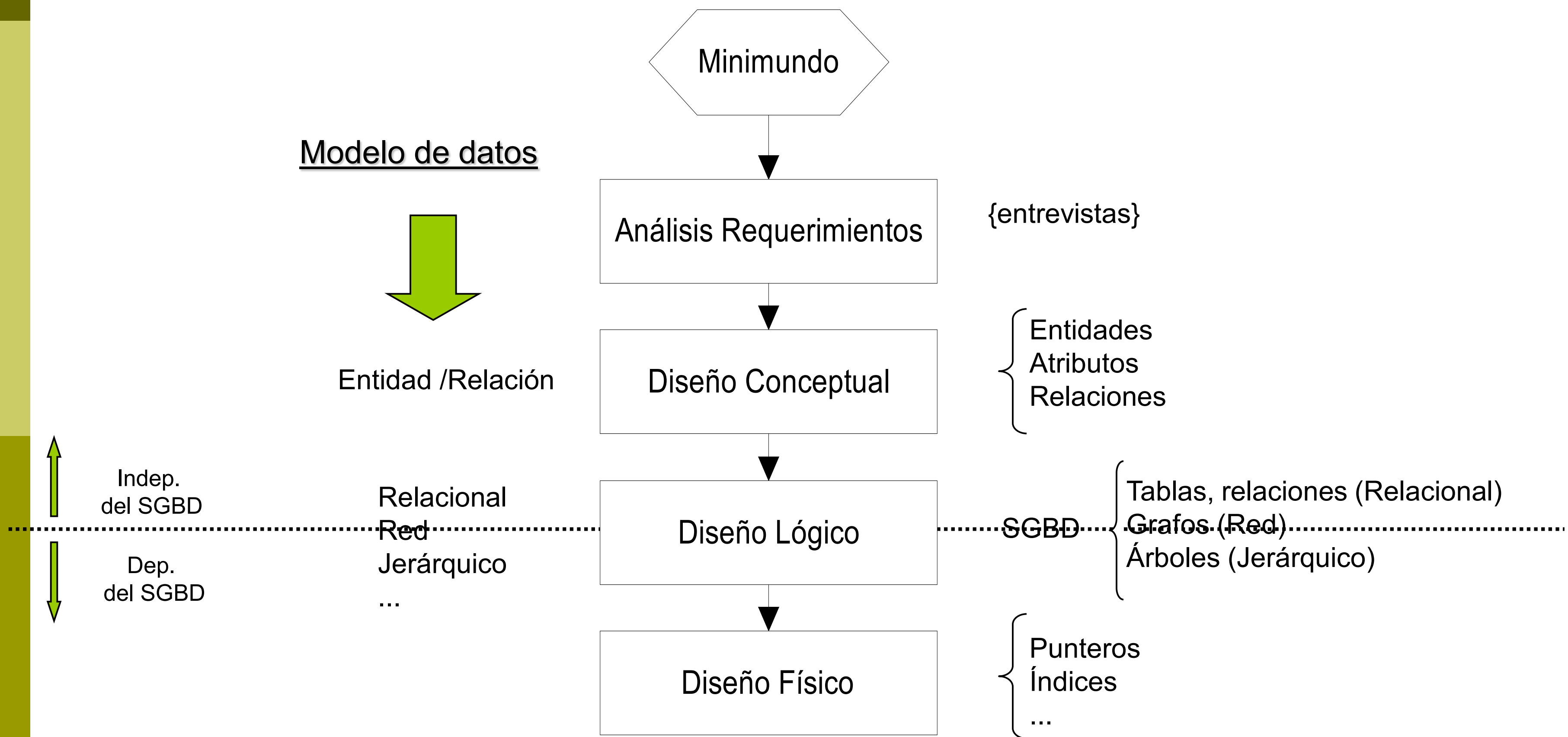
Bases de Datos

Tema 5: Teoría de diseño de Bases de Datos Relacionales (I)

5.1. Introducción

- Fases de diseño de una base de datos



5.1. Introducción

- Fases de diseño de una base de datos
  1. Mod. Conceptual (MERE) → Mod. Lógico (Relacional)
  2. Mod. Lógico (Relacional)
- En el primer caso se genera un esquema relacional estructurado y con poca redundancia, aunque siempre resulta conveniente aplicar un conjunto de reglas, conocidas como **teoría de la normalización**, que permiten asegurar que un esquema relacional cumple unas ciertas propiedades.
- En el segundo caso, la teoría de la normalización resulta imprescindible.

