UNIVERSIDAD DE GRANADA

DECSAI

Diseño y Desarrollo de Sistemas de Información

Grado en Ingeniería Informática

Tema 4 – Diseño avanzado de Bases de Datos Relacionales

©I. J. Blanco, F. J. Cabrerizo, C. Cruz, M. J. Martín, D. Sánchez

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

Diseño Lógico Relacional: conceptos

Esquema relacional: conjunto de relaciones en el Modelo Lógico de Datos Relacional, conectadas entre sí, que permiten almacenar la información y mantener la semántica relacionadas con un sistema dado.

- **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.**

- Normalización: Proceso de refinamiento del diseño lógico propio del modelo relacional
- Objetivos:
- Corregir defectos del modelo conceptual
- Eliminar redundancias, problemas de actualización
- Plasmar restricciones semánticas adicionales

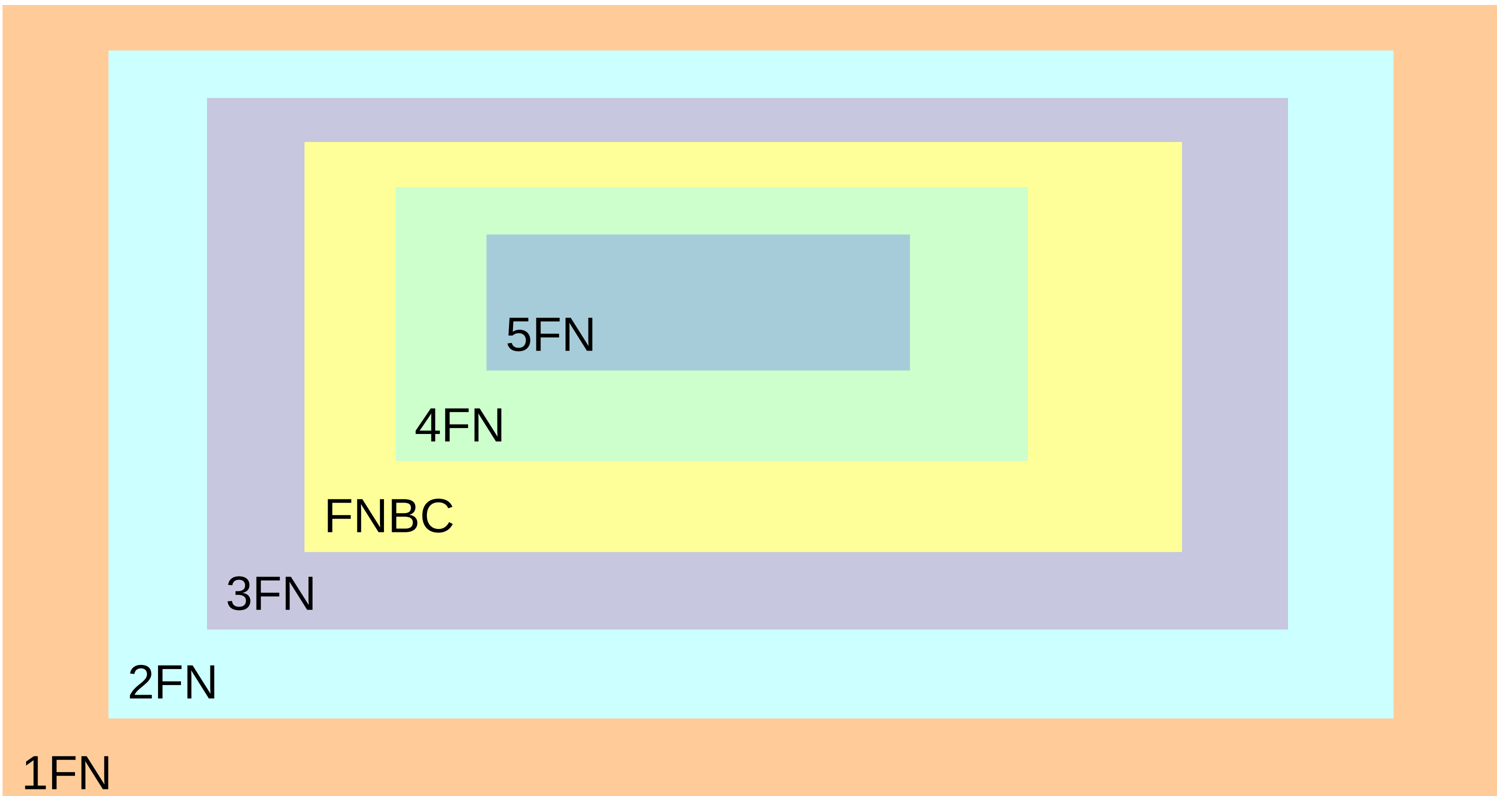
- Tratamos de conseguir un diseño relacional de una base de datos que cumpla una serie de características, lo que garantiza su buen comportamiento.
- Las buenas propiedades que cumple una relación se denomina **forma normal**.

