Bases de Datos

Tema 7 (parte 1)

Teoría de la Normalización

Francisco Ruiz

abr-2001

documentación preparada con ayuda de Esperanza Marcos (Universidad Rey Juan Carlos) y Mario Piattini (Universidad de Castilla-La Mancha)

Tema 7 (parte 1)

Teoría de la Normalización

Complementar con:

** capítulos 4, 5 y 6 del libro "Diseño de Bases de Datos Relacionales". De Miguel, A.; Piattini, M.; Marcos, E.; Ra-Ma, 1999.*

- *Conocer los conceptos y algoritmos principales de la teoría de la normalización.*
- *Aprender a optimizar esquemas relacionales utilizando esta teoría.*

- **Principales:**
  - [de Miguel et al, 1999]
    - Caps 4, 5 y 6
    - De Miguel, A.; Piattini, M.; Marcos, E.; Diseño de Bases de Datos Relacionales. Ra-Ma, 1999.

- **Otras:**
  - Elmasri, R.; Navathe, S.B.; Sistemas de Bases de Datos: Conceptos fundamentales (2ª edición). Addison-Wesley, 1997. Capítulos 12 (dependencias funcionales y normalización) y 13 (algoritmos de diseño).

Índice (1)

- Dependencias:
  - son propiedades inherentes al contenido semántico de los datos;
  - son un tipo especial de restricción de usuario en el modelo relacional, que afecta únicamente a los atributos dentro de una única relación; y
  - se han de cumplir para cualquier extensión de un esquema de relación.

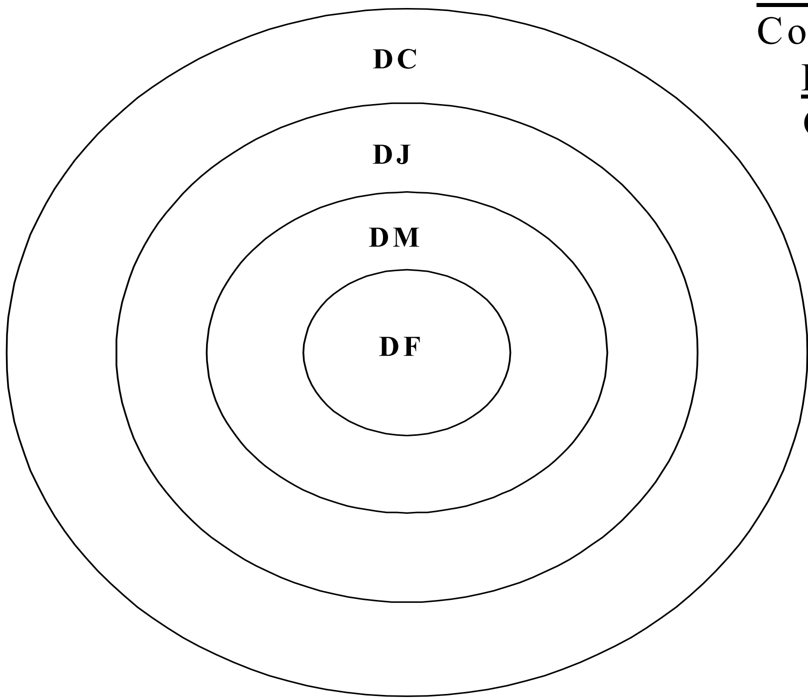
- A fines de simplificación, se considera que un esquema de relación es un par de la forma:

  R (A, DEP)

  - donde:
    - A es el conjunto de atributos de la relación, y
    - DEP es el conjunto de dependencias existentes entre los atributos.

- Existen distintos tipos de dependencias:
  - Funcionales (DF),
  - Multivaluadas (DM),
  - Jerárquicas (DJ), y
  - de Combinación (DC) (también llamadas producto).
- Cada tipo de dependencia se caracteriza por ser una asociación particular entre los datos.
- El grupo más restrictivo (y también más numeroso) es el de las dependencias funcionales. Sobre este conjunto de dependencias, se apoyan las tres primeras formas normales y la forma normal de Boyce_Codd.