5.1. Introducción

- En grandes esquemas de relación, pueden aparecer los siguientes problemas cuando el diseño del esquema relacional no es adecuado:
  - **Incapacidad para almacenar ciertos hechos**
  - **Redundancias**, y por tanto, posibilidad de inconsistencias
  - **Inconsistencias**
  - Aparición en la base de datos, como consecuencia de las **redundancias**, de estados que no son válidos en el mundo real, es lo que se conoce como:
    - anomalías de inserción
    - anomalías de borrado
    - anomalías de modificación

5.1. Introducción

➔ *Ejemplo de diseño inadecuado*

ESCRIBE

<u>AUTOR</u>	NACIONALIDAD	<u>COD_LIBRO</u>	TITULO	EDITORIAL	AÑO
Date, C.	Norteamericana	23433	Databases	Adisson-W.	2008
Date, C.	Norteamericana	54654	SQL Standard	Adisson-W.	2006
Date, C.	Norteamericana	53235	Guide To Ingres	Adisson-W.	2008
Codd, E.	Norteamericana	97875	Relational M.	Adisson-W.	2013
Gardarin	Francesa	34245	Base de Datos	Paraninfo	2006
Gardarin	Francesa	86754	Comparación BD	Eyrolles	2014
Valduriez	Francesa	86754	Comparación BD	Eyrolles	2014
Kim, W.	Norteamericana	23456	OO Databases	ACM Press	2009
Lochovsky	Canadiene	23456	OO Databases	ACM Press	2009

- Redundancia:
  - la nacionalidad se repite por cada libro
  - la editorial y el año se repite con libros que tienen más de un autor

5.1. Introducción

→ *Ejemplo de diseño inadecuado*

- La redundancia provoca a su vez:
  - **<u>Anomalías de inserción</u>**: al dar de alta un libro es preciso insertar tantas tuplas como autores tenga el libro.
  - **<u>Anomalías de modificación</u>**: al cambiar la editorial un libro es preciso modificar todas la tuplas que corresponden a ese libro.
  - **<u>Anomalías de borrado</u>**: el borrado de un libro obliga a borrar varias tuplas, tantas como autores tenga ese libro y, viceversa, el borrado de un autor lleva a borrar tantas tuplas como libros ha escrito ese autor.
- Otros problemas:
  - **Imposibilidad de almacenar ciertos hechos**:
    - No es posible almacenar información sobre autores sobre los que no existe ningún libro en la base de datos, ya que ***cod_libro*** forma parte de la clave primaria.
    - No es posible introducir obras anónimas (sin autor).
  - **Desaparición de información**:
    - Al dar de baja un libro se pierde toda la información sobre los autores si estos sólo tienen ese libro en la base de datos.
    - Al dar baja un autor se borran todos los libros escritos por ese autor (salvo que hubiese sido escrito por más de un autor).

5.1. Introducción

➔ *Ejemplo de diseño inadecuado*

- La relación anterior debería haberse diseñado con el siguiente esquema relacional:
  - **LIBRO** (<u>COD_LIBRO</u>, TITULO, EDITORIAL, AÑO)
    AUTOR (<u>NOMBRE</u>, NACIONALIDAD)
    ESCRIBE (<u>COD_LIBRO</u>, <u>NOMBRE</u>)

- En general, realizando un buen diseño conceptual en el modelo E/R, seguido de una cuidadosa transformación al modelo relacional, se evitarían gran parte de las anomalías citadas anteriormente.

