont)

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



2NF (trocear la relación en varias)

– Se elimina redundancia

R1 (DNI, Nombre)

R2 (Nvuelo, Origen, Destino, Hora)

R3 (Nvuelo, Fecha, Precio)

R4 (DNI, Nvuelo, Fecha)

3ª forma normal (3NF)

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

- ♦ Def: Un esquema es 3NF si es 2NF y 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, un atributo no puede depender de algo que no sea una superclave, excepto acaso los atributos que forman parte de alguna clave
- ♦ Def: Un esquema de relación R es 3NF si satisface 2FN y ningún atributo no primo de R es transitivamente dependiente en la clave principal.
- ♦ Def: Una dependencia funcional $X \rightarrow Y$ 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 $X \rightarrow Z$ como $Z \rightarrow Y$

3ª forma normal: ejemplo (cont)

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

- $Nvuelo \rightarrow \{ Origen, Destino, Ciudad_origen, Ciudad_destino, Hora \}$

– Origen \rightarrow Ciudad_origen

– Destino \rightarrow Ciudad_destino

← **No 3NF**

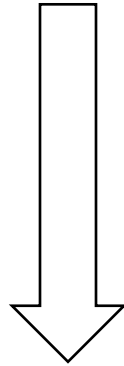
} *Ok 2NF*

No superclaves

No primarios

3ª forma normal: ejemplo 1

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



3NF (trocear la relación en varias)

– Se elimina redundancia

R1 (Nvuelo, Origen, Destino, Hora)

R2 (codigo, ciudad) /* con código igual a origen o
destino y ciudad correspondería a origen y destino */

3ª forma normal: ejemplo 2

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

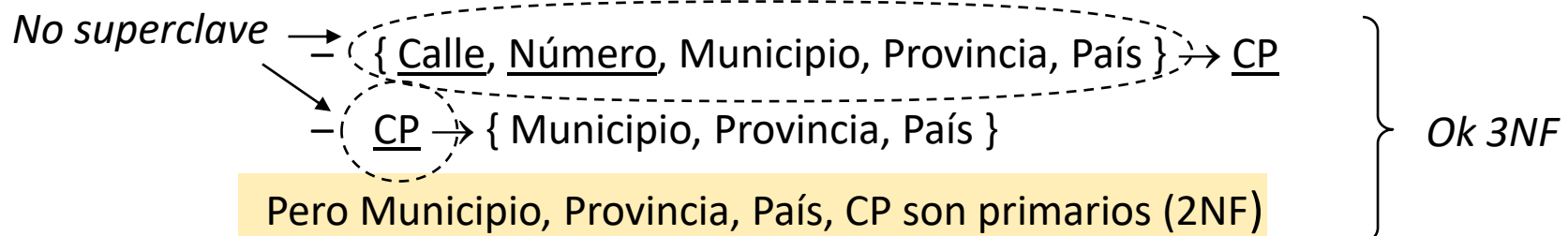
♦ Ejemplos

- 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



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

“The key, the whole key, and nothing but the key –so help me Codd”

Forma normal Boyce-Codd: ejemplo 1

Vuelo (Nvuelo, Origen, Destino, Hora)

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

Pasajero (Dni, Nombre)

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

Reserva (Dni, Nvuelo, Fecha)

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

Forma normal Boyce-Codd: ejemplo 2

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

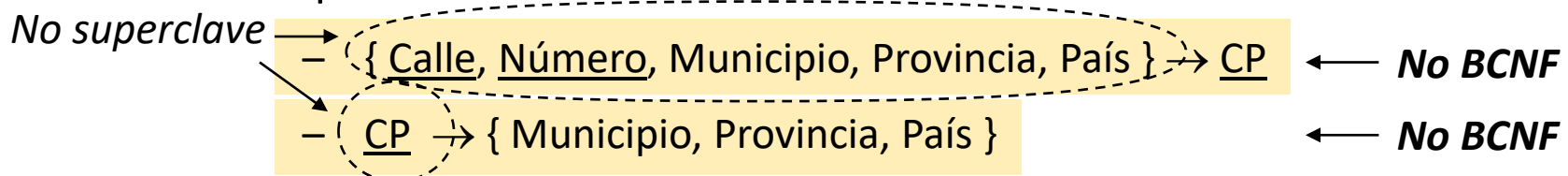
♦ Ejemplos

- 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



Un último par de ejemplos...