DECSAI

Normalización: formas normales

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

Relación entre las formas normales



Normalización basada en dependencias: definiciones

- Dependencia funcional:

$$R(A_1, A_{2,\ldots}, A_n), \alpha \subset R, \beta \subset R, \alpha \rightarrow \beta \, sii$$
$$\forall t, s \in r, t[\alpha] = s[\alpha] \rightarrow t[\beta] = s[\beta]$$

Normalización basada en dependencias: definiciones

- **Dependencia funcional:**

- Dada una relación *R* con n atributos y dos subconjuntos de atributos de *R* llamados α y β, se dice que α determina funcionalmente a β o que β depende funcionalmente de α si para cualquier pareja de tuplas *s* y *t* de la instancia *r* de la relación, que tengan iguales valores para los atributos del subconjunto α se verifica que tienen los mismos valores para los atributos del subconjunto β.

Normalización basada en dependencias: definiciones

- **Dependencia funcional:**

- Las dependencias funcionales establecen restricciones semánticas que deben verificarse.
- Una forma de conseguirlo es mediante un diseño adecuado de la base de datos, es decir, determinar los esquemas de tablas adecuados.
- Un conjunto de dependencias funcionales en una tabla sirve asimismo para determinar TODAS las claves candidatas.

Normalización basada en dependencias: definiciones

- Dependencia funcional completa:

$$R(A_1, A_{2,\ldots}, A_n), \alpha \subset R, \beta \subset R, \alpha \rightarrow\rightarrow \beta \mathit{sii}$$
$$\alpha \rightarrow \beta \wedge \nexists \gamma \subset \alpha \mid \gamma \rightarrow \beta$$

Normalización basada en dependencias: definiciones

- **Dependencia funcional completa:**

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

Normalización basada en dependencias: definiciones

- **Atributo primo:**
- Se llama así a aquel atributo que forma parte de una clave candidata.

Normalización basada en dependencias: 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.

Normalización basada en dependencias: segunda forma normal

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

Normalización basada en dependencias: segunda forma normal

Ejemplo:

STOCK (<u>#almacén, #producto</u>, 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:

$$DF = \left\{ \begin{array}{c} \{almacén, producto\} \rightarrow cantidad, \{almacén, producto\} \rightarrow direccionalmacen, \\ almacén \rightarrow direccionalmacen \end{array} \right.$$

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 (R1,r1) y (R2,r2), se dice que la descomposición es sin pérdidas sii:

$$R_1 \cup R_2 = R$$
$$r_1 \, JOIN \, r_2 = r$$

Normalización basada en dependencias: segunda forma normal

- En nuestro ejemplo, 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.