- Existen **cuatro medidas informales** de calidad para el diseño de esquemas de relación:
  - **Semántica de los atributos**: relación semántica entre los atributos de una tabla.
  - **Reducción de información redundante** en las tuplas: Esto evitaría las *anomalías* ya vistas.
  - **Reducción de valores *nulos***.
  - **Eliminación de la posibilidad de generación de tuplas *espurias***.

5.2. Dependencias funcionales (DF's) → *Esquema de Relación*

- Toda relación puede definirse de dos formas:
  - Por extensión: especificando todas y cada una de las tuplas que componen la relación.
  - Por intensión: especificando el esquema de la relación.
- **Esquema de relación**: estructura abstracta que define una relación a través de un nombre, un conjunto de atributos y un conjunto de restricciones que caracterizan a esa relación.
  - Esquema de relación: r= R(T, L) donde:
    - R es el nombre de la relación r.
      - Ej: R=COCHE
    - T es el conjunto de atributos que definen a R.
      - Ej: T={matricula, color, potencia, marca, modelo}
    - L es el conjunto de restricciones que caracterizan a R. Las restricciones pertenecientes al conjunto L, son las restricciones que permiten definir la semántica del problema, se conocen como **dependencias funcionales** y se han de cumplir para cualquier extensión de la relación.
      - $L = \{L_1, L_2, \ldots, L_n\}$.
  - **Descriptor**: Subconjunto de T.

5.2. Dependencias funcionales (DF's) → *Definición*

- **DF**: Es una **restricción** entre dos conjuntos de atributos (descriptores) de una relación.
- Dada una relación $R(A_1, A_2, \ldots, A_n)$ y siendo X e Y descriptores de la relación R, se dice que Y es funcionalmente dependiente de X, y se denota por X→Y, si cada valor de X determina unívocamente el valor de Y.
  - Es decir, si se cumple:
    - si $\pi_X(t1)=\pi_X(t2) \Rightarrow \pi_Y(t1)=\pi_Y(t2)$ $\forall$ EXTENSIÓN de R, $\forall$ t1, t2 tuplas de R
  - Es decir:
    - si dos tuplas son iguales para los atributos X, también lo son para los Y
    - a cada valor de X le corresponde un ÚNICO valor de Y
  - Ejemplo de DF's en el esquema EMPLEADO(T, *L*) definido como sigue:
    - T={DNI, nombre, dirección, puesto, antigüedad, salario}
      L = {DNI→nombre, dirección, puesto, antigüedad, salario;
      puesto, antigüedad→salario}

5.2. Dependencias funcionales (DF's) → *Ejemplo*

- Para la relación coche:

coche	matrícula	marca	modelo	potencia	color
	OU1234M	SEAT	ALTEA1.9TDI	90	AZUL
	PO1234M	SEAT	ALTEA1.9TDI	90	ROJO
	BNL1243	AUDI	A3	105	NEGRO

  - Se podría extraer el siguiente conjunto de DF's
    L ={matrícula→color,modelo→potencia,modelo→marca,matrícula→modelo}
  - ¿Existen otras DF's?
    - ¿modelo→color?
    - ¿potencia→matrícula?
    - ¿marca →modelo?

5.2. Dependencias funcionales (DF's) → *Ejemplo*

- Dada la relación **Persona(dni, calle, mun, prov, cp)** que almacena el dni, calle, municipio (mun), provincia (prov) y código postal (cp) donde vive una persona. Se cumplen las siguientes restricciones semánticas:

  a) dni es el identificador de una persona, ya que una persona sólo vive en una calle, un municipio, una provincia y un cp:

  dni→calle, mun, prov, cp

  b) Un municipio pertenece a una provincia y no existen dos municipios con el mismo nombre:

  mun→prov

  d) No existen códigos postales que incluyan más de un municipio, pero un municipio puede incluir más de un CP:

  cp→mun

  e) Una calle de un municipio de una provincia pertenece a un único cp:

  calle, mun, prov→cp

- Al conjunto resultante de DF's se denomina **L:{ }**

5.2. Dependencias funcionales (DF's) → *Consideraciones*

- Una **DF**:
  - es inherente al contenido **semántico** de los datos.
  - no puede deducirse de la EXTENSIÓN, es **CONCEPTUAL**.
  - El hecho de que X→Y no indica necesariamente que Y→X.
    - En caso de que X→Y ^ Y→X, se dice que X e Y son *descriptores equivalentes*.
  - especifica restricciones sobre los atributos de una relación que deben cumplirse **en todo momento** (son innvariantes en el tiempo). Da lugar a algún tipo de restricción semántica (*clave primaria*, *clave foránea*, *aserciones*, etc) que se deberá cumplir para cualquier extensión de la relación. Por ejemplo:
    - Si X es una clave candidata de R $\Rightarrow$ X→Y $\forall$Y de R

5.3. DF's parciales, totales, triviales, elementales

- Una DF, $X \rightarrow Y$, se dice que es una DF **parcial** cuando $\exists X' \subset X \mid X' \rightarrow Y$:
  - Ejemplo:
    - $\left.\begin{array}{l} AB \rightarrow C \\ B \rightarrow C \end{array}\right\}$ AB→C es parcial, ya que hay un subconjunto de AB del cual depende C
- Una DF, $X \Rightarrow Y$, se dice que es una DF **total** cuando $\neg\exists X' \subset X \mid X' \rightarrow Y$, es decir:
  - $X \rightarrow Y$
  - Y no depende funcionalmente de ningún subconjunto de X.
- Una DF, $X \rightarrow Y$, se dice que es una DF **trivial** cuando $Y \subseteq X$:
- Una DF, $X \rightarrow A$, se dice que es una DF **elemental** cuando:
  - A es un único atributo $\not\subset X$
  - $\neg\exists X' \subset X \mid X' \rightarrow A$

Es decir, es una DF:
- **TOTAL**
- **NO TRIVIAL**
- el **consecuente** es un único atb

5.4. DF's - Cierre Transitivo

→ L^+

- El conocer ciertas DF's puede llegar a inferir la existencia de nuevas DF's.
  - P. ej: **COCHE(T, L)**
    - Donde:
      - **T** = {matricula, modelo, marca, potencia, color}
      - **L** ={matricula→modelo; modelo→marca; modelo→potencia; matricula→color}
    - Sabiendo que *matricula→modelo* y que *modelo→marca* ¿se podría inferir alguna otra DF que no está presente en el conjunto inicial L?
      - **matricula → marca**
    - ¿Se puede deducir alguna otra DF?

      matricula→potencia
  - Al conjunto de DF's que son **consecuencia lógica** del conjunto inicial L de DF's se le denomina **cierre transitivo de L** y se representa por L^+
    - $L^+ = L \cup$ {matricula→marca, matricula→potencia}

5.4. DF's - Cierre Transitivo → *L*⁺

- Intuitivamente, dado un conjunto de DF's, hay otras que se deducen como consecuencia de ellas.
- ***L***$^+$(cierre de ***L***): conjunto inicial de DF's completado con las que se deducen a partir de él.
  - Se deducen aplicando los Axiomas de Armstrong:

Axiomas de **ARMSTRONG**

- **Reflexividad**: X → Y, siendo Y $\subseteq$ X
- **Aumento**: X → Y $\Rightarrow$ ZX → ZY
- **Transitividad**: X→Y, Y →Z $\Rightarrow$ X→Z
- **Unión**: X→Y, X→Z $\Rightarrow$X → YZ
- **Descomposición**: X →YZ $\Rightarrow$ X →Y y X→Z
- **Pseudo-Transitividad**: X →Y, WY→Z $\Rightarrow$ WX→Z