- Cada tipo de dependencia es un caso particular del grupo que le sigue:



DF (Dependencias Funcionales)
Como caso particular de:
  DM (Dependencias Multivaluadas)
  Como caso particular de:
    DJ (Dependencias Jerárquicas)
    Como caso particular de:
      DC (Dependencias de Combinación)

- Definición de DF:
  - Sea el esquema de relación R(A, DF) y sean X e Y dos descriptores (subconjuntos de atributos de A). Se dice que existe una DF entre X e Y, de forma que X determina a Y, si y sólo si se cumple que para cualesquiera dos tuplas de R, u y v tales que u[X] = v[X], entonces necesariamente u[Y] = v[Y].
  - Esto significa que a cada valor x del atributo X, le corresponde un único valor y del atributo Y.
- Determinante:
  - Un **determinante** o implicante es un conjunto de atributos del que depende funcionalmente otro conjunto de atributos al que llamamos determinado o **implicado**.
- Ejemplo:
  - El código de estudiante determina el nombre del mismo:

$$Cód\_Estudiante \rightarrow Nombre$$

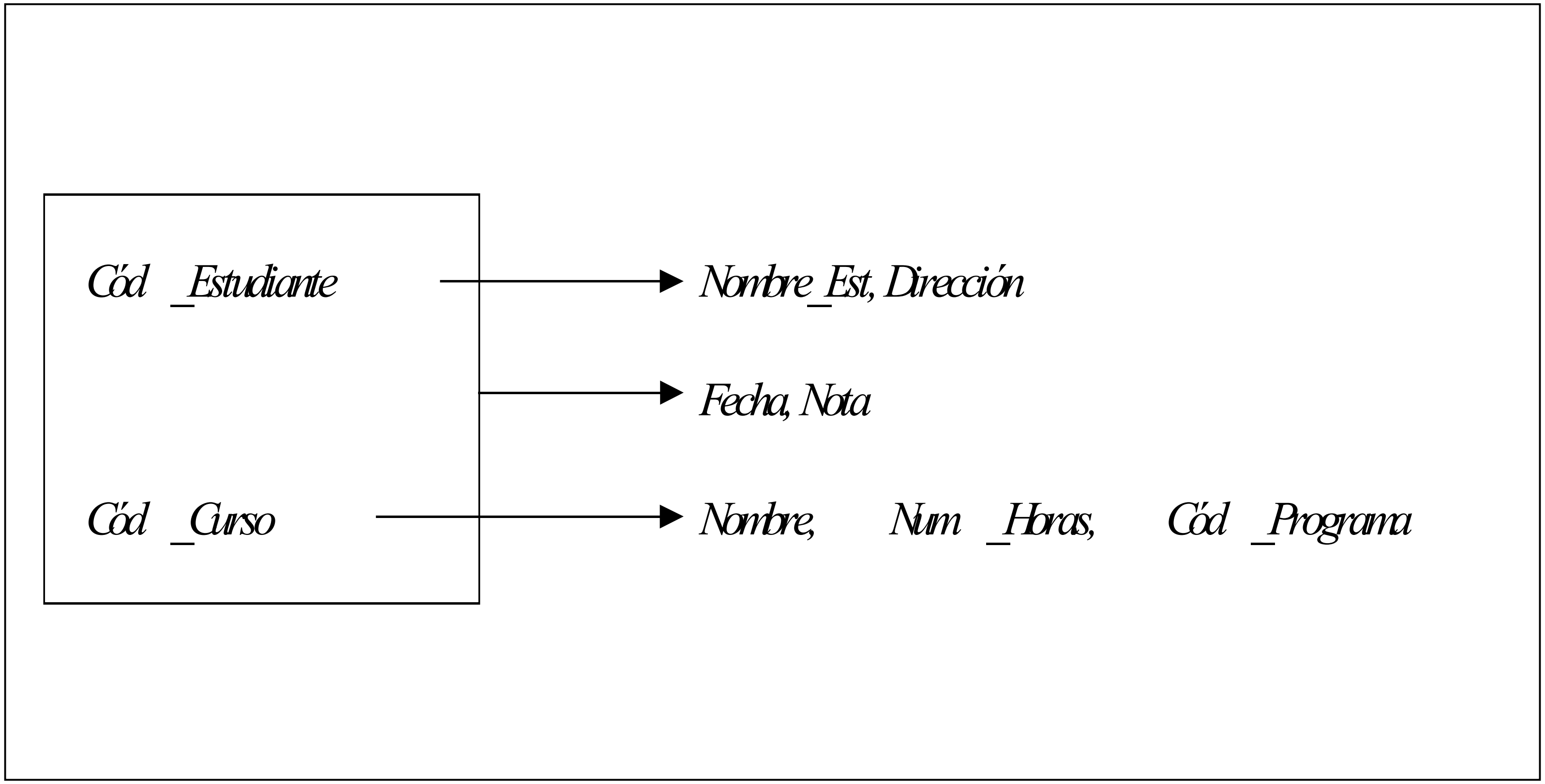
- <u>Descriptores equivalentes</u>:
  - Dos descriptores X e Y se dice que son equivalentes si