Normalización basada en dependencias: segunda forma normal

- Ejemplo (descomposición sin pérdidas):

$$R_1 = \{almacén, producto, cantidad\}$$
$$DF_1 = \{\{almacén, producto\} \rightarrow cantidad\}$$
$$R_2 = \{almacén, direcciónalmacén\}$$
$$DF_2 = \{almacén \rightarrow direcciónalmacén\}$$

Normalización basada en dependencias: 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 {#almacén, producto}$\rightarrow$dirección_almacén que parece haber se perdido.
- ¿Se ha perdido realmente?

Normalización basada en dependencias: preservación de dependencias

- ¿Se ha perdido realmente?

- La respuesta no es tan sencilla porque, el hecho de la definición de dependencia funcional hace que se deriven una serie de axiomas y reglas que nos permiten operar con ellas. Se conocen como los **Axiomas de Armstrong**.
- A lo mejor, podemos recuperar lo que supuestamente se ha perdido a partir de las dependencias que quedan.

Normalización basada en dependencias: preservación de dependencias

Axiomas:

- Reflexividad:

$$\forall \alpha, \beta \mid \beta \subseteq \alpha \textit{ se verifica } \alpha \to \beta$$

- Ampliación:

$$\forall \alpha, \beta, \gamma \mid \alpha \to \beta \textit{ se verifica } \alpha\gamma \to \beta\gamma$$

- Transitividad:

$$\forall \alpha, \beta, \gamma \mid \alpha \to \beta \wedge \beta \to \gamma \textit{ se verifica } \alpha \to \gamma$$

Normalización basada en dependencias: preservación de dependencias

Reglas:

- **Unión:**

$$\forall \alpha, \beta, \gamma \mid \alpha \to \beta \wedge \alpha \to \gamma \quad \textit{se verifica} \quad \alpha \to \beta\gamma$$

- **Descomposición:**

$$\forall \alpha, \beta, \gamma \mid \alpha \to \beta\gamma \quad \textit{se verifica} \quad \alpha \to \beta \wedge \alpha \to \gamma$$

- **Pseudotransitividad:**

$$\forall \alpha, \beta, \gamma, \delta \mid \alpha \to \beta \wedge \beta\gamma \to \delta \quad \textit{se verifica} \quad \alpha\gamma \to \delta$$

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 α en base a un conjunto de dependencias funcionales F:

- Se nota por α^+ y representa el conjunto de todos los atributos que son determinados por los atributos de α en conjunto mediante dependencias de F^+
- Son equivalentes:

$$\alpha \rightarrow A \in F^+$$

$$A \notin \alpha_F^+ \quad \textit{implica que} \quad \alpha \rightarrow A \notin F^+$$

Normalización basada en dependencias: preservación de dependencias

Cálculo de α^+:

- Inicializamos $\alpha^+=\alpha$
- Mientras que α^+ cambie
  - Si $\beta \rightarrow \gamma \in F$ y $\beta \in \alpha^+ \Rightarrow \gamma \in \alpha^+$

Normalización basada en dependencias: preservación de dependencias

Cálculo de α^+ (un 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

Cálculo de α^+ (un ejemplo):

$BC^+=\{B, C\}$
$BC^+=\{B, C, A\}$ 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}→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 $(DF_1 \text{ U } DF_2)^+$

- Y eso es fácil, porque basta con comprobar si dirección_almacén pertenece al cierre de atributos *{#almacén, producto}*$^+_{DF_1 \, U \, DF_2}$

Normalización basada en dependencias: preservación de dependencias

- Calculando:

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

- Dado que *dirección_almacén* está en el cierre de atributos, se puede deducir de ellos y la dependencia existe, por lo que no se ha perdido.

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*.