BÁSICOS

DERIVADOS

- Es decir:
  - ***L***$^+$ = ***L*** $\cup$ {DF's inferidas mediante los axiomas de Armstrong}
    - Lógicamente ***L*** $\subseteq$***L***$^+$

5.4. DF's - Cierre Transitivo → L^+

- Ejemplos:
  - **PERSONA(T, L)**
    - **Donde:**
      - **T=** {nss,nombre_emp, fecha_nac, direccion, numero_dpto, nss_jefe_dpto}
      - **L** ={**nss**→nombre_emp,fecha_nac, direccion, numero_dpto;
        numero_dpto→nombre_dpto, nss_jefe_dpto
        }
    - **¿Qué otras DF's se pueden inferir?**
      - **L**$^+$ = **L** ∪ {nss→nombre_dpto, nss_jefe_dpto; ← **Transitividad y Descompos.**
        nss→nss; ← **Reflexividad**
        numero_dpto→nombre_dpto;
        numero_dpto→nss_jefe_dpto; ← **Descomposición**
        ...}

5.4. DF's - Cierre Transitivo $\rightarrow L^+$

- ***Ejercicio***
  - Dado el esquema de relación:
    - R(A, B, C, D, E; {A → B, C → D, D → E})
  - Demostrar, aplicando los axiomas de Armstrong que: **AC→ABCDE**

1. A→B (dada)
2. AC→ABC (Aumento y Reflexividad)
3. C→D (Dada)
4. D→E (Dada)
5. C→E (Transitividad)
6. C→DE (Unión 3 y 5)
7. ABC→ABCDE (Aumento 6 por ABC)
8. AC→ABCDE (Transitividad 2 y 7)

5.5. Superclave y Clave candidata

- **Superclave**: se dice que un descriptor SK, subconjunto propio del conjunto de atributos del esquema ($SK \subseteq T$), es superclave del esquema R(T,L) cuando se cumple $SK \rightarrow T$, es decir, $(SK \rightarrow T) \in L^+$. *Restricción de Unicidad* (NO existen dos tuplas con la misma combinación de valores para SK)

- ***Ejemplo***: Dado el esquema:

  R(**T, L**), donde:

  T= {A, B, C, D, E}

  L= {$A \rightarrow B$, $C \rightarrow D$, $D \rightarrow E$})

  - ¿AC es ***Superclave***?

    es decir ¿$AC \rightarrow ABCDE \in L^+$?

5.5. Superclave y Clave candidata

- **Clave candidata**: se dice que un descriptor K, subconjunto propio del conjunto de atributos del esquema ($K \subseteq T$), es **clave** del esquema R(T,L) si, además de ser superclave, no existe ningún subconjunto estricto de K con la misma propiedad, es decir, $(K \Rightarrow T) \in L^+$. *Restricción de Irreductibilidad.*

- ***Ejemplo***: Dado el esquema:

  R(T, L)

  T= {A, B, C, D, E}

  L= {$A \rightarrow B$, $C \rightarrow D$, $D \rightarrow E$})

  - ¿AC es ***Clave Candidata***?

    Sabemos que AC es superclave, es decir $AC \rightarrow T \in L^+$, entonces, *¿existe algún subconjunto de AC con la misma propiedad*?

- **Atributos Principales**: aquéllos que forman parte de la clave
- **Atributos No Principales**: aquéllos que NO forman parte de la clave

5.6. Cierre de un descriptor respecto de **L**

- Operación fundamental para calcular las claves de un esquema.

- Una forma sistemática de inferirlas es determinar primero todos los conjuntos de atributos X que aparezcan como **miembro izquierdo de alguna DF** para después determinar **el conjunto de todos los atributos que son dependientes de X**.

- *Ejemplo*:
  - L:

$X \rightarrow Y$
$X \rightarrow Z$
$Y \rightarrow B$

¿Qué conjunto de atributos puedo identificar a partir de X?