    $X \rightarrow Y \wedge Y \rightarrow X$
  - también se puede representar como:

    $X \longleftrightarrow Y$
- Ejemplo:
  - Los atributos Cód_Estudiante y DNI son equivalentes (se supone que dos alumnos distintos no pueden tener ni el mismo código ni el mismo DNI), es decir:

    Cód_Estudiante $\longleftrightarrow$ *DNI*

- <u>Grafo de dependencias</u>: Representa un conjunto de atributos y las DF existentes entre ellos. Es una herramienta muy útil a la hora de explicitar las DF.



- Se dice que Y tiene ***dependencia funcional plena*** *o completa* de X, si depende funcionalmente de X, pero no depende de ningún subconjunto de X.
  - Se representa por $X \Rightarrow Y$.
  - Por tanto,

$$X \Rightarrow Y \text{ sii } \neg\exists X' \subset X \mid X' \rightarrow Y$$

- Ejemplo: en la relación

  SE_MATRICULA (Cód_Curso, Cód_Edición, Cód_Estudiante, Nota)

  - La DF plena *Cód_Curso, Cód_Edición, Cód_Estudiante ⇒ Nota*
    - refleja que la nota la obtiene un estudiante en una edición determinada de un curso determinado.

- <u>Atributo extraño</u>: son los atributos del determinante de una DF que hacen que ésta no sea plena. También se llaman ajenos o extraños. Ejemplo según la figura T11:
  - La DF *Cód_Estudiante, Cód_Curso ⇒ Cód_Programa*
  - Es no plena y *Cód_Estudiante* es un atributo ajeno.

- DF **trivial**:
  - Una DF X → Y es ***trivial*** si Y es un subconjunto de X ($Y \subseteq X$).
  - Ejemplo: las siguientes DF son triviales:

    Cód_Estudiante → Cód_Estudiante

    Cód_Curso, Cód_Edición → Cód_Curso

- DF **elemental**:
  - Decimos que una DF X → Y es ***elemental*** si
    - Y es un atributo **único** no incluido en X, y
    - no existe X' incluido en X tal que X' → Y.
  - Una DF elemental es una DF plena, no trivial y en la que el implicado es un atributo único:
    - $X \rightarrow Y$ es elemental sii $(\neg\exists Y'\subset Y) \wedge (Y \subseteq X) \wedge (\neg \exists X'\subset X \mid X' \rightarrow Y)$
  - Únicamente las DF elementales son útiles para la normalización. El resto de DF no interesan y no se tienen en cuenta.

- Dado el esquema de relación

$$R\ (X, Y, Z)$$

en el que se cumple que

$$X \rightarrow Y\ ,\ \ Y \rightarrow Z\ ,\ \ Y \not\longrightarrow X$$

se dice que Z tiene una **DF transitiva** respecto a X a través de Y.

  – Se representa por

$$X \longrightarrow\rightarrow Z$$

  – Notar que X e Y no tienen que ser equivalentes

- DF transitiva **estricta**:
  – Es cuando además de las condiciones anteriores, también se cumple que

$$Z \not\longrightarrow Y$$

- Ejemplo de DF transitiva:
  - Dada la relación

    CURSO_PROGRAMA(Cód_Curso, Cód_Programa, Cód_Departamento)

