



BASES DE DATOS

Tema 7

Teoría de la Normalización

(parte 1)

UCLM- E.S. de Informática

Coral Calero, Marcela Genero, Francisco Ruiz



Objetivos

- Suministrar una sólida base teórica (**teoría de la normalización** basada en las dependencias entre datos) al diseño lógico de bases de datos.



Contenido

- Fundamentos de la teoría de la normalización
 - Tipos de dependencias entre datos
 - Dependencias funcionales (DF)
 - Definición formal y cálculo de claves
 - Algoritmos basados en DF



Bibliografía

- Básica
 - Piattini et al. (2006)
 - Cap. 9
- Complementaria
 - Atzeni et al. (1999)
 - Cap. 8
 - Elmasri y Navathe (2002)
 - Cap. 14-15



Tipos de Dependencias entre Datos

- **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, vamos a considerar 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.



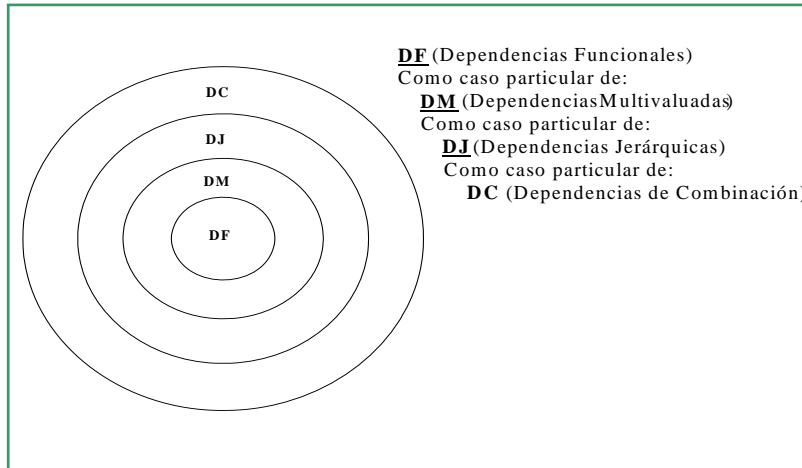
Tipos de Dependencias entre Datos

- 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 es el de las dependencias funcionales. Sobre este conjunto de dependencias, se apoyan las formas normales básicas.



Tipos de Dependencias entre Datos

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



UCLM ESI-BDa

7-Parte 1.7



Dependencias Funcionales

- **Definición:**
 - 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
 $X \rightarrow Y$
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 .
 - Lo contrario, decir X no determina funcionalmente a Y :
 $X \not\rightarrow Y$ o $X \nrightarrow Y$
- 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 dni de estudiante determina el nombre del mismo:
 $DNI_Estudiante \rightarrow Nombre$

UCLM ESI-BDa

7-Parte 1.8



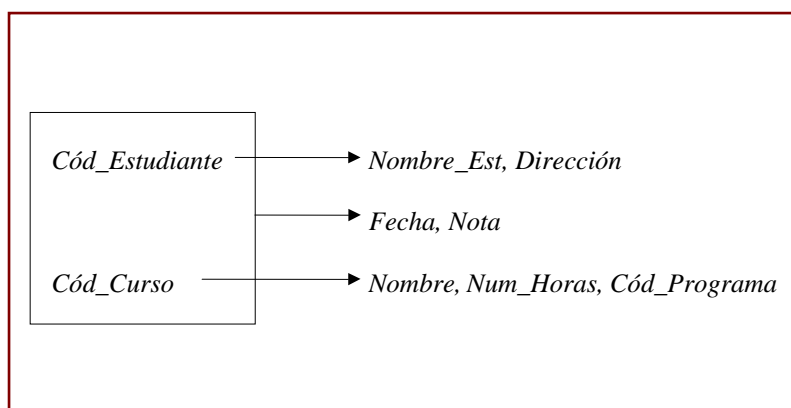
Dependencias Funcionales Descriptores Equivalentes

- Dos descriptores X e Y se dice que son **equivalentes** si
$$X \rightarrow Y \wedge Y \rightarrow X$$
 - también se puede representar como:
$$X \leftrightarrow Y$$
- 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:
$$\textit{Cód_Estudiante} \leftrightarrow \textit{DNI}$$



Dependencias Funcionales Diagrama de Dependencias

- Es un **grafo** que representa un conjunto de atributos y las DF existentes entre ellos. Es una herramienta muy útil a la hora de explicitar las DF.