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)

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

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* α 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→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→B se cambia por CD→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:

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

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

- Paso 3:
- $AB \rightarrow C$ 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

- Paso 3:
- $D \rightarrow F$ 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

- Paso 3:
- $BE \rightarrow C$ 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

- Paso 3:
- $CF \rightarrow B$ 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

- Paso 3:
- $CD \rightarrow B$ 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

- Paso 3:
- $CE \rightarrow F$ 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 $F^{(3)}$

- El resultado es pués:

$$F^{(3)} = \{AB \rightarrow C,\ D \rightarrow E,\ D \rightarrow F,\ C \rightarrow A,\ BE \rightarrow C,\ BC \rightarrow D,\ CF \rightarrow D,\ CD \rightarrow B,\ CE \rightarrow F\} = F'$$

Normalización basada en dependencias: Dependencia funcional transitiva

Sea R un esquema de relación, F un conjunto de DFs asociado y $CK \subset R$ una clave candidata de R. Decimos que $CK \rightarrow \beta$, con β formado por algún atributo no primo, es transitiva si $\exists \alpha \subset R \mid \alpha \rightarrow R \notin F^{+}$ tal que

$$CK \rightarrow \alpha \in F \quad y \quad \alpha \rightarrow \beta \in F$$

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

Normalización basada en dependencias: tercera forma normal

Decimos que una relación R está en tercera forma normal (3FN) sii

- Está en segunda forma normal.
- 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:

- #asig $\rightarrow$ nombre curso plan ct cp
- plan ct cp $\rightarrow$ coste

que es el origen de su “mal comportamiento”.

Normalización: Descomposición sin pérdidas

- 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)

Normalización basada en dependencias: forma normal de Boyce-Codd

- La definición original de 3FN dada por Codd tiene deficiencias ya que no produce diseños satisfactorios cuando:
  - Hay varias claves candidatas
  - Las claves candidatas son compuestas
  - Las claves candidatas se solapan.

Normalización basada en dependencias: forma normal de Boyce-Codd

- La FNBC considera estos casos:

más restrictiva que la 3FN, aunque equivalente a ésta si no se dan las anteriores condiciones.

- Definición formal:

  Dada una relación R y F su conjunto de DFs asociado, decimos que R está en forma normal de Boyce y Codd (FNBC) si y sólo si $\forall \alpha \to \beta \in F$ se verifica:

- α es llave candidata y $\beta \not\subset \alpha$

o bien

- $\beta \subseteq \alpha$

Normalización basada en dependencias: forma normal de Boyce-Codd

- Definición:
  - **Determinante de una relación**: Todo conjunto de atributos del cual depende de forma completa otro atributo de la relación.
- FNBC, definición alternativa:
  - Dada una relación R y F su conjunto de DFs asociado, decimos que R está en forma normal de Boyce y Codd (FNBC) si y sólo si todo determinante es una clave candidata.

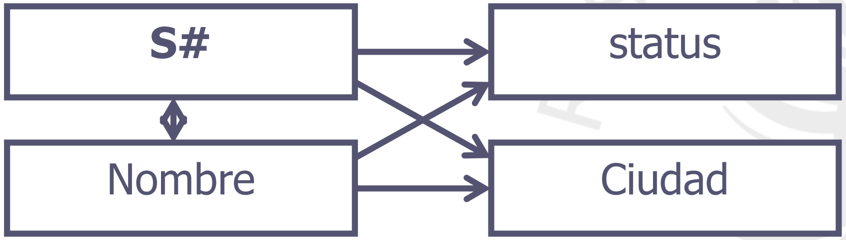
Normalización basada en dependencias: forma normal de Boyce-Codd

Ejemplos

- Consideremos la siguiente relación:

  PROVEEDOR (S#, Nombre, Ciudad, Status)

- Donde S# y Nombre son claves candidatas.
- No se verifica la dependencia ciudad$\rightarrow$status



La relación PROVEEDOR está en FNBC.