  - en donde se tiene para cada curso su código, el programa que lo incluye y el departamento del que depende el programa (suponemos que un curso se imparte en un único programa y que un programa lo prepara un único departamento) se tendrán las siguientes DF:

$$Cód\_Curso \rightarrow Cód\_Programa$$

$$Cód\_Programa \rightarrow Cód\_Departamento$$

  - Además, como en un programa se imparten varios cursos:

$$Cód\_Programa \not\longrightarrow Cód\_Curso$$

  - y por tanto, se cumple la DF transitiva

$$Cód\_Curso \dashrightarrow Cód\_Departamento$$

que también es estricta porque

$$Cód\_Departamento \not\longrightarrow Cód\_Programa$$

- El conocimiento de ciertas DF puede llevar a inferir la existencia de otras que no se encontraban en el conjunto inicial:
  - Dado un esquema de relación: R (A, DF)

    es posible deducir de DF nuevas dependencias funcionales que sean una consecuencia lógica del conjunto de partida.
  - Las nuevas dependencias f que se cumplen para cualquier extensión de r de R son **consecuencia lógica** de DF (vienen implicadas por DF). Se representan como:

    $$DF \models f$$
- Ejemplo:
  - Dado el esquema de relación

    SOLICITA ({Cód_Estudiante, DNI_Est, Cód_Beca, Fecha_Solicitud}, DF)

    donde: DF = { *Cód_Estudiante* → *DNI_Est* , *DNI_Est* → *Cód_Estudiante* , *Cód_Estudiante, Cód_Beca* → *Fecha_Solicitud* }

    se cumple que

    DF |= *DNI_Est, Cód_Beca* → *Fecha_Solicitud*

- El **cierre** de un conjunto de dependencias funcionales DF (que se denota DF^+) es el conjunto de todas las dependencias que son consecuencia lógica de DF:

$$DF^+ = \{ X \rightarrow Y \mid DF \models X \rightarrow Y \}$$

  – DF será siempre un subconjunto del cierre ($DF \subseteq DF^+$). Por lo tanto, las notaciones R(A, DF) y R(A, DF^+) definen el mismo esquema de relación.

- Estas definiciones no permiten el cálculo del cierre, siendo necesarias unas <u>reglas de derivación</u> que faciliten la implicación lógica de dependencias.
  – Estas reglas de derivación, se conocen como **Axiomas de Armstrong**, y forman un conjunto completo y correcto de axiomas.

- Dado un conjunto DF de dependencias funcionales, se dice que f se **deriva** de DF, lo que se representa por

$$DF \vdash f$$

si f se puede obtener por aplicación sucesiva de los axiomas de Armstrong a DF (o a un subconjunto de DF), es decir, si existe una secuencia de dependencias f_1, f_2, … f_n tal que $f_n = f$, donde cada f_i es bien un elemento de DF o ha sido derivada a partir de las dependencias precedentes aplicando las reglas de derivación.

- Aunque son conceptos distintos, se cumple siempre que si una dependencia f es una **consecuencia lógica** de un conjunto de dependencias, también será posible **derivarla** de dicho conjunto aplicando los axiomas de Armstrong, y viceversa; es decir:

$\forall f \mid DF \vdash f$ se_implica_que $DF \models f$ (propiedad de corrección)

y

$\forall f \mid DF \models f$ se_implica_que $DF \vdash f$ (propiedad de plenitud)

Axiomas Básicos

- **A1: Reflexividad**

  Si $Y \subseteq X$, $X \rightarrow Y$ ($X \rightarrow Y$ es una DF trivial)
- **A2: Aumentatividad**

  Si $X \rightarrow Y$ y $Z \subseteq W$, entonces $XW \rightarrow YZ$
- **A3: Transitividad**

  Si $X \rightarrow Y$ e $Y \rightarrow Z$, entonces $X \rightarrow Z$

Axiomas Derivados

- **D1: Proyectividad**

  Si $X \rightarrow Y$, entonces $X \rightarrow Y'$ si $Y' \subset Y$
- **D2: Unión a aditividad**

  Si $X \rightarrow Y$ y $X \rightarrow Z$, entonces $X \rightarrow YZ$
- **D3: Pseudotransitividad**

  Si $X \rightarrow Y$ e $YW \rightarrow Z$, entonces $XW \rightarrow Z$

- Ejemplo de aplicación:
  - Dado el esquema de relación:

    $$R(A, B, C, D, E; A \rightarrow B, C \rightarrow D, D \rightarrow E)$$

  - Demostrar que $AC \rightarrow ABCDE$

    Se demuestra aplicando los axiomas de Armstrong de la siguiente forma:

1. $A \rightarrow B$ (dada)
2. $AC \rightarrow ABC$ (aumentatividad de la anterior por AC)
3. $C \rightarrow D$ (dada)
4. $D \rightarrow E$ (dada)
5. $C \rightarrow E$ (transitividad de 3 y 4)
6. $C \rightarrow DE$ (unión de 3 y 5)
7. $ABC \rightarrow ABCDE$ (aumentatividad de 6 por ABC)
8. $AC \rightarrow ABCDE$ (transitividad de 2 y 7)

- Aunque los axiomas de Armstrong facilitan un procedimiento algorítmico para calcular el cierre DF^+ de un conjunto de dependencias, su cálculo consume mucho tiempo, ya que, aunque el número inicial de dependencias sea pequeño, el número total de dependencias en el cierre es muy elevado.
- Para evitar este problema habrá que buscar procedimientos algorítmicos que no estén basados en el cierre de un conjunto de dependencias.
- Por otro lado, no todas las dependencias incluidas en el cierre son útiles en el proceso de diseño de una base de datos, por lo cual se introducirá el concepto de **recubrimiento** o **cobertura irredundante** también llamado **minimal**.

- Dado un esquema de relación R(A, DF), se denomina **superclave** SK de la relación R a un subconjunto no vacío de A, tal que SK → A es una consecuencia lógica de DF, siendo, por tanto, un elemento de su cierre:

$$(SK \neq \emptyset) \Lambda (SK \rightarrow A \in DF^{+)}$$

  – Esta condición se conoce como propiedad de **unicidad**.
  – Significa que una superclave determina a todos los atributos de la relación.

- Ejemplo: para la relación de la T20

$$R(A, B, C, D, E; A \rightarrow B, C \rightarrow D, D \rightarrow E)$$

  – el atributo A es superclave porque

$$AC \rightarrow ABCDE \quad \text{(todos los atributos)}$$

- Dado un esquema de relación R(A, DF), K es una **clave candidata** de R si, además de ser una superclave, no existe ningún subconjunto estricto K' de K tal que K' implique también a A:

$$(K \neq \emptyset) \wedge (K \rightarrow A \in DF^{+}) \wedge (\neg \exists K' \subset K \mid K' \rightarrow A)$$

  - Esta condición se conoce como propiedad de **minimalidad**.
  - Significa que una clave candidata tiene como determinante al conjunto mínimo de atributos necesario.

- Ejemplo: para la misma relación de la T20

$$R(A, B, C, D, E; A \rightarrow B, C \rightarrow D, D \rightarrow E)$$

  - el atributo A es clave candidata porque es superclave (ya demostrado) y

$$A \not\rightarrow ABCDE$$
$$C \not\rightarrow ABCDE$$

- Para disponer de métodos eficientes para el diseño de bases de datos relacionales, es necesario disponer de algoritmos adecuados relacionados con la manipulación de DF. Los principales servirán para:

  A. Determinar si una dependencia $X \rightarrow Y$ pertenece al cierre DF^+
  B. Encontrar un procedimiento eficiente no basado en el cierre de un conjunto de dependencias, para determinar la equivalencia entre dos conjuntos de dependencias.
  C. Hallar un **recubrimiento irredundante**, necesario para abordar el tema de la normalización, tanto en los algoritmos de síntesis como de análisis.
  D. Verificar si un descriptor es clave de un esquema de relación.
  E. Obtener todas las claves candidatas de un esquema relación.

- A fin de abordar los problemas anteriores, es necesario antes definir el concepto de cierre transitivo de un descriptor.
- Dado un esquema de relación R(A, DF), el **cierre transitivo de un descriptor** X de R respecto al conjunto de dependencias DF, denotado como

$$X^{+}_{DF}$$

es el subconjunto de los atributos de A tales que

$$X \rightarrow X^{+}_{DF} \in DF^{+}$$

siendo X^{+}_{DF} máximo en el sentido de que la adición de cualquier atributo vulneraría la condición anterior.

- Algoritmo de Ullman para calcular el cierre de un descriptor:

Entrada :

Un conjunto de dependencias DF y de atributos, R(A,DF)

Un descriptor X subconjunto de A

Salida :

X^+ , cierre de X respecto a DF.