X^+ es el conj. de valores identificados por X

5.6. Cierre de un descriptor respecto de **L** → *Algoritmo*

- **Algoritmo**:
  - **Entrada**:
    - Un conj. de DF's y atributos R(T, L)
    - Un descriptor X, subconjunto de T
  - **Salida**:
    - X^+, cierre de X respecto de L
  - **Proceso**:
    1. $X^+ = X$
    2. Repetir hasta que X^+ no cambie más
       - Para cada DF $(Y \rightarrow Z) \in \mathbf{L}$ tal que $Y \subseteq X^+$

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

- El proceso es **finito**, ya que T también lo es, y X^+ es **máximo**, ya que van a estar todos los atributos que dependen de X.

5.6. Cierre de un descriptor respecto de **L** → *Algoritmo*

- Ejemplo: sea el esquema de relación R(T, **L**),donde:

  T={A, B, C, D, E, F}

  L={AB $\rightarrow$ C, BC $\rightarrow$ AD, D $\rightarrow$ E, CF $\rightarrow$ B}

- Calcular el cierre del descriptor *AB* respecto de **L**:

  X^+= AB
  X^+= ABC (ya que AB$\subset X^+$ y AB→C)
  X^+= ABCD (ya que BC$\subset X^+$ y BC→AD)
  X^+= ABCDE (ya que D$\subset X^+$ y D→E)

5.6. Cierre de un descriptor respecto de **L** → *Algoritmo*

- Ejemplo: sea el esquema de relación R(T, **L**),donde:

  T={A, B, C, D, E, F}

  L : AB→C D→EF

  C→A BE→C

  BC→D CF→BD

  ACD→B CE→AF

- Calcular el cierre del descriptor *BD* respecto de **L**:

  X^+= BD
  X^+= BDEF (ya que D⊂X^+ y D→EF)
  X^+= BDEFC (ya que BE⊂X^+ y BE→C)
  X^+= BDEFCA (ya que C⊂X^+ y C→A)

  Como X^+ es igual al conj. de todos los atb's, el proceso termina.

VII. Recubrimientos no redundantes (r.n.r) ➔ *Equivalencia de conj. de DF's*

- ***Ejemplo***: Dados los siguientes conjuntos de DF's comprobar **si son equivalentes**:

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

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

  - Para que sean equivalentes, **M** debe ser un recubrimiento de **L** y **L** un recubrimiento de **M**.
  - Se realizaría del siguiente modo:
    - Las dependencias $A \rightarrow B$ y $B \rightarrow A$ están en ambos conjuntos, por lo que las únicas dependencias de **L** que no están en **M** son $A \rightarrow C$ y $A \rightarrow D$. Por tanto, debe calcularse el cierre de A (A^+) respecto al conjunto **M**:

      A^+ respecto de **M** = ABCD

      como C y D están contenidos en el cierre de A (A^+), queda demostrado que todas las dependencias de **L** están en **M**$^+$, luego **M** es un recubrimiento de **L**.

    - Análogamente, el cierre de B con respecto a **L** es:

      B^+ respecto de **L** = *BACD*

      y por tanto, las dependencias $B \rightarrow C$ y $B \rightarrow D$ de **M** están contenidas en **L**$^+$, por lo que **L** es un recubrimiento de **M**.

  - Como conclusión, **L** y **M** **<u>son dos conj. de DF's equivalentes</u>**. Porque **L**$^+$ = **M**$^+$

VII. Recubrimientos no redundantes (r.n.r)

- Un recubrimiento no redundante (r.n.r) de un conj. de DF's ***L*** es el conjunto **mínimo** de DF's que **es equivalente** a ***L***. Conjunto mínimo significa que:
  - Todas las DF's son **elementales**
  - **NO existen DF's redundantes**
- Otras acepciones: *cobertura mínima*, *recubrimiento minimal, recubrimiento canónico*.
- Dado un conj. de DF's siempre será posible hallar, por lo menos, un r.n.r.