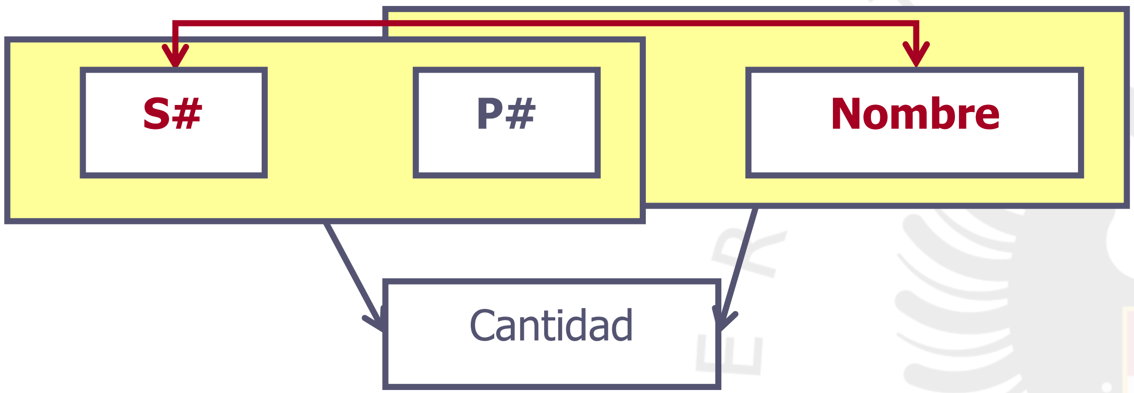
Normalización basada en dependencias: forma normal de Boyce-Codd

- Consideremos la siguiente relación:

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

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

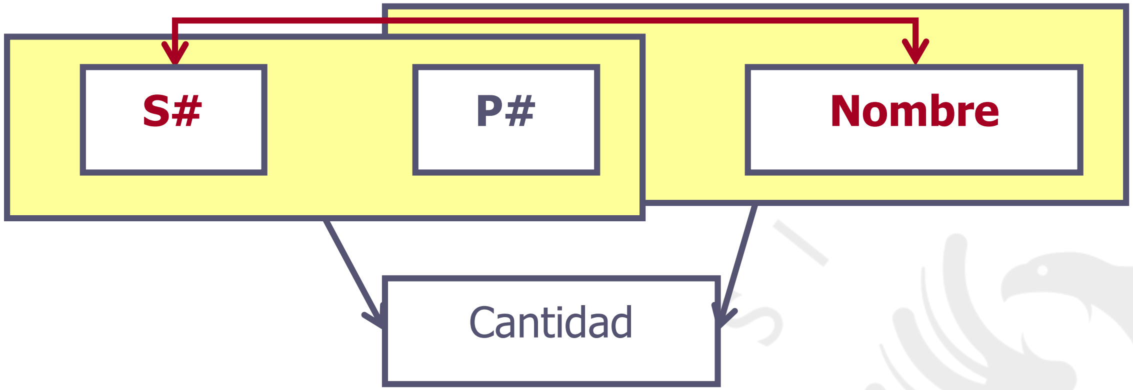
  ¿Está en FNBC?



NO, ya que hay dos determinantes que no son CKs.

Normalización basada en dependencias: forma normal de Boyce-Codd

¿Está en 3FN?



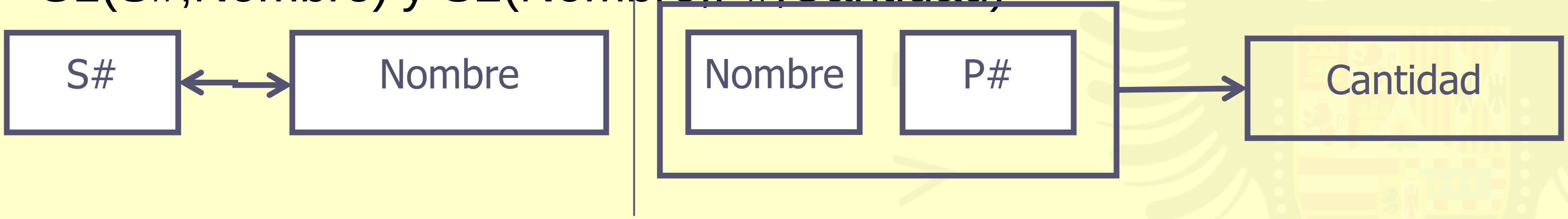
SÍ, porque todos los atributos no primos dependen de forma completa de las claves candidatas y no hay transitividad a través de atributos no primos.