Proceso:

1) $X^+ = X$

2) Repetir hasta que no se añadan más atributos a X^+:

Para cada dependencia $Y \rightarrow Z \in DF$

si $(Y \in X)$ y $\neg(Z \in X^+)$ entonces

$X^+ = X^+ \cup Z$

- Ejemplo de cálculo del cierre de un descriptor:

Dada la relación R ({CE, NE, P, G, CP, C}, DF)

con DF={CE $\rightarrow$ NE, NE $\rightarrow$ CE, P $\rightarrow$ CE, G $\rightarrow$ P, (CP, P) $\rightarrow$ G, CE $\rightarrow$ C, P $\rightarrow$ C}

hallar el cierre del descriptor (CP, P).

CP, P $\rightarrow$ CP, P *iteración 0*

CP, P $\rightarrow$ CP, P, CE, G, C *iteración 1*

CP, P $\rightarrow$ CP, P, CE, G, C, NE *iteración 2*

Luego el cierre transitivo del descriptor es:

$(CP, P)^{+}$ = CP, P, CE, G, C, NE

- Comprobar si una dependencia funcional $X \rightarrow Y$ se deriva de un conjunto de dependencias DF equivale a comprobar si

  $X \rightarrow Y$ pertenece a DF^+

- El algoritmo de comprobación es el siguiente:
  1. Calcular el cierre X^+_{DF} de X
  2. Si $Y \subseteq X^+_{DF}$ la dependencia $X \rightarrow Y \in DF^+$

     (o lo que es igual $DF \models X \rightarrow Y$),

     en caso contrario $X \rightarrow Y \notin DF^+$.

- Ejemplo:
  – Dado el conjunto de dependencias funcionales anterior (T27), comprobar si la dependencia $NE \rightarrow C$ se deriva de DF.
  1. Se calcula el cierre de NE: $NE^+ = NE, CE, C$
  2. Como C está en el cierre de NE, se cumple que

     $NE \rightarrow C$ pertenece a DF^+ y por tanto, se deriva de DF

- El problema de la equivalencia de dos conjuntos de DF es fundamental en el proceso de normalización, a fin de comprobar si la transformación de un esquema relacional se ha realizado conservando la semántica, al menos en lo que a dependencias se refiere.
- Dos conjuntos de dependencias DF_1 y DF_2 son equivalentes si sus cierres son iguales:

  $$DF^{+}_{1} = DF^{+}_{2}$$

- Para evitar el coste computacional del cálculo de los cierres, se puede comprobar si cada dependencia de DF_1 se encuentra en DF_2 y, viceversa, si cada dependencia de DF_2 se encuentra en DF_1.

- Algoritmo:

1. Si para toda dependencia $X \rightarrow Y$ de DF_2 se cumple

$$Y \subseteq X^{+}_{DF1}$$

significa que toda dependencia de DF_2 está en DF_1 y, por tanto, DF_1 es un recubrimiento de DF_2.

2. Recíprocamente, si para toda dependencia $Z \rightarrow W$ de DF_1, se cumple

$$W \subseteq Z^{+}_{DF2}$$

significa que toda dependencia de DF_1 está en DF_2 y, por tanto, DF_2 es un recubrimiento de DF_1.

3. Si se cumplen 1 y 2, DF_1 y DF_2 son mutuamente recubrimientos y, por tanto, son equivalentes.

- Ejemplo:
  - Dados los siguientes conjuntos de dependencias:

$$DF_1 = \{A \to B, B \to A, A \to C, A \to D\}$$

$$DF_2 = \{A \to B, B \to A, B \to C, B \to D\}$$

  - Las dependencias $A \to B$ *y* $B \to A$ están en ambos conjuntos, por lo que las únicas dependencias de DF_1 que no están en DF_2 son $A \to C$ y $A \to D$. Por tanto, debe calcularse el cierre de A con respecto al conjunto DF_2:

$$A^+_{DF2} = A, B, C, D$$

    - como C y D están contenidos en el cierre, queda demostrado que todas las dependencias de DF_1 están en DF^+_2, luego DF_2 es un recubrimiento de DF_1.
  - Análogamente, el cierre de B con respecto a DF_1 es:

$$B^+_{DF1} = B, A, C, D$$

    - y por tanto, las dependencias $B \to C$ y $B \to D$ de DF_2 están contenidas en DF^+_1, por lo que DF_1 es un recubrimiento de DF_2.
  - Como conclusión, DF_1 y DF_2 son equivalentes.