Dependencias Funcionales

Dependencias Plenas

- Se dice que Y tiene **DF 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 $Cód_Curso, Cód_Edición, Cód_Estudiante \Rightarrow Nota$ refleja que la nota la obtiene un estudiante en una edición determinada de un curso determinado.
- Atributo extraño**: son los atributos del determinante de una DF que hacen que ésta no sea plena. También se llaman **ajenos**. Ejemplo según el diagrama anterior:
 - La DF $Cód_Estudiante, Cód_Curso \Rightarrow Cód_Programa$
 - Es no plena y *Cód_Estudiante* es un atributo extraño.



Dependencias Funcionales

Dependencias Triviales y Elementales

- Una DF $X \rightarrow Y$ es **trivial** si Y es un subconjunto de X:
 - $Y \subseteq X$.
 - Ejemplo: las siguientes DF son triviales:
$$Cód_Estudiante \rightarrow Cód_Estudiante$$
$$Cód_Curso, Cód_Edición \rightarrow Cód_Curso$$
- Decimos que una DF $X \rightarrow Y$ es **elemental** si
 - Y es un atributo único** no incluido en X, y
 - no existe X'** incluido en X tal que $X' \rightarrow Y$.
 - Es decir, 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.



Dependencias Funcionales

Dependencias Transitivas

- Dado el esquema de relación

$R(X, Y, Z)$

en el que se cumple que

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

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

- Se representa por

$X \text{ --- } \rightarrow Z$ (flecha discontinua)

- Notar que X e Y no tienen que ser equivalentes

- DF transitiva **estricta**:

- Es cuando además de las condiciones anteriores, también se cumple que

$Z \not\rightarrow Y$



Dependencias Funcionales

Dependencias Transitivas - ejemplos

- 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:

$Cód_Programa \not\rightarrow Cód_Curso$

y por tanto, se cumple la **DF transitiva**

$Cód_Curso \text{ --- } \rightarrow Cód_Departamento$

que también es **estricta** porque

$Cód_Departamento \not\rightarrow Cód_Programa$



Dependencias Funcionales

Consecuencia Lógica

- 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: $DF = \{ \text{Cód_Estudiante} \rightarrow \text{DNI_Est} ; \text{DNI_Est} \rightarrow \text{Cód_Estudiante} ; \text{Cód_Estudiante}, \text{Cód_Beca} \rightarrow \text{Fecha_Solicitud} \}$
se cumple que
 $DF \models \text{DNI_Est}, \text{Cód_Beca} \rightarrow \text{Fecha_Solicitud}$



Dependencias Funcionales

Cierre de un Conjunto de Dependencias

- 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 **reglas de derivación** 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.



Dependencias Funcionales

Reglas de Derivación

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

$$DF \mid - f$$

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, \dots, 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:

$$\forall f \mid DF \mid - f \text{ se_implica_que } DF \mid = f \text{ (propiedad de corrección)}$$

y

$$\forall f \mid DF \mid = f \text{ se_implica_que } DF \mid - f \text{ (propiedad de plenitud)}$$



Dependencias Funcionales

Axiomas de Armstrong

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 $X \rightarrow Z$

Derivados

- D1: Proyectividad**
Si $X \rightarrow Y$, entonces $X \rightarrow Y'$ si $Y' \subset Y$
- D2: Unión o aditividad**
Si $X \rightarrow Y$ y $X \rightarrow Z$, entonces $X \rightarrow YZ$
- D3: Pseudotransitividad**
Si $X \rightarrow Y$ e $YW \rightarrow Z$, entonces $XW \rightarrow Z$



Axiomas de Armstrong

- 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)



Axiomas de Armstrong

- Trabajar con los axiomas de Armstrong para optimizar (normalizar) esquemas de relación tiene varios **inconvenientes**:

- 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**.