1. Si todas las claves tienen un solo atributo, ¿sabemos ya algo de la forma normal del esquema?
2. ¿Cuál es la mínima forma normal de un esquema en el que la combinación de todos sus atributos forma una clave?

Un último par de ejemplos...

1. Caso 1

1. Es 1NF pues todos los atributos son atómicos y univaluados
2. Es 2NF pues cualquier atributo depende de toda la clave primaria pues ésta no tiene partes.
3. Es 3NF pues para cualquier dependencia funcional $X \rightarrow Y$, X es la clave, que no tiene partes, e Y atributo no primo.
4. Es BCNF pues para cualquier dependencia $X \rightarrow Y$, X es la clave

2. Caso 2

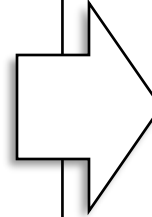
1. Todos los atributos son primarios o primos \Rightarrow BCNF

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	Dolor
Gelocatil	Paracetamol	Cefalea
Tylenol	Paracetamol	Fiebre
Tylenol	Paracetamol	Dolor
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	Dolor
Paracetamol	Cefalea
Codeína	Tos
Codeína	Dolor

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

Resumen

Dada $X \rightarrow Y \dots$

		X superclave		
		Sí	No	
			X primario	
			Sí	No
		BCNF	3NF	
Y primario	No		1NF	2NF

Normalizar o Desnormalizar

- ♦ La normalización lleva a eliminar redundancias y por tanto potenciales errores durante la actualización.
- ♦ Una consecuencia de la normalización es:
 - Se generan mas relaciones (tablas).
 - Las consultas sobre estas relaciones posiblemente requerirán de JOIN.
 - Si las tablas son grandes el JOIN es lento y por tanto la consulta también será lenta
- ♦ Existe un compromiso entre la normalización y la velocidad de consulta. En ocasiones se sacrifica normaliza por velocidad

Normalización de esquemas relacionales

Hay dos maneras de obtener esquemas en una determinada forma normal

1. Diseñamos esquemas normalizados desde el principio

- Esto se produce generalmente, por ejemplo, de forma natural cuando aplicamos una conversión sistemática de E/R a modelo relacional

2. Partimos de un diseño no normalizado

- Por ejemplo, es un diseño que hemos heredado, o que no hemos sabido diseñar mejor inicialmente
- Normalizamos el esquema → esto consiste en descomponer el esquema inicial en varios esquemas en la forma normal deseada
- Pero no de cualquier manera...

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, $\bigcup_{i=1}^n R_i = R$
- ♦ 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 otras palabras, la unión de las dependencias de las tablas R_i equivale a las dependencias de la tabla original.

- ♦ Join sin pérdida (join no aditivo). Evita la generación de tuplas falsas al hacer NJ
 - 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
 - En otras palabras: el join de las tablas R_i es la tabla original R
 - Evita las tuplas falsas al hacer natural join

Operaciones de álgebra relacional (más adelante...)