Normalización basada en dependencias: forma normal de Boyce-Codd

.Normalización:

–S1(S#,Nombre) y S2(S#,P#,Cantidad)

–S1(S#,Nombre) y S2(Nombre,P#,Cantidad)



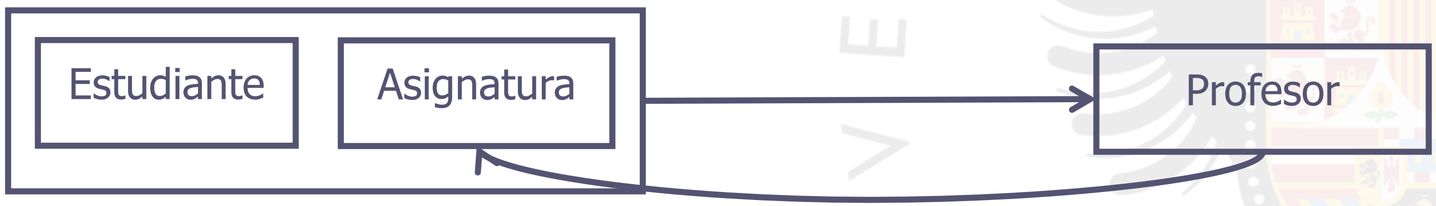
Ambas descomposiciones están en FNBC.

Normalización basada en dependencias: forma normal de Boyce-Codd

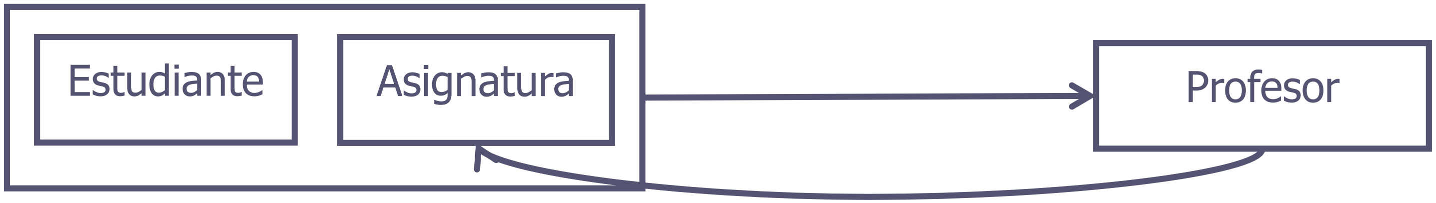
Otro ejemplo:

- 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)
  - (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
- Toda relación 3FN con una única llave candidata está en FNBC.
- Toda relación en 3FN con llaves 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.

Normalización basada en dependencias: algoritmo de cálculo de llaves candidatas

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

1. Eliminación de atributos independientes:

2. 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 llaves 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 llaves candidatas

3. Selección de una clave de Rsie en la que no aparecen determinantes que sean determinados:

- Se selecciona como primer candidato a clave candidata Kp cualquier determinante de Rsie que no sea determinado;
  - Si no quedan más determinantes en Rsie que sean a la vez determinados, Kp 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 llaves candidatas

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

a) 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+.

I. 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.

II. Se repite la operación para cubrir todas las claves posibles.

III. Se añade a cada clave de R'sie las obtenidas del paso 3 para obtener las claves de Rsie.

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

Normalización basada en dependencias: algoritmo de cálculo de llaves candidatas

5. Añadir los atributos independientes a las claves obtenidas para Rsie.
6. Replicar las claves con las equivalencias eliminadas en el paso 2 para generar todas las claves.