- Un conjunto de DF es **mínimo** cuando cumple:
  - Todas sus dependencias son elementales, y
  - No existe en el conjunto de dependencias ninguna redundante, es decir, que se pueda deducir del resto aplicando los axiomas de Armstrong.
- De todos los posibles conjuntos equivalentes a un conjunto dado de dependencias, hay algunos de ellos que son mínimos diciéndose que son recubrimientos **irredundantes** (también llamados minimales) del conjunto dado de dependencias.
- Puesto que las dependencias son restricciones semánticas, es de interés eliminar todas aquellas que sean redundantes.
  - Por esta razón, además de para reducir la complejidad algorítmica, los algoritmos de normalización y los de cálculo de claves candidatas parten siempre de recubrimientos irredundantes.

- **Atributo extraño**: Dada la dependencia $X \rightarrow Y \in DF$, un atributo $A \in X$ se dice que es un atributo extraño si la dependencia $(X - A) \rightarrow Y \in DF^+$.
- **Dependencia redundante**: Una dependencia funcional f de DF se dice que es redundante si puede derivarse de {DF – f} mediante la aplicación de los axiomas de Armstrong:

$$\{DF - f\} \vdash f$$

- Un conjunto M de dependencias es **recubrimiento irredundante** si:
  - Todas sus dependencias son elementales.
  - No hay atributos extraños, es decir,

    $\neg\exists(X \rightarrow A \in M) \wedge (Z \subset X) \mid M \in \{M - (X \rightarrow A) \cup (Z \rightarrow A)\}^+$

    y $\neg\exists(Z \subset X) \mid \{M - (X \rightarrow A) \cup (Z \rightarrow A)\} \in M^+$
  - No existen dependencias redundantes, es decir,

    $\neg\exists(X \rightarrow A \in M) \mid \{M - (X \rightarrow A)\} \equiv M$
- Dado un conjunto de dependencias DF siempre es posible encontrar un recubrimiento irredundante.
- Pueden existir varios recubrimientos irredundantes de un mismo conjunto de dependencias.

- La utilización de recubrimientos irredundantes tiene dos objetivos:
  1. Reducir la complejidad algorítmica al disminuir el número de dependencias de partida), y
  2. Minimizar el número de restricciones de integridad que han de ser mantenidas en la base de datos.
- Por ambas razones, debe ser un *objetivo de diseño conseguir que el número de dependencias y el número atributos involucrados sean mínimos*.

- Además, existe <u>otro objetivo de diseño</u>, que es aún más importante:
  - *que las dependencias resultantes tengan un significado claro para los usuarios*.
  - Este problema no puede ser resuelto con la teoría de la normalización, ya que realiza transformaciones algorítmicas de tipo sintáctico que pueden conducir a dependencias y a esquemas de relación **absurdos** desde el punto de vista del usuario.

- Algoritmo de Ullman y Atkins para calcular el recubrimiento irredundante:

Entrada: DF (conjunto de dependencias elementales)

Salida: H (recubrimiento minimal de DF)

Proceso :

1) Eliminación de atributos extraños.

1.1) Repetir para cada dependencia $X \rightarrow B$ de DF:

1.1.1) $L = X$

1.1.2) Repetir para cada atributo A de X:

Si $B \in (L - A)^+$ entonces $L = L - A$

1.1.3) Reemplazar $X \rightarrow B$ por $L \rightarrow B$

2) Eliminación de dependencias redundantes.

2.1) $H = F$

2.2) Repetir para cada dependencia $X \rightarrow A$ de DF:

$G = H - \{ X \rightarrow A \}$

Si A pertenece a X^+_G entonces $H = G$

- Dada una relación R(A,DF), para comprobar si un descriptor X es una superclave y/o clave candidata:
    1. Se calcula el cierre X^+_{DF} :

       Si $X^+_{DF} = A \Rightarrow$ X es una superclave

       en caso contrario $\Rightarrow$ X no es una superclave
    2. Si X es una superclave:

       Si $\exists(X' \mid (X' \subset X) \wedge (X'^+_{DF} = A)) \Rightarrow$ X no es una clave candidata

       en caso contrario $\Rightarrow$ X es una clave candidata

- Ejemplo: Dado el esquema de relación R(*AT, DF*)

  Con AT={*A,B,C,D,E,F*} *y* DF={$A \rightarrow B; B \rightarrow A; C \rightarrow E; E \rightarrow F; (A,C) \rightarrow D$}

  – Como $(A,C)^+_{DF} = (A,C,B,E,D,F) = AT \Rightarrow$ (*A,C*) es una superclave.