VII. Recubrimientos no redundantes (r.n.r) → *Propiedades*

- Un r.n.r ha de cumplir las propiedades:

  1. **Todas sus DF's tienen un solo atributo en el consecuente**, es decir, de la forma $X \rightarrow A_i$.

  2. **No hay atributos extraños**. Un atributo $B_i \in X$ es extraño en la DF $X \rightarrow Ai$ de $\mathbf{L}$ cuando llamando Z al descriptor $X-\{Bi\}$, el cierre de $\mathbf{L}$ no se altera al sustituir $X \rightarrow Ai$ por $Z \rightarrow Ai$, es decir:
     $(\mathbf{L} - \{X \rightarrow Ai\} \cup \{Z \rightarrow Ai\})^{+} = \mathbf{L}^{+}$.

  3. **No hay DF's redundantes**. Un DF es redundante cuando su supresión no altera el cierre:
     $(\mathbf{L} - \{X \rightarrow Ai\})^{+} = \mathbf{L}^{+}$
     Es decir, no se puede quitar ninguna DF de L y seguir teniendo un conj. de DF equivalente a L.

VII. Recubrimientos no redundantes (r.n.r) → *Algoritmo*

- **Algoritmo**:
  - **Entrada**
    - **L** (conj. de DF's)
  - **Salida**:
    - **M** (r.n.r de L)
  - **Proceso**

1º paso. <u>Segundos miembros simples</u>: Para toda dependencia $X \rightarrow Y$ de ***L*** se sustituye por $X \rightarrow A_1$, $X \rightarrow A_2$, ..., $X \rightarrow A_k$, siendo $A_1, A_2, ..., A_k$ atributos de Y. Al conjunto resultante se le llama ***L****(1)*.

2º paso. <u>Eliminación de atributos extraños</u>: Para toda dependencia $X \rightarrow A_i$ de ***L****(1)*, si $B_i \in X$ y siendo $Z=X-\{B_i\}$ se calcula Z^+ respecto de ***L****(1)*.

- Si $A_i \in Z^+$, esto quiere decir que $(Z \rightarrow A_i) \in L^{(1)+}$, de modo que la sustitución de $X \rightarrow A_i$ por $Z \rightarrow A_i$ conduce a un conjunto equivalente.
- Al conjunto resultante se le llama ***L****(2)*

3º paso. <u>Eliminación de DF's redundantes</u>: Para toda dependencia $X \rightarrow A_i$ de ***L****(2)* se determina X^+ respecto de $L^{(2)}-\{X \rightarrow Ai\}$.

- Si $A_i \in X^+ \Rightarrow X \rightarrow A_i$ es redundante en ***L****(2)* y se elimina.
- Al conjunto resultante se le llama ***L****(3)* **= *M***

VII. Recubrimientos no redundantes (r.n.r) → *Ejercicio 1*

- *Ejemplo*: Calcular el r.n.r. del esquema de relación R(T,**L**) donde:

  T= {A, B, C, D, E} y L={ABCD→E, E→D, A→B, AC→D}

PASO 1: Convertir las DFs en elementales (segundos miembros simples)

$L^{(1)} = L$

PASO 2: Eliminación de atributos extraños

ABCD→E

$(ABC)^+ =$ ABCDE — $(AB)^+ =$ AB; $(AC)^+ =$ ACDBE; $(BC)^+ =$ BC

$(AC)^+ =$ ACDBE — $(A)^+ =$ AB; $(C)^+ =$ C ⇒ **AC→E**

$(ABD)^+ =$ ABD

$(ACD)^+ =$ ACDBE — $(AC)^+ =$ ACDBE ⇒ **AC→E**; $(AD)^+ =$ ADB; $(CD)^+ =$ CD

$(BCD)^+ =$ BCD

AC→D

En ABCD→E hay dos atributos extraños: D y B

En AC→D no hay atributos extraños

VII. Recubrimientos no redundantes (r.n.r) → *Ejercicio 1*

$L^{(2)}$= {AC→E, E→D, A→B, AC→D}

PASO 3: Comprobar si alguna DF es redundante

$(AC)^+$ respecto a L – {AC→E}= ACBD. Como E $\not\subset$ ABCD, AC→E, NO es redundante.