Matrícula

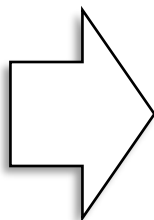
<u>NIE</u>	Nombre	Teléfono	<u>Asignatura</u>
12345	Isabel	123456789	17824
12345	Isabel	123456789	17825
67890	David	321654987	17826
67890	David	321654987	17827
89456	Isabel	755326284	17835
89456	Isabel	755326284	17838
⋮	⋮	⋮	⋮

Estudiante

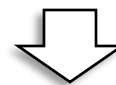
<u>NIE</u>	Nombre	Teléfono
12345	Isabel	123456789
67890	David	321654987
89456	Isabel	755326284
⋮	⋮	⋮

Matrícula

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



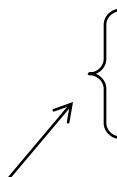
SELECT * FROM Estudiante
NATURAL JOIN Matrícula



Tuplas espurias (falsas): ejemplo

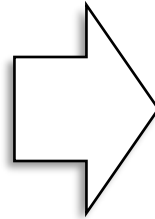
Tuplas espurias

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



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
⋮	⋮	⋮	⋮



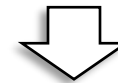
Estudiante

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

Matrícula

<u>NIE</u>	<u>Asignatura</u>
12345	17824
12345	17825
67890	17826
67890	17827
89456	17835
89456	17838
⋮	⋮

SELECT * FROM Estudiante
NATURAL JOIN Matrícula



**NO Tuplas
espurias:**

<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
⋮	⋮	⋮	⋮

Algoritmos de normalización

- ♦ Comprobación de
 - preservar dependencias en descomposiciones
 - join sin pérdida en descomposiciones
 - Se mantienen las superclaves
- ♦ 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) /* Esta descomposición tiene join sin perdida y preserva dependencias, R es cualquier relación y F un conjunto de DFs de R */

1. $D := \emptyset$

2. $G :=$ cobertura mínima de F

/* sacar a tablas los atributos de todos lo conjuntos de dependencias con la misma parte izquierda */

3. 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 no hay tablas 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)

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” (que será una clave de la tabla)
3. Si no ha salido ninguna tabla con una clave original completa, crear esa tabla
4. Eliminar esquemas redundantes

Ejemplo

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

1. $D := \emptyset$

2. $G :=$ cobertura mínima de R

- Pasamos de $B \rightarrow ACDE$ a $B \rightarrow A$, $B \rightarrow C$, $B \rightarrow D$, $B \rightarrow E$
- Pasamos de $E \rightarrow FG$ a $E \rightarrow F$, $E \rightarrow G$
- Las partes izquierdas ya están en su forma simple
- No hay dependencias redundantes (DFs que se infieran de otras DFs)
- $G := \{B \rightarrow A, B \rightarrow C, B \rightarrow D, B \rightarrow E, E \rightarrow F, E \rightarrow G\}$
- B es la clave

La forma normal mínima es 2NF (por $E \rightarrow F$, $E \rightarrow G$)

Ejemplo (cont)

/* sacar a tablas los atributos de todos lo conjuntos de dependencias con la misma parte izquierda */

3. 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

Así que tenemos:

$D := \{B \cup \{A, C, D, E\}, E \cup \{F, G\}\} = \{\{B, A, C, D, E\}, \{E, F, G\}\} = \{R1, R2\}$

/* si no hay tablas 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 (Ya hay un esquema con la clave B)
5. Eliminar los esquemas redundantes de D (esquemas incluidos en otros). No es el caso
6. La clave primaria de R1 es B, la de R2 es E que es la clave foránea con R1.

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 los dos esquemas $(Q - Y), (X \cup Y)$

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

Ejemplo 1

$R = \{A, B, C, D\}$, A clave primaria y clave candidata $\{BC\}$, las dependencias funcionales son $DF1: A \rightarrow BCD$, $DF2: BC \rightarrow AD$, $DF3: D \rightarrow B$.

Hay una DF que no es BCNF, $DF3: D \rightarrow B$

Usamos la $DF3$. Sustituir 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 la de $R2$ es D . D , en $R1$, es clave foránea con $R2$.

