



UNIVERSIDAD  
DE GRANADA



# Diseño y Desarrollo de Sistemas de Información

## Grado en Ingeniería Informática

### Tema 3 – Diseño avanzado de Bases de Datos Relacionales

©I. J. Blanco, F. M. Castro, C. Cruz, C de Mesa, M. J. Martín, J. Paños, D. Sánchez

**Departamento de Ciencias de la  
Computación e Inteligencia Artificial**  
<http://decsai.ugr.es>

- ❑ Recordatorio: El modelo de datos relacional
- ❑ Diseño lógico relacional: Normalización
  - Formas normales. Primera forma normal
  - Concepto de dependencia funcional
  - Segunda forma normal (2FN)
  - Teorema de Heath y descomposición sin pérdidas
  - Cierre de un conjunto de dependencias
  - Preservación de dependencias en descomposiciones sin pérdidas
  - Tercera forma normal (3FN)
  - Forma normal de Boyce-Codd (FNBC)
- ❑ Algoritmo de cálculo de claves candidatas
- ❑ Algoritmo de Normalización a FNBC
- ❑ Recubrimiento minimal de un conjunto de dependencias. Algoritmo
- ❑ Dependencias multivaluadas y cuarta forma normal (4FN)

- ❑ Basado en el concepto de **Relación** que, formalmente, puede verse como un par de conjuntos  $(R, r)$  donde **R es el esquema** y **r es la instancia** de la relación.
- ❑ El **esquema** es un conjunto de atributos con dominios asociados

$$R = \{A_1 : D_1, A_2 : D_2, \dots, A_n : D_n\}$$

- ❑ La **instancia** es un subconjunto del producto cartesiano de los dominios que debe cumplir las restricciones semánticas y de integridad

$$r \in D_1 \times D_2 \times \dots \times D_n$$

Visualmente, es una estructura bidimensional formada por columnas (**atributos**) y filas (**tuplas**)

A1	A2	A3	A4	A5

Un conjunto CC de atributos en una relación se dice que es una **clave candidata** si se verifica lo siguiente:


**Unicidad.** Para toda instancia de la relación, si dos tuplas de dicha instancia coinciden en los atributos de la clave candidata, entonces coinciden en el resto de atributos de la relación.

**Minimalidad.** La propiedad anterior no se verifica para ningún subconjunto propio de CC.

- Un conjunto de atributos SC en una relación se llama **superclave** si verifica la unicidad.
- Se llama **clave primaria** a una clave candidata elegida por el diseñador (sólo una por relación)

Dada una clave candidata **CC** de una relación **R**, y un conjunto de atributos **CE** de una relación **S**, con **CE** contenido en **CC**.

Se dice que **CE** es **clave externa** respecto a **CC** si el dominio activo (conjunto de valores que puede tomar) de **CE** debe estar contenido en el dominio activo de **CC**.



Profesor	DNI
G.N.G	21323431
M.P.C	34235571
K.B.A	55555111
L.E.Z	12432139
B.P.C	19392931

DNI	Departamento
21323431	DECSAI
34235571	DECSAI
55555111	LSI

**Es necesario que los DNI sean de profesores!!**

### Reglas de integridad genéricas:

- ❑ **Integridad de entidad.** Los atributos de una clave primaria no pueden tomar valores nulos
- ❑ **Integridad referencial.** El valor de una clave externa debe ser igual a un valor del dominio activo de la clave primaria, o nulo si la semántica lo permite.

**Restricciones específicas**, son aquellas que provienen de la semántica del atributo y son propias de cada base de datos concreta. Ejemplos:

- $EDAD \geq 0$
- $NUM\_ALUMNOS \leq NUM\_MAX\_ALUMNOS$
- $SUELDO \geq IMPUESTOS + SEG\_SOCIAL$
- ...

- ❑ El **esquema de una base de datos** es una colección de **esquemas de relación** junto con una serie de **restricciones de integridad** que definen las configuraciones válidas de los datos.
- ❑ La **instancia de una base de datos** es el conjunto de instancias de sus tablas.
- ❑ Una **instancia** de la base de datos (conjunto de instancias de sus tablas) es **válida si respeta la reglas de integridad**.



**Diseño Lógico Relacional:** proceso que permite **generar un esquema relacional** a partir de una representación conceptual (*esquema entidad-relación*) de la información relacionada con un sistema dado. También se le conoce como *paso a tablas*.

El diseño Lógico relacional obtenido a partir del esquema entidad-relación puede refinarse mediante un proceso de **Normalización**, propio del modelo relacional.

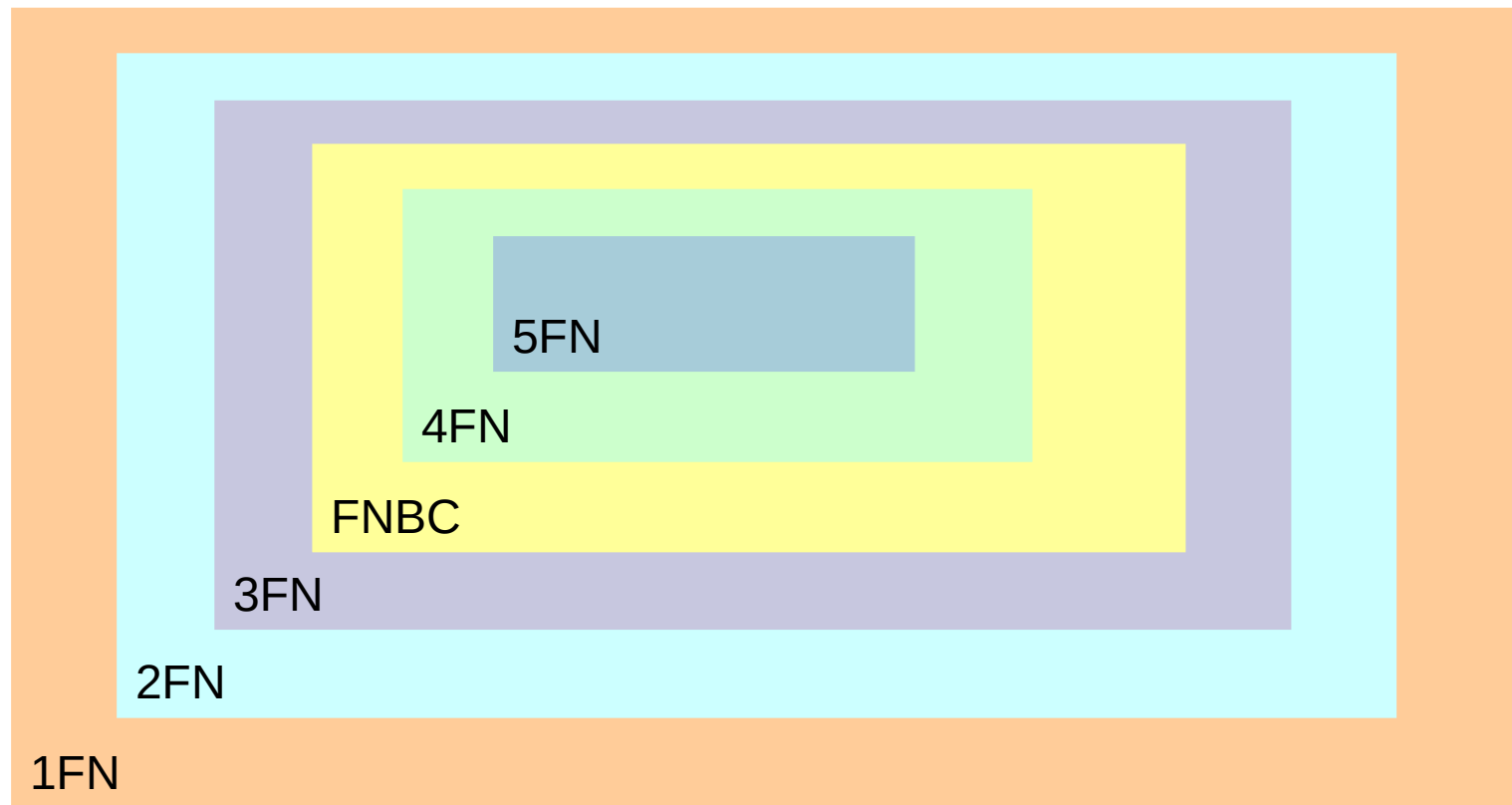
Objetivos de la Normalización:

- ☐ Corregir defectos del modelo conceptual
- ☐ Eliminar redundancias, problemas de actualización
- ☐ Plasmar restricciones semánticas adicionales

- ❑ Tratamos de conseguir un diseño relacional de una base de datos que cumpla una serie de características, lo que garantiza su buen comportamiento.
- ❑ Las buenas propiedades que cumple una relación se denominan **formas normales**.

Forma Normal	Año	Autor	Basada en...
<b>1FN</b>	1970	Codd	No basada en dependencias: Impone dominios atómicos
<b>2FN</b>	1970	Codd	Dependencias funcionales
<b>3FN</b>	1970	Codd	
<b>FNBC</b>	1972	Boyce & Codd	
<b>4FN</b>	1977	Fagin	Dependencias multivaluadas
<b>5FN</b>	1979	Rissanen	Dependencias de reunión

Relación entre conjuntos de instancias que cumplen cada una de las formas normales



## Dependencia funcional:

Dada una relación  $R$  con  $n > 0$  atributos y dos subconjuntos de atributos de  $R$  llamados  $\alpha$  y  $\beta$ , se dice que  $\alpha$  determina funcionalmente a  $\beta$  (o equivalentemente que  $\beta$  depende funcionalmente de  $\alpha$ ) si para cualquier instancia válida  $r$  de  $R$  y cualquier pareja de tuplas  $s$  y  $t$  de  $r$  que tengan los mismos valores para los atributos del subconjunto  $\alpha$ , se verifica que tienen los mismos valores para los atributos del subconjunto  $\beta$ .

**Dicho de otra manera:** la combinación de valores de  $\alpha$  determina la combinación de valores de  $\beta$ .

$$R(A_1, A_2, \dots, A_n), \alpha \subset R, \beta \subset R, \alpha \rightarrow \beta \text{ si i}$$

$$\forall \text{ instancia } r \text{ de } R, \forall t, s \in r, t[\alpha] = s[\alpha] \Rightarrow t[\beta] = s[\beta]$$

- Las **dependencias funcionales** establecen **restricciones semánticas** conocidas y que deben verificarse en los datos

DNI, Nombre, Cargo, Salario, etc.

- Una forma de conseguirlo es mediante un **diseño adecuado de la base de datos**, es decir, determinar los esquemas de tablas adecuados. Concretamente, los esquemas adecuados son aquellos que cumplen las formas normales 2FN, 3FN y FNBC.
- Un conjunto de dependencias funcionales en una tabla sirve asimismo para determinar TODAS las claves candidatas, como veremos más adelante.

## Dependencia funcional completa:

Dada una relación  $R$  con  $n$  atributos y dos subconjuntos de atributos de  $R$  llamados  $\alpha$  y  $\beta$ , se dice que  $\alpha$  *determina funcionalmente a  $\beta$  de forma completa o determina completamente a  $\beta$*  si  $\alpha$  determina funcionalmente a  $\beta$  y no hay ningún subconjunto de atributos de  $\alpha$  que determine funcionalmente a  $\beta$ .

$$R(A_1, A_2, \dots, A_n), \alpha \subset R, \beta \subset R, \alpha \twoheadrightarrow \beta \text{ sii} \\ \alpha \rightarrow \beta \wedge \nexists \gamma \subset \alpha \mid \gamma \rightarrow \beta$$

- Se llama **dependencia trivial** a toda dependencia  $\alpha \rightarrow \beta$  tal que  $\beta \subseteq \alpha$
- Nótese que en un esquema el número de dependencias triviales es **exponencial** en el número de atributos.
- Se llama **atributo primo** a cualquier atributo que forma parte de alguna clave candidata.



## segunda forma normal

Decimos que una relación  $R$  está en segunda forma normal (2FN) sii:

- Está en primera forma normal (1NF) y
- Todos sus atributos no primos dependen de forma completa de las claves candidatas.

Un buen diseño conceptual genera tablas que están en segunda forma normal.

# Normalización basada en dependencias: segunda forma normal

## Ejemplo:

**STOCK** (almacén, producto, cantidad, dirección\_almacén)

## Problemas

- La dirección del almacén se repite para cada producto existente en el inventario del almacén.
- Si la dirección del almacén cambia, hay que actualizar todas las tuplas relativas a los distintos productos almacenados en el almacén.
- Debido a la redundancia existente, pueden aparecer inconsistencias si distintas tuplas contienen distintos valores para la dirección de un mismo almacén.
- Si en algún momento no existiese stock alguno en el almacén, no habría ningún sitio donde almacenar su dirección.

# Normalización basada en dependencias: segunda forma normal

## Ejemplo:

**STOCK** (almacén, producto, cantidad, dirección\_almacén)

## Causa:

La clave de la relación es una clave compuesta:

**{almacén, producto}**

El atributo 'dirección\_almacén' no pertenece a la clave y depende sólo de parte de ella (del atributo '**almacén**').