Claves

Definición Formal de Superclaves

- 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 $SK \rightarrow A$ es una consecuencia lógica de DF , siendo, por tanto, un elemento de su cierre:

$$(SK \neq \emptyset) \wedge (SK \rightarrow A \in DF^+)$$

- Esta condición se conoce como propiedad de **unicidad**.
 - Significa que **una superclave determina a todos los atributos de la relación**.
- Ejemplo: para la relación ya vista
 $R(A, B, C, D, E; \{A \rightarrow B, C \rightarrow D, D \rightarrow E\})$
 - el descriptor AC es superclave porque
 $AC \rightarrow ABCDE$ (todos los atributos)

UCLM ESI-BDa

7-Parte 1.21



Claves

Definición Formal de Claves Candidatas

- 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 $K' \rightarrow A$.

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

- 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 antes
 $R(A, B, C, D, E; \{A \rightarrow B, C \rightarrow D, D \rightarrow E\})$
 - AC es clave candidata porque es superclave (ya demostrado) y
 $A \not\rightarrow ABCDE$
 $C \not\rightarrow ABCDE$

UCLM ESI-BDa

7-Parte 1.22



Algoritmos Basados en DF

- Para disponer de métodos eficientes para el diseño de BD relacionales, es necesario disponer de algoritmos adecuados relacionados con la manipulación de DF. Los principales sirven 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.



Algoritmos Basados en DF Cierre de un Descriptor

- 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.



Algoritmos Basados en DF Cierre de un Descriptor

- **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 → Z ∈ DF**
si (**Y ∈ X**) y **¬(Z ∈ X⁺)** entonces
X⁺ = X⁺ ∪ Z



Algoritmos Basados en DF Cierre de un Descriptor

- **Ejemplo** de cálculo del cierre de un descriptor:

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

con **DF={CE → NE, NE → CE, P → CE, G → P, (CP, P) → G,
CE → C, P → C}**

hallar el cierre del descriptor **(CP,P)**

CP,P → CP, P *iteración 0*

CP,P → CP, P, CE, G, C *iteración 1*

CP,P → CP, P, CE, G, C, NE *iteración 2*

Luego el cierre transitivo del descriptor es:

(CP, P)⁺ = CP, P, CE, G, C, NE



Comprobar Dependencias

- 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, 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



Equivalencia de Conjuntos de Dependencias

- 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 .



Equivalencia de Conjuntos de Dependencias

- Algoritmo:

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

$$Y \subseteq X^+_{DF_1}$$

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

$$W \subseteq Z^+_{DF_2}$$

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.



Equivalencia de Conjuntos de Dependencias

- Ejemplo:

- Dados los siguientes conjuntos de dependencias:

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

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

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

$$A^+_{DF_2} = A, B, C, D$$

- 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:

$$B^+_{DF_1} = B, A, C, D$$

- y por tanto, las dependencias $B \rightarrow C$ y $B \rightarrow D$ 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.



Algoritmos Basados en DF Recubrimiento Irredundante

- 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**.



Algoritmos Basados en DF Recubrimiento Irredundante

- **Atributo extraño**: Dada la dependencia $X \rightarrow Y \in DF$, un atributo $A \in X$ se dice que es un atributo extraño si $(X - A) \rightarrow Y \in DF^+$.
- **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:

$$\{DF - f\} \mid - f$$
- Un conjunto M 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.



Algoritmos Basados en DF

Recubrimiento Irredundante

- 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 **otro objetivo de diseño**, 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.



Algoritmos Basados en DF

Recubrimiento Irredundante

- **Algoritmo de Ullman y Atkins:**
 - 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) $H = DF$
 - 2.2) Repetir para cada dependencia $X \rightarrow A$ de DF:
 $G = H - \{ X \rightarrow A \}$
Si A pertenece a X^+_G entonces $H = G$



Algoritmos Basados en DF Identificación de Claves

- 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**.



Algoritmos Basados en DF Procedimiento de Cálculo de Claves

- 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.



Procedimiento de Cálculo de Claves

- 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 ($X \leftrightarrow Y \dots$), 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: $R(A,B; \emptyset)$ implica que los atributos A y B son independientes.