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

La dependencia $DF2$ ya no existe

Ejemplo 2

$R = \{\text{Estudiante}, \underline{\text{Asignatura}}, \text{Profesor}\}$

$R = \{\underline{A}, \underline{B}, C\}$

DF1: $\{\text{Estudiante}, \text{Asignatura}\} \rightarrow \text{Profesor}$

$AB \rightarrow C$

DF2: $\text{Profesor} \rightarrow \text{Asignatura}$

$C \rightarrow B$

Solo hay una DF que no es BCNF que es DF2: $C \rightarrow B$

Sustituir el esquema R por $(R-B)$ y $(C \cup B)$

$R = \{R1, R2\} = \{\{\underline{A}, \underline{C}\}, \{\underline{C}, B\}\}$

El algoritmo genera una descomposición que tiene join sin pérdida, pero no asegura la preservación de todas las dependencias, DF1 se ha perdido

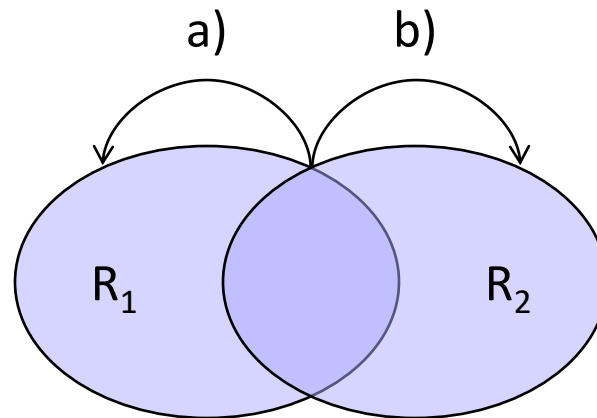
Natural Join sin pérdida

Test sencillo:

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

O bien a) $R_1 \cap R_2$ es una superclave en R_1 (y actúa como clave externa en R_2)

O bien b) $R_1 \cap R_2$ es una superclave en R_2 (y actúa como clave externa en R_1)



Ejemplo: test NATURAL JOIN sin perdida

Según el ejercicio anterior

$R = \{\underline{\text{Estudiante}}, \underline{\text{Asignatura}}, \text{Profesor}\}$

$R = \{\underline{A}, \underline{B}, C\}$

DF1: $\{\text{Estudiante}, \text{Asignatura}\} \rightarrow \text{Profesor}$

$AB \rightarrow C$

DF2: $\text{Profesor} \rightarrow \text{Asignatura}$

$C \rightarrow B$

Se descompone en: $R = \{R_1, R_2\} = \{\{\underline{A}, \underline{C}\}, \{\underline{C}, B\}\}$ con DF1: $C \rightarrow B$

a) $R_1 \cap R_2$ es superclave de R_1 y clave externa de R_2

b) $R_1 \cap R_2$ es superclave de R_2 y clave externa de R_1

$R_1 \cap R_2 = C$ es superclave de R_2 y clave externa para R_1

Se cumple, en este caso la opción b, y por tanto $R = \{R_1, R_2\}$ producen un NJ sin perdida

Ejemplo: Test NATURAL JOIN sin perdida

$$R = \{R_1, R_2\} = \{\{\underline{A}, \underline{C}\}, \{\underline{A}, \underline{B}\}\}$$

a) $R_1 \cap R_2 = A$ es superclave de R_1 y clave externa de R_2 (No es el caso)

b) $R_1 \cap R_2 = A$ es superclave de R_2 y clave externa de R_1 (No es el caso)

No produce, por tanto, un NJ sin perdida y daría lugar a tuplas falsas como resultado.

$$R = \{R_1, R_2\} = \{\{B, \underline{C}\}, \{\underline{A}, \underline{B}\}\} \text{ y por tanto DF1: } C \rightarrow B$$

a) $R_1 \cap R_2 = B$ es superclave de R_1 y clave externa de R_2 (No es el caso)

b) $R_1 \cap R_2 = B$ es superclave de R_2 y clave externa de R_1 (No es el caso)

No produce por tanto un NJ sin perdida y daría lugar a tuplas falsas como resultado

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 desnormalización
 - 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