**La relación STOCK no está en 2FN**

## Normalización basada en dependencias: segunda forma normal

Ejemplo:

$$DF = \left\{ \begin{array}{l} \{\text{almacén}, \text{producto}\} \rightarrow \text{cantidad}, \{\text{almacén}, \text{producto}\} \rightarrow \text{dirección almacén}, \\ \text{almacén} \rightarrow \text{dirección almacén} \end{array} \right\}$$

Podemos solucionarlo mediante la aplicación del **Teorema de Heath** usando la dependencia que nos da problemas.

## Normalización basada en dependencias: segunda forma normal

### Teorema de Heath:

Sea  $R$  una relación con atributos  $A$ ,  $B$ , y  $C$ , donde se verifica  $B \rightarrow C$ . Entonces, la descomposición de  $R$  en dos relaciones:

$R_1(A, B)$

$R_2(B, C)$

es una *descomposición sin pérdidas*

## Normalización basada en dependencias: segunda forma normal

### Descomposición sin pérdidas:

Sea  $(R, r)$  una relación que se descompone en  $(R_1, r_1)$  y  $(R_2, r_2)$ . Se dice que la descomposición es sin pérdidas sii:

$$R_1 \cup R_2 = R$$

$$r_1 \text{ JOIN } r_2 = r$$

Es decir, podemos recuperar  $R$  mediante reunión natural.

## Normalización basada en dependencias: segunda forma normal

En nuestro ejemplo,

la dependencia entre almacén (parte de la clave) y dirección\_almacén, hace que la relación no esté en segunda forma normal. Aplicamos el **Teorema de Heath** sobre esa dependencia y nos quedan dos relaciones.

$$R_1 = \{\text{almacén, producto, cantidad}\}$$

$$DF_1 = \{\{\text{almacén, producto}\} \rightarrow \text{cantidad}\}$$

$$R_2 = \{\text{almacén, dirección almacén}\}$$

$$DF_2 = \{\text{almacén} \rightarrow \text{dirección almacén}\}$$

## Preservación de dependencias

No obstante, la relación  $R$  cumplía una serie de restricciones (dependencias funcionales) antes de descomponerse.

¿Esas restricciones se siguen cumpliendo en las dos relaciones resultantes?

En nuestro ejemplo, todas las dependencias originales se encuentran dentro de  $DF_1$  o de  $DF_2$  excepto la dependencia

**{almacén, producto} → dirección\_almacén**

¿Se ha perdido esta dependencia?



# Normalización basada en dependencias: Preservación de dependencias

## ¿Se ha perdido realmente?

- La respuesta no es tan sencilla porque de la definición de dependencia funcional se derivan una serie de axiomas y reglas que nos permiten operar con ellas. Se conocen como los **Axiomas de Armstrong** y las correspondientes **reglas** que se derivan de ellos.
- Pudiera ser que podamos recuperar lo que supuestamente se ha perdido a partir de las dependencias que quedan aplicando dichos axiomas y reglas.

## Normalización basada en dependencias: Preservación de dependencias

### Axiomas de Armstrong:

- **Reflexividad:**  $\forall \alpha, \beta \mid \beta \subseteq \alpha$  se verifica  $\alpha \rightarrow \beta$
- **Ampliación:**  $\forall \alpha, \beta, \gamma \mid \alpha \rightarrow \beta$  se verifica  $\alpha\gamma \rightarrow \beta\gamma$
- **Transitividad:**  $\forall \alpha, \beta, \gamma \mid \alpha \rightarrow \beta \wedge \beta \rightarrow \gamma$  se verifica  $\alpha \rightarrow \gamma$

## Normalización basada en dependencias: Preservación de dependencias

### Reglas

#### Unión:

$\forall \alpha, \beta, \gamma \mid \alpha \rightarrow \beta \wedge \alpha \rightarrow \gamma$  se verifica  $\alpha \rightarrow \beta\gamma$

#### Descomposición:

$\forall \alpha, \beta, \gamma \mid \alpha \rightarrow \beta\gamma$  se verifica  $\alpha \rightarrow \beta \wedge \alpha \rightarrow \gamma$

#### Pseudotransitividad:

$\forall \alpha, \beta, \gamma, \delta \mid \alpha \rightarrow \beta \wedge \beta\gamma \rightarrow \delta$  se verifica  $\alpha\gamma \rightarrow \delta$

# Normalización basada en dependencias: Preservación de dependencias

- **Volver a obtener dependencias** que parecen haberse perdido, **es un proceso tedioso** y requiere claridad de visión, ya que se pueden aplicar secuencias de axiomas y reglas que pueden llevarnos a callejones sin salida.
- Para verificar si las dependencias siguen existiendo sin que estén explícitamente presentes (es decir, que se pueden deducir de otras) **se emplean otros conceptos**

# Normalización basada en dependencias: Preservación de dependencias

## Cierre de un conjunto de dependencias funcionales $F$ :

Se nota por  $F^+$  y representa el conjunto de todas las dependencias funcionales que pueden deducirse de las dependencias funcionales de  $F$  aplicando los Axiomas y las Reglas de Armstrong en una secuencia finita de pasos.

## Normalización basada en dependencias: Preservación de dependencias

Cierre de un conjunto de atributos  $\alpha$  en base a un conjunto de dependencias funcionales  $\Phi$ :

Se nota por  $\alpha_F^+$  y representa el conjunto de todos los atributos que son determinados por los atributos de  $\alpha$  en conjunto mediante dependencias de  $F^+$

- Ambos cierres son equivalentes ya que:

$$A \in \alpha_F^+ \Leftrightarrow \{\alpha \rightarrow A\} \in F^+$$

## Normalización basada en dependencias: Preservación de dependencias

**Cálculo** del cierre de un conjunto de atributos  $\alpha$  en base a un conjunto de dependencias funcionales  $F$ :

Inicializamos  $\alpha_F^+ = \alpha$

Mientras que  $\alpha_F^+$  cambie

Si  $\beta \rightarrow \gamma \in F$  y  $\beta \subseteq \alpha_F^+$  entonces  $\alpha_F^+ = \alpha_F^+ \cup \gamma$

## Normalización basada en dependencias: Preservación de dependencias

### Ejemplo:

$R(A,B,C,D,E,F)$

$F = \{AB \rightarrow C, D \rightarrow EF, C \rightarrow A, BE \rightarrow C, BC \rightarrow D, CF \rightarrow BD, ACD \rightarrow B, CE \rightarrow AF\}$

¿ $BC^+$ ?