Procedimiento de Cálculo de Claves

- **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.



Procedimiento de Cálculo de Claves

- **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} .



Procedimiento de Cálculo de Claves

- **Paso 5:** *Tratamiento de atributos independientes para obtener una clave de la relación original.*
 - A las claves de R_{sie} obtenidas en los pasos 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.



Procedimiento de Cálculo de Claves - ejemplos

- **Ejemplo 1**

Sea el esquema de relación:

$R(\{A,B,C,D,E,F,G,H,I,J\}; \{AB \rightarrow C, C \rightarrow AB, E \rightarrow D, D \rightarrow E, E \rightarrow F, F \rightarrow E, ABD \rightarrow G, CF \rightarrow H\})$

- **Paso 1**

- Los atributos **I y J** son **independientes** porque no forman parte de ninguna DF, luego, en este primer paso se eliminan de la relación:

- $R_{si}(\{A,B,C,D,E,F,G,H\}, \{AB \leftrightarrow C, D \leftrightarrow E \leftrightarrow F, ABD \rightarrow G, CF \rightarrow H\})$

- **Paso 2**

- Existen **dos grupos de descriptores equivalentes**:

- a) **AB y C**
- b) **D, E y F**

- Del grupo a) nos quedaríamos, por ejemplo, con C y del grupo b) con D (eliminaríamos, por tanto, AB, E y F); la relación resultante sin equivalencias sería:

- $R_{sie}(\{C, D, G, H\}; \{CD \rightarrow G, CD \rightarrow H\})$



Procedimiento de Cálculo de Claves - ejemplos

- **Paso 3**

- En la R_{sie} anterior, **CD** es el único implicante, pero no implicado, luego una K_p sería CD, como el resto son sólo implicados, **CD es clave de R_{sie}** (no haría falta hallar el cierre de CD). Pasaríamos al paso 5.

- **Paso 5**

- Si a **CD** le añadimos los atributos independientes **I y J** tenemos **CDIJ** que es la clave de R.

- **Paso 6**

- Los descriptores equivalentes eran: **$AB \leftrightarrow C$** y **$D \leftrightarrow E \leftrightarrow F$**

- La clave **CDIJ** genera las siguientes claves candidatas de R:
 $\{C|AB\}\{D|E|F\}IJ$

- En total, son **6 claves**: **CDIJ, CEIJ, CFIJ, ABDIJ, ABEIJ y ABFIJ**



Procedimiento de Cálculo de Claves - ejemplos

- Ejemplo 2

Sea el esquema de relación:

$R(\{A,B,C,D,E,F\}; \{AB \rightarrow C, DE \rightarrow F, F \rightarrow D\})$

- Paso 1

- No hay atributos independientes => $R_{si} = R$

- Paso 2

- No hay descriptores equivalentes => $R_{sie}(\{A,B,C,D,E,F\}; \{AB \rightarrow C, DE \rightarrow F, F \rightarrow D\})$

- Paso 3

- $K_p = ABE$ y $K_p^+ = ABCE$; luego K_p no es clave, por lo que iríamos al paso 4.



Procedimiento de Cálculo de Claves - ejemplos

- Paso 4

- Obtenemos una nueva relación R'_{sie} eliminando de R_{sie} los atributos A B C que forman la primera DF (no eliminamos E porque en la dependencia de la que forma parte aparecen D y F que no están en R_{sie}) y nos queda:

$R'_{sie} (\{DEF\}; \{DE \rightarrow F, F \rightarrow D\})$

- Formaríamos una clave provisional K'_p con E que es sólo implicante (y, por tanto, está en R_{sie}), añadiendo un descriptor implicante e implicado, por ejemplo F:

- $K'_p = EF$ y $K'^+_p = EFD$; luego EF es una clave de la partición R'_{sie}

- Otra clave sería ED. Por tanto, las claves de R_{sie} serían:

ABEF
ABED

- Paso 5

- Como en el paso 1 no hubo atributos independientes, las claves son ABEF y ABED.

- Paso 6

- Como tampoco hubo descriptores equivalentes en el paso 2, las claves son ABEF y ABED.