  – Además, como $A^+_{DF} = (A, B) \neq AT$ y $C^+_{DF} = (C, E, F) \neq AT$

    $\Rightarrow$ (*A,C*) es una clave candidata.

- Dado un esquema de relación R(A,DF), si se eliminan de DF todas aquellas dependencias que supongan la equivalencia de descriptores, dejando sólo uno de cada grupo de descriptores equivalentes.
- Para **calcular las claves candidatas de R** se ha de tener en cuenta lo siguiente:
  - Todo atributo independiente (que no interviene en ninguna dependencia funcional ni como implicante ni como implicado) forma parte de todas las claves.
  - Los descriptores equivalentes dan lugar a varias claves.
  - Ningún atributo implicado que no es implicante forma parte de ninguna clave.
  - Todo atributo implicante pero no implicado forma parte de todas las claves (siempre que no tenga otros equivalentes).
  - Aquellos atributos que son implicantes e implicados pueden formar parte de alguna clave.

- Dado un esquema de relación R(A,DF), siendo DF un recubrimiento irredundante, los pasos para calcular sus claves candidatas son:
  - ***Paso 1:*** *Eliminación de atributos independientes.*
    - Se eliminan de R todos los atributos independientes (que no forman parte de ninguna dependencia) obteniendo una relación R_{si}.
  - ***Paso 2:*** *Eliminación de descriptores equivalentes.*
    - Por cada grupo de descriptores equivalentes (X↔Y...), se elige uno (por ejemplo X), eliminando las dependencias de equivalencia anteriores de DF y sustituyendo en las dependencias restantes los descriptores eliminados (por ejemplo Y) por el atributo que se ha elegido del grupo (X en este caso).
    - Se obtiene así una relación R_{sie}.
    - Cuando, como resultado de este paso, las relaciones no tienen dependencias, los atributos de las mismas son independientes:
      - Ejemplo: R(A,B; ∅) implica que los atributos A y B son independientes.

- ***Paso 3:*** *Determinación de un descriptor (en el que no haya implicados) que sea clave de R_{sie}.*
  - Los atributos de una relación R_{sie} que son implicantes pero no implicados son parte de la clave, tomamos estos atributos y con ellos formamos una clave posible (K_p).
    - Si no hay ningún otro implicante que, a la vez, sea implicado, K_p es una clave y se va al paso 5.
    - En caso contrario, se realiza el paso 4.

- ***Paso 4:*** *Determinación de un descriptor clave de R_{sie} (en el que puede haber implicados siempre que sean también implicantes)..*
  - Si es posible, se obtiene una partición R'_{sie} eliminando de R_{sie} todos aquellos atributos que entran en K^+_p y que no forman parte de otras dependencias funcionales, distintas a las que han servido para calcular K^+_p.
    - En R'_{sie} se obtiene una clave provisional K'_p con los implicantes que estaban también en K_p añadiendo un nuevo implicante que, a su vez, sea también implicado. Si K'^+_p contiene todos los atributos de R'_{sie} es una clave, en caso contrario se añade un nuevo descriptor hasta obtener una clave.
    - Se repite esta operación porque puede haber más claves.
    - Una vez obtenidas las claves de R'_{sie} se hace la unión de cada una de ellas con la clave obtenida en el paso 3 para obtener así las claves de R_{sie}.
  - Si no fuese posible obtener la partición R'_{sie} se actuaría de la misma manera que se acaba de explicar, pero con R_{sie}.

– ***Paso 5:*** *Tratamiento de atributos independientes para obtener una clave de la relación original.*
  - A las claves de R_{sie} obtenidas en los paso 3 o 4 se añaden los atributos independientes obtenidos en el paso 1 (o en el 2).

– ***Paso 6:*** *Tratamiento de descriptores equivalentes.*
  - Cuando en el paso 2 se han obtenido descriptores equivalentes habrá que obtener todas las claves, sustituyendo en las claves obtenidas en el paso 5 (si hubiese atributos independientes) o en los pasos 3 o 4 (si no los hubiera), los descriptores por sus equivalentes.
  - De esta forma se obtienen todas las claves candidatas.