## Normalización basada en dependencias: Preservación de dependencias

**Ejemplo** (omitimos dependencias en subíndice):

$BC^+ = \{B, C\}$
$BC^+ = \{B, C, A\}$ por $C \rightarrow A$
$BC^+ = \{B, C, A, D\}$ por $BC \rightarrow D$
$BC^+ = \{B, C, A, D, E, F\}$ por $D \rightarrow EF$

# Normalización basada en dependencias: Preservación de dependencias

## Volviendo a nuestro ejemplo:

- Hemos partido la relación en dos relaciones y sus conjuntos de dependencias funcionales en otros dos conjuntos.
- Por el **Teorema de Heath**, se verifica que la descomposición es sin pérdidas, es decir, que la reunión natural de  $R_1$  y  $R_2$  tiene todos los datos pero ¿qué dependencias observa esa reunión?

## Normalización basada en dependencias: Preservación de dependencias

- Parece lógico pensar que observa todas las dependencias que hay en  $DF_1$  y  $DF_2$ , ...
- ... y todas las que se puedan deducir de ellas, es decir,  $(DF_1 \cup DF_2)^+$
- Entonces, para saber si hemos perdido  $\{almacén, producto\} \rightarrow dirección\_almacén$  ¿hemos de calcular todo  $(DF_1 \cup DF_2)^+$ ?

## Normalización basada en dependencias: Preservación de dependencias

- **No es necesario**, sino que basta con comprobar si  $\{almacén, producto\} \rightarrow dirección\_almacén$  pertenece a  $(DF_1 \cup DF_2)^+$

Y eso es fácil, porque **basta con comprobar si**  $dirección\_almacén$  **pertenece al** cierre de atributos  $\{almacén, producto\}^{+}_{DF_1 \cup DF_2}$

## Normalización basada en dependencias: Preservación de dependencias

- Calculando

$\{\text{almacén, producto}\}^+ = \{\text{almacén, producto}\}$
---

$\{\text{almacén, producto}\}^+ = \{\text{almacén, producto, cantidad}\}$ por $\{\text{almacén, producto}\} \rightarrow \text{cantidad}$
---

$\{\text{almacén, producto}\}^+ = \{\text{almacén, producto, cantidad, dirección\_almacén}\}$ por $\text{almacén} \rightarrow \text{dirección\_almacén}$
---

## Tercera forma normal

Decimos que una relación  $R$  está en tercera forma normal (3FN) sii para toda dependencia **no trivial**  $\alpha \rightarrow \beta$  se cumple que:

- $\alpha$  es una **superclave**, ó
- $\alpha$  contiene **solamente atributos primos**.

Esta definición implica inmediatamente 2FN.

La definición original de Codd de 3FN es equivalente y se basa en el concepto de Dependencia Transitiva Problemática (DTP):

## Dependencia transitiva problemática (DTP)

Sea  $R$  un esquema de relación,  $F$  un conjunto de DFs asociado y  $CK \subset P$  una clave candidata de  $P$ . Decimos que  $CK \rightarrow \beta$ , con  $\beta$  formado por algún atributo no primo, es una **DTP** si  $\exists \alpha \subset P$  tal que  $\alpha$  no es superclave y

$$CK \rightarrow \alpha \in F \quad \text{y} \quad \alpha \rightarrow \beta \in F$$

Es decir, la dependencia  $CK \rightarrow \beta$  que debe cumplirse por ser  $CK$  clave candidata se verifica también  $\alpha$  través de  $CK \rightarrow \alpha$ ,  $\alpha \rightarrow \beta$  y el axioma de transitividad de Armstrong.

La definición original de Codd estipula que una relación  $R$  está en tercera forma normal (3FN) sii

- Está en segunda forma normal, y
- No presenta dependencias transitivas problemáticas.



## Normalización basada en dependencias: Tercera forma normal

### Ejemplo:

La relación ASIGNATURA (#asig, nombre, curso, plan, ct, cp, coste), con #asig como clave primaria, presenta una dependencia funcional transitiva problemática:

$$\begin{aligned} \#asig &\rightarrow \text{nombre curso plan ct cp} \\ \text{plan ct cp} &\rightarrow \text{coste} \end{aligned}$$

Desde el punto de vista de la primera definición, no se cumple 3FN porque existe una dependencia cuya parte izquierda no es superclave (plan, ct, cp) y cuya parte derecha contiene un atributo no primo (coste).

## Normalización basada en dependencias: Tercera forma normal

### Ejemplo (cont.):

La dependencia causante de que la tabla no esté en 3FN es, independientemente de la definición usada, la dependencia  
**plan ct cp → coste**

**Normalización:** aplicando Teorema de Heath:

– $R_1$  (#asig, nombre, curso, plan, ct, cp)

PK: #asig, dependencia “directa” (no transitiva)

– $R_2$  (plan, ct, cp, coste)

PK: (plan, ct, cp), dependencia “directa” (no transitiva)

## Forma normal de Boyce-Codd

La definición de 3FN tiene deficiencias ya que no produce diseños satisfactorios cuando:

Hay varias claves candidatas

Las claves candidatas son compuestas

Las claves candidatas se solapan.

Para solventar este problema se introduce una variante de la 3FN que se denomina **forma normal de Boyce-Codd**, más restrictiva que la 3FN, aunque equivalente a ésta si no se dan las anteriores condiciones

## Normalización basada en dependencias: Forma normal de Boyce-Codd

Decimos que una relación  $R$  está en Forma normal de Boyce-Codd (FNBC) si y solo si para toda dependencia **no trivial**  $\alpha \rightarrow \beta$  se cumple que  $\alpha$  es una **superclave**.

Equivalentemente:

Dada una relación  $R$  y  $F$  su conjunto de DFs asociado, decimos que  **$R$  está en FNBC si y sólo si todo determinante es una clave candidata.**

Donde un **determinante de una relación** es todo conjunto de atributos del cual depende de forma completa otro atributo de la relación.

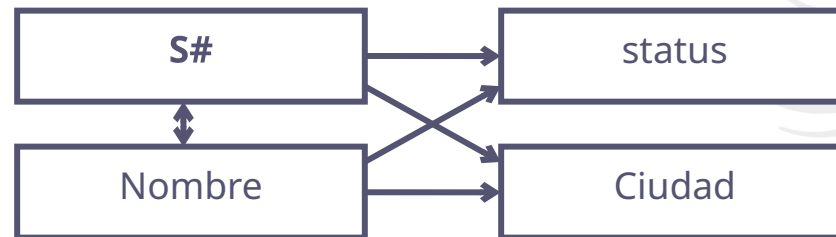
## Normalización basada en dependencias: Forma normal de Boyce-Codd

### Ejemplos

Consideremos la siguiente relación:

PROVEEDOR (S#, Nombre, Ciudad, Status)

donde S# y Nombre son claves candidatas y no se verifica la dependencia ciudad  $\rightarrow$  status



está en FNBC

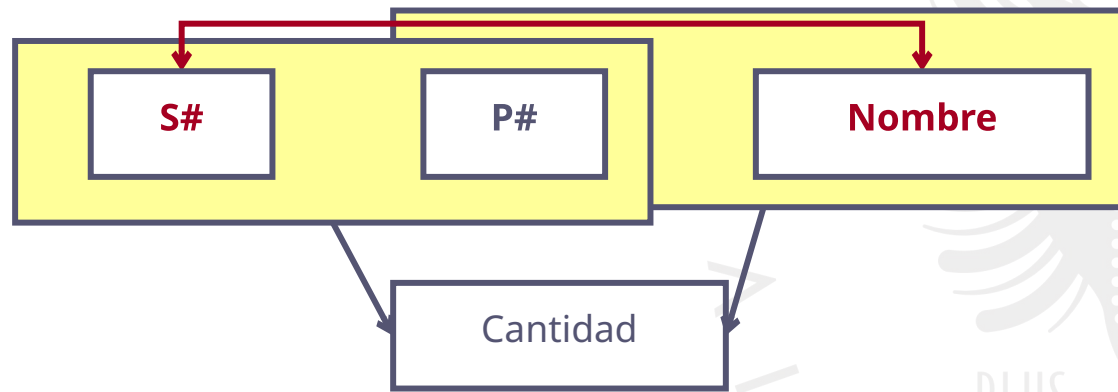
## Normalización basada en dependencias: Forma normal de Boyce-Codd

Consideremos la siguiente relación:

SSP (S#,Nombre,P#,Cantidad)

Claves Candidatas: (S#,P#), (Nombre,P#)

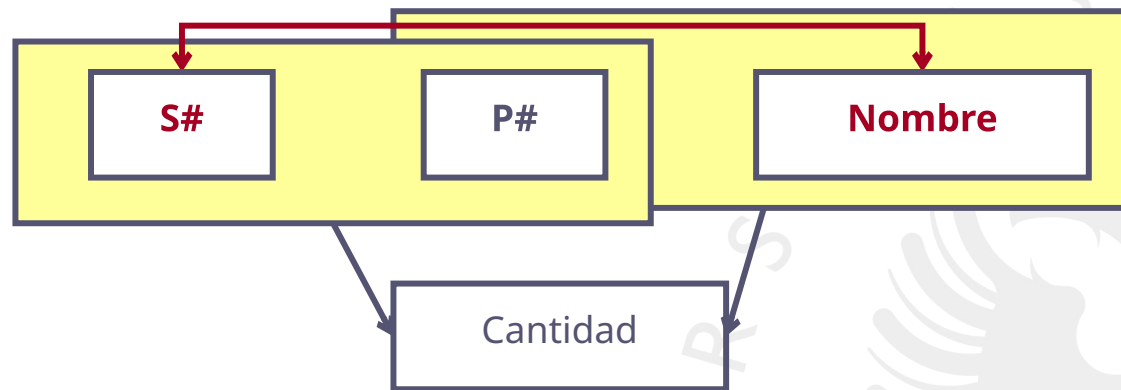
¿Está en FNBC?



**NO**, ya que hay dos determinantes (S# y Nombre) que no son CKs.

## Normalización basada en dependencias: Forma normal de Boyce-Codd

¿Está en 3FN?

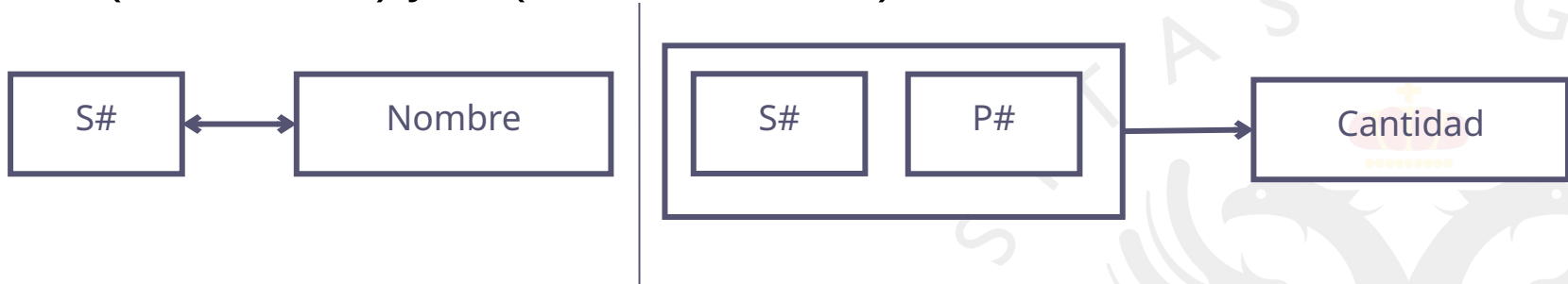


**SÍ**, porque todos los atributos no primos dependen de forma completa de las claves candidatas y no hay transitividad a través de atributos no primos. O visto de otro modo, porque para toda dependencia no trivial se cumple o bien que la parte izquierda es una superclave, o bien que la parte derecha está compuesta de atributos primos exclusivamente.

## Normalización basada en dependencias: Forma normal de Boyce-Codd

**Normalización:** dos alternativas

$S1(S\#, Nombre)$  y  $S2(S\#, P\#, Cantidad)$



$S1(S\#, Nombre)$  y  $S2(Nombre, P\#, Cantidad)$



**Ambas descomposiciones están en FNBC.**



## Normalización basada en dependencias: Forma normal de Boyce-Codd

Otro ejemplo:

EAP (Estudiante, Asignatura, Profesor)

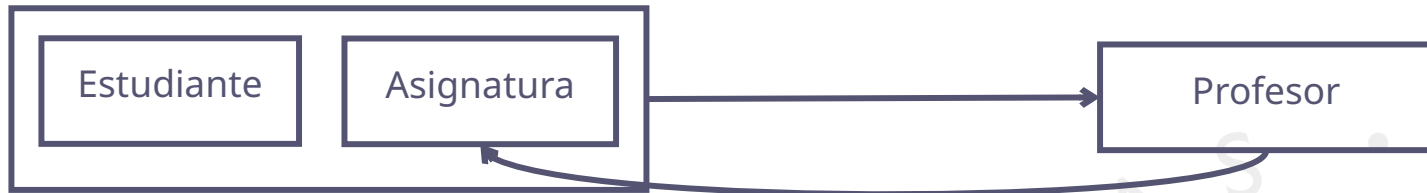
Cada estudiante tiene un único profesor por asignatura.

Cada profesor da una única asignatura, pero cada asignatura es impartida por varios profesores.

Diagrama de dependencias funcionales:



## Normalización basada en dependencias: Forma normal de Boyce-Codd



A la vista del diagrama, las claves candidatas son:  
(Estudiante, Asignatura) y (Profesor, Estudiante)

EAP **no está en FNBC**, pero **sí está en 3FN**.

**Normalización:**

EP(Estudiante,Profesor)

PA(Profesor,Asignatura)

**¿Es una buena solución?**

# Normalización basada en dependencias:

## Relación entre 3FN y FNBC

Toda relación en FNBC está en 3FN, ya que la primera es más estricta (no le importa si los atributos de la derecha son primos o no).  
Toda relación 3FN con una única clave candidata está en FNBC.  
Toda relación en 3FN con claves candidatas no solapadas está en FNBC.

Es siempre posible obtener una descomposición en 3FN sin pérdidas y que preserve dependencias.

Es siempre posible obtener una descomposición en FNBC sin pérdidas, **pero no siempre preservando dependencias**. Hay que decidir si conviene o no.

Si se pierden dependencias, el diseño lógico de la BD no puede garantizar el cumplimiento de dichas restricciones. Pueden implementarse mediante la inclusión de código en la BD, como veremos en la [Práctica 3](#).

## Algoritmo de Normalización hasta FNBC

**Input:** una relación  $R$  y un conjunto asociado de dependencias  $F$ .

**Output:** un conjunto de relaciones que forman una descomposición sin pérdidas de  $R$  y que están en FNBC con respecto al conjunto de dependencias  $F$ .

1. Tablas = {  $R$  }
2. Si existe  $\alpha \rightarrow \beta$  no trivial con  $\alpha, \beta \subseteq R$  tal que  $\alpha$  no es superclave:
  - 1.1 Tablas = Tablas \ { $R$ }.
  - 1.1 Aplicar Teorema de Heath: dividir  $R$  en
    - $R_1$  con esquema  $R \setminus \beta$
    - $R_2$  con esquema  $\alpha \cup \beta$
  - 1.2 Tablas = Tablas FNBC( $R_1, F$ ) // Llamada recursiva
  - 1.3 Tablas = Tablas FNBC( $R_2, F$ ) // Llamada recursiva
3. Return (Tablas)

# Algoritmo de Normalización hasta FNBC

Nótese que **es necesario conocer ó calcular las claves candidatas de R**, y por tanto en cada llamada recursiva

- **Pueden obtenerse distintos resultados válidos** dependiendo del orden en que se consideren las dependencias. Una buena estrategia es empezar por las dependencias que rompan 2FN, seguir con las que rompan 3FN, y finalizar por las que rompan FNBC.

Como ya hemos explicado, no se garantiza la preservación de dependencias.

El resultado estará en FNBC con respecto al conjunto de dependencias inicial, **pero no necesariamente con respecto a su cierre**.

Hay otros algoritmos en la literatura.

## Normalización basada en dependencias:

# Algoritmo de cálculo de claves candidatas

Dado el esquema  $R$  con un conjunto de dependencias funcionales  $F$ , el procedimiento para el cálculo de las claves candidatas es el siguiente:

### 1. Eliminación de atributos independientes:

Construir, a partir de  $R$ , un conjunto de atributos  $R_{si}$  en el que se han eliminado los atributos independientes, dado que estos participan en cualquier clave candidata y de ellos no se puede deducir ningún otro (salvo ellos mismos).

# Normalización basada en dependencias: Algoritmo de cálculo de claves candidatas

## 2. Eliminación de atributos equivalentes:

Construir, a partir de  $R_{si}$ , un conjunto de atributos  $R_{sie}$  en el que se han eliminado los atributos equivalentes, escogiendo uno de los dos atributos de cada equivalencia y sustituyendo el eliminado por el elegido en cada dependencia funcional de  $F$  en la que aparezca;

Como resultado de este paso, puede darse el caso de que determinados atributos aparezcan como independientes entre sí.



## Normalización basada en dependencias: Algoritmo de cálculo de claves candidatas

### 3. Selección de una clave de $R_{sie}$ en la que no aparecen determinantes que sean determinados:

Se selecciona como primer candidato a clave candidata  $K_p$  cualquier determinante de  $R_{sie}$  que no sea determinado



## Normalización basada en dependencias: Algoritmo de cálculo de claves candidatas

Si no quedan más determinantes en  $R_{sie}$  que sean a la vez determinados,  $K_p$  es clave candidata y se pasa al [paso 5](#). En caso contrario, se pasa al [paso 4](#).

## Normalización basada en dependencias: Algoritmo de cálculo de claves candidatas

### 4. Selección de una clave de $R_{sie}$ en el que pueden aparecer determinantes que puedan ser determinados:

Se construye el conjunto  $R'_{sie}$  eliminando de  $R_{sie}$  aquellos atributos que aparecen en  $K_p+$  y no están implicados en otras dependencias que no sean las necesarias para calcular  $K_p+$ .

Se obtiene una clave provisional  $K'_p$  en  $R'_{sie}$ , con los determinantes de  $K_p$  y añadiendo a estos un nuevo determinante que sea determinado. Si  $K'_p+ = R'_{sie}$ , entonces  $K'_p$  es una clave de  $R'_{sie}$ . En caso contrario, se añade un nuevo atributo que sea determinado y que no pertenezca al cierre de  $K'_p$  y se vuelve a comprobar.

Se repite la operación para cubrir todas las claves posibles.

Se añade a cada clave de  $R'_{sie}$  las obtenidas del paso 3 para obtener las claves de  $R_{sie}$ .

Si no se pudiese construir  $R'_{sie}$ , se procede considerando  $R_{sie}$  como  $R'_{sie}$ .

## Normalización basada en dependencias: Algoritmo de cálculo de claves candidatas

5. Añadir los atributos independientes a las claves obtenidas para  $R_{sie}$ .
6. Replicar las claves con las equivalencias eliminadas en el paso 2 para generar todas las claves.

## Normalización basada en dependencias: **recubrimiento minimal de dependencias**

Se llama **recubrimiento minimal o canónico** de un conjunto de dependencias funcionales  $F$  y se nota por  $F'$  al conjunto que cumple:

$$F^+ = (F')^+$$

es decir, que cualquier dependencia que se puede obtener a través de  $F$  se puede obtener a través de  $F'$ , pero  $F'$  está formada por dependencias con estructura mucho más simple que las de  $F$  (*aunque puede haber más dependencias en  $F'$  que en  $F$* ).

# Normalización basada en dependencias: recubrimiento minimal de dependencias

El proceso de obtención del recubrimiento minimal consiste en partir de  $F$  para simplificar las dependencias, simplificando:

- La parte derecha de la dependencia
- La parte izquierda de la dependencia
- La dependencia en sí

La obtención de  $F'$  se basa en un algoritmo con tres pasos.

## Normalización basada en dependencias: recubrimiento minimal de dependencias

Lo explicaremos con el mismo ejemplo anterior:

$R(A,B,C,D,E,F)$

$F = \{AB \rightarrow C, D \rightarrow EF, C \rightarrow A, BE \rightarrow C, BC \rightarrow D, CF \rightarrow BD, ACD \rightarrow B, CE \rightarrow AF\}$

## Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 1: obtención de  $F^{(1)}$  mediante aplicación de la regla de descomposición a todas las dependencias que tengan parte derecha compuesta.

$F^{(1)} = \{AB \rightarrow C, D \rightarrow E, D \rightarrow F, C \rightarrow A, BE \rightarrow C, BC \rightarrow D, CF \rightarrow B, CF \rightarrow D, ACD \rightarrow B, CE \rightarrow A, CE \rightarrow F\}$

Se ve claramente que si aplicamos la regla de unión sobre cada pareja de dependencias en rojo, volvemos a obtener la original.

## Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 2: obtención de  $F^{(2)}$  mediante simplificación de la parte izquierda de las dependencias eliminando *atributos raros*.

Sea una dependencia con la parte izquierda compuesta de la forma  $\alpha A \rightarrow B$ , se dice que  $A$  es *raro con respecto a  $\alpha$*  sii  $A \in \alpha^+$ , es decir, que  $A$  depende funcionalmente de los atributos que le acompañan.

Cada atributo raro que aparezca con respecto a los que le acompañan, se suprime.



# Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 2:

$AB \rightarrow C$ ,

$A^+ = \{A\}$ , B no pertenece a  $A^+$  luego B no es raro con respecto a A

$B^+ = \{B\}$ , A no pertenece a  $B^+$  luego A no es raro con respecto a B

luego  $AB \rightarrow C$  se queda como está.

...

## Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 2: ...

$BE \rightarrow C$ ,

$B^+ = \{B\}$ , E no pertenece a  $B^+$  luego E no es raro con respecto a B

$E^+ = \{E\}$ , B no pertenece a  $E^+$  luego B no es raro con respecto a E

luego  $BE \rightarrow C$  se queda como está.

...

## Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 2: ...

$BC \rightarrow D$ ,

$B^+ = \{B\}$ ,  $C$  no pertenece a  $B^+$  luego  $C$  no es raro con respecto a  $B$

$C^+ = \{C, A\}$ ,  $B$  no pertenece a  $C^+$  luego  $B$  no es raro con respecto a  $C$

luego  $BC \rightarrow D$  se queda como está.

...

## Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 2: ...

$CF \rightarrow B, CF \rightarrow D,$

$F^+ = \{F\}$ ,  $C$  no pertenece a  $F^+$  luego  $C$  no es raro con respecto a  $F$

$C^+ = \{C, A\}$ ,  $F$  no pertenece a  $C^+$  luego  $F$  no es raro con respecto a  $C$

luego  $CF \rightarrow B, CF \rightarrow D$  se quedan como están.

...

## Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 2: ...

$ACD \rightarrow B$ ,

$\{AC\}^+ = \{A, C\}$ , D no pertenece a  $\{AC\}^+$  luego D no es raro con respecto a  $\{AC\}$

$\{AD\}^+ = \{A, D, E, F\}$ , C no pertenece a  $\{AD\}^+$  luego C no es raro con respecto a  $\{AD\}$

$\{CD\}^+ = \{C, D, E, F, A, B\}$ , A pertenece a  $\{CD\}^+$  luego **A es raro con respecto a  $\{CD\}$**

luego  $ACD \rightarrow B$  se cambia por  $CD \rightarrow B$ , pero hay que seguir comprobando dentro de  $\{CD\}$

## Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 2: ...

$CD \rightarrow B$ ,

$C^+ = \{C, A\}$ , D no pertenece a  $C^+$  luego D no es raro con respecto a C

$D^+ = \{D, E, F\}$ , C no pertenece a  $D^+$  luego C no es raro con respecto a D

luego  $CD \rightarrow B$  queda como está.

...

# Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 2: ...

$CE \rightarrow A, CE \rightarrow F,$

$C^+ = \{C, A\}$ , E no pertenece a  $C^+$  luego E no es raro con respecto a C

$E^+ = \{E\}$ , C no pertenece a  $E^+$  luego C no es raro con respecto a E

luego  $CE \rightarrow A, CE \rightarrow F$  se quedan como están.

## Normalización basada en dependencias: recubrimiento minimal de dependencias

El resultado de este paso es:

$F^{(2)} = \{AB \rightarrow C, D \rightarrow E, D \rightarrow F, C \rightarrow A, BE \rightarrow C, BC \rightarrow D, CF \rightarrow B, CF \rightarrow D, \text{CD} \rightarrow \text{B}, CE \rightarrow A, CE \rightarrow F\}$



## Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 3: obtención de  $F^{(3)}$  o  $F'$  mediante eliminación de dependencias redundantes.

Una dependencia  $\alpha \rightarrow \beta \in F$  es redundante si se puede obtener a partir de las demás mediante aplicación de los axiomas y las reglas de Armstrong en una secuencia finita de pasos, es decir, sii:

$$\alpha \rightarrow \beta \in (F - \{ \alpha \rightarrow \beta \})^+$$

Difícil de comprobar

## Normalización basada en dependencias: recubrimiento minimal de dependencias

Paso 3: ...

Pero existe una relación entre el cierre de dependencias y el cierre de atributos, y éste último es más fácil de comprobar:

$$\alpha \rightarrow \beta \in (F - \{\alpha \rightarrow \beta\})^+ \Leftrightarrow \beta \in \alpha^+_{F - \{\alpha \rightarrow \beta\}}$$

Cada dependencia redundante se suprime.

## Normalización basada en dependencias: recubrimiento minimal de dependencias

$AB \rightarrow C$  es redundante si  $C \in \{AB\}^+_{F(2)-\{AB \rightarrow C\}}$

$-\{AB\}^+_{F(2)-\{AB \rightarrow C\}} = \{A, B\}, C \notin \{AB\}^+_{F(2)-\{AB \rightarrow C\}}$  luego  $AB \rightarrow C$  no es redundante y aparece en  $F^{(3)}$

$D \rightarrow E$  es redundante si  $E \in D^+_{F(2)-\{D \rightarrow E\}}$

$-D^+_{F(2)-\{D \rightarrow E\}} = \{D, F\}, E \notin D^+_{F(2)-\{D \rightarrow E\}}$  luego  $D \rightarrow E$  no es redundante y aparece en  $F^{(3)}$

## Normalización basada en dependencias: recubrimiento minimal de dependencias

$D \rightarrow F$  es redundante si  $F \in D^+_{F(2)-\{D \rightarrow F\}}$

–  $D^+_{F(2)-\{D \rightarrow F\}} = \{D, E\}$ ,  $F \notin D^+_{F(2)-\{D \rightarrow F\}}$  luego  $D \rightarrow F$  no es redundante y aparece en  $F^{(3)}$

$C \rightarrow A$  es redundante si  $A \in C^+_{F(2)-\{C \rightarrow A\}}$

–  $C^+_{F(2)-\{C \rightarrow A\}} = \{C\}$ ,  $A \notin C^+_{F(2)-\{C \rightarrow A\}}$  luego  $C \rightarrow A$  no es redundante y aparece en  $F^{(3)}$

## Normalización basada en dependencias: recubrimiento minimal de dependencias

$BE \rightarrow C$  es redundante si  $C \in \{BE\}^+_{F(2)-\{BE \rightarrow C\}}$

$-\{BE\}^+_{F(2)-\{BE \rightarrow C\}} = \{B, E\}, C \notin \{BE\}^+_{F(2)-\{BE \rightarrow C\}}$  luego  $BE \rightarrow C$  no es redundante y aparece en  $F^{(3)}$

$BC \rightarrow D$  es redundante si  $D \in \{BC\}^+_{F(2)-\{BC \rightarrow D\}}$

$-\{BC\}^+_{F(2)-\{BC \rightarrow D\}} = \{B, C, A\}, D \notin \{BC\}^+_{F(2)-\{BC \rightarrow D\}}$  luego  $BC \rightarrow D$  no es redundante y aparece en  $F^{(3)}$

## Normalización basada en dependencias: recubrimiento minimal de dependencias

$CF \rightarrow B$  es redundante si  $B \in \{CF\}^+_{F(2)-\{CF \rightarrow B\}}$

$-\{CF\}^+_{F(2)-\{CF \rightarrow B\}} = \{C, F, A, D, B, E\}$ ,  $B \in \{CF\}^+_{F(2)-\{CF \rightarrow B\}}$  luego

$CF \rightarrow B$  **es redundante** y no aparece en  $F^{(3)}$

$CF \rightarrow D$  es redundante si  $D \in \{CF\}^+_{F(2)-\{CF \rightarrow D\}}$

$-\{CF\}^+_{F(2)-\{CF \rightarrow D\}} = \{C, F, A\}$ ,  $D \notin \{CF\}^+_{F(2)-\{CF \rightarrow D\}}$  luego  $CF \rightarrow D$

no es redundante y aparece en  $F^{(3)}$

## Normalización basada en dependencias: recubrimiento minimal de dependencias

es redundante si  $B \in \{CD\}^+_{F(2)-\{CD \rightarrow B\}}$

$-\{CD\}^+_{F(2)-\{CD \rightarrow B\}} = \{C, D, E, F, A\}$ ,  $B \notin \{CD\}^+_{F(2)-\{CD \rightarrow B\}}$  luego

$CD \rightarrow B$  no es redundante y aparece en  $F^{(3)}$

$CE \rightarrow A$  es redundante si  $A \in \{CE\}^+_{F(2)-\{CE \rightarrow A\}}$

$-\{CE\}^+_{F(2)-\{CE \rightarrow A\}} = \{C, E, A, F, D, B\}$ ,  $A \in \{CE\}^+_{F(2)-\{CE \rightarrow A\}}$  luego

$CE \rightarrow A$  es redundante y no aparece en  $F^{(3)}$

## Normalización basada en dependencias: recubrimiento minimal de dependencias

$CE \rightarrow F$  es redundante si  $F \in \{CE\}^+_{F(2)-\{CE \rightarrow F\}}$

$-\{CE\}^+_{F(2)-\{CE \rightarrow F\}} = \{C, E, A\}, F \notin \{CE\}^+_{F(2)-\{CE \rightarrow F\}}$  luego  $CE \rightarrow F$   
no es redundante y aparece en  $F^{(3)}$



## Dependencias multivaluadas y 4FN

Sea  $R$  una relación y sean  $X, Y$  subconjuntos de  $R$ . Decimos que “ $X$  multidetermina  $Y$ ” o, equivalentemente, que hay una dependencia multivaluada  $X \twoheadrightarrow Y$  si para toda instancia  $r$  de  $R$ , dadas dos tuplas  $t, s$  de  $r$  que cumplen  $t[X] = s[X]$ , existen dos tuplas  $u, v$  en  $r$  tales que:

$$u[X] = v[X] = t[X] = s[X]$$

$$u[Y] = t[Y] \text{ y además } u[R-X-Y] = v[R-X-Y]$$

$$v[Y] = s[Y] \text{ y además } v[R-X-Y] = s[R-X-Y]$$

En otras palabras, si dado un valor de  $X$ , hay un conjunto de valores de  $Y$  asociados y este conjunto de valores de  $Y$  NO está relacionado (ni funcional ni multifuncionalmente) con los valores de  $R - X - Y$  (donde  $R$  es el esquema), es decir  $Y$  es independiente de los atributos de  $R - X - Y$ .

## Normalización basada en dependencias: Dependencias multivaluadas y 4FN

Almacén	Tipo	Tamaño
1	Tornillo	10
1	Tornillo	15
1	Tuerca	10
1	Tuerca	15
2	Arandela	10
2	Arandela	20

## Normalización basada en dependencias: Dependencias multivaluadas y 4FN

Si  $XY = R$  entonces  $X \twoheadrightarrow Y$  se cumple trivialmente.

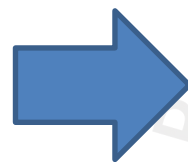
$X \twoheadrightarrow Y$  sii  $X \twoheadrightarrow R-X-Y$

Una relación  $R$  está en **4FN** si para toda dependencia multivaluada no trivial  $X \twoheadrightarrow Y$  con  $Y$  no incluida en  $X$  se cumple que  $X$  es una superclave de  $R$ .

Si  $R$  no está en **4FN** debido a una dependencia multivaluada  $X \twoheadrightarrow Y$ , la solución es dividir el esquema  $R$  en dos esquemas:  $XY$  por un lado y  $R-Y$  por otro, que formarán una descomposición sin pérdidas de  $R$ .

## Normalización basada en dependencias: Dependencias multivaluadas y 4FN

Almacén	Tipo	Tamaño
1	Tornillo	10
1	Tornillo	15
1	Tuerca	10
1	Tuerca	15
2	Arandela	10
2	Arandela	20



Almacén	Tipo
1	Tornillo
1	Tuerca
2	Arandela

Almacén	Tamaño
1	10
1	15
2	10
2	20

- ❑ Hay más dependencias y formas normales. La 5FN se define en términos de Dependencias de Reunión: una relación  $R$  está en 5FN sii está en 4FN y todas las dependencias de reunión son implicadas por claves candidatas.
- ❑ Para más información sobre dependencias y algoritmos para normalización: [Ullman, J.D. "Principles of Database and Knowledge-Base Systems" Vol. I. Computer Science Press, 1988.](#)