$(E)^+$ respecto a L – {E→D}= E. Como D $\not\subset$ E, E→D, NO es redundante.

$(A)^+$ respecto a L – {A→B}= A. Como B $\not\subset$ A, A→B, NO es redundante.

$(AC)^+$ respecto a L – {AC→D}=ACED. Como D⊂ACED, AC→D, ES redundante.

$L^{(3)}$= **M**= {AC→E, E→D, A→B}

VII. Recubrimientos no redundantes (r.n.r) → *Ejercicio 2*

EJEMPLO: Calcular el r.n.r. del esquema de relación R(T,***L***) donde:

T= {A, B, C, D, E, G}.

L = {AB→C, C→A, BC→D, ACD→B, D→EG, BE→C, CG→BD, CE→AG}

PASO 1: Convertir las DFs en elementales (segundos miembros simples)

$\boldsymbol{L}^{(1)}$= {AB→C, C→A, BC→D, ACD→B, D→E, D→G, BE→C, CG→B, CG→D, CE→A, CE→G}

PASO 2: Eliminación de atributos extraños

AB→C

$(A)^+$ respecto a $L^{(1)}$= A

$(B)^+$ respecto a $L^{(1)}$= B

BC→D

$(C)^+$ respecto a $L^{(1)}$= CA

ACD→B

$(AC)^+$ respecto a $L^{(1)}$= AC

$(AD)^+$ respecto a $L^{(1)}$= ADEG

$(CD)^+$ respecto a $L^{(1)}$= CDA**B** ⟹ CD→B

$(D)^+$ respecto a $L^{(1)}$= DEG

CD→B

BE→C

$(E)^+$ respecto a $L^{(1)}$= E

CG→B; CG→D

$(G)^+$ respecto a $L^{(1)}$= G

CE→A ⟹ **C→A**

CE→G

$\boldsymbol{L}^{(2)}$**= {AB→C, C→A, BC→D, CD→B, D→E, D→G, BE→C, CG→B, CG→D, CE→G}**

VII. Recubrimientos no redundantes (r.n.r) → *Ejercicio 2*

$L^{(2)}$= {AB→C, C→A, BC→D, CD→B, D→E, D→G, BE→C, CG→B, CG→D, CE→G}

PASO 2: Comprobar si alguna DF es redundante

$(AB)^+$ respecto a $L^{(2)}$ – {AB→C}= AB. Como C $\not\subset$ AB, AB→C, NO es redundante.

$(C)^+$ respecto a $L^{(2)}$ – {C→A}= C. Como A $\not\subset$ C, C→A, NO es redundante.

$(BC)^+$ respecto a $L^{(2)}$ – {BC→D}= BCA. Como D $\not\subset$ BC, BC→D, NO es redundante.

$(CD)^+$ respecto a $L^{(2)}$ – {CD→B}=CDAEG**B**. Como B $\subset$ ACDEGB, CD→B, ES redundante.

$L^{(2)}$ = {AB→C, C→A, BC→D, D→E, D→G, BE→C, CG→B, CG→D, CE→A, CE→G}

VII. Recubrimientos no redundantes (r.n.r) → *Ejercicio 2*

$L^{(2)}$ = {AB→C, C→A, BC→D, D→E, D→G, BE→C, CG→B, CG→D, CE→A, CE→G}

$(D)^+$ respecto a $L^{(2)}$ – {D→E}=DG. Como E $\not\subset$ DG, D→E, NO es redundante.

$(D)^+$ respecto a $L^{(2)}$ – {D→G}=DE. Como G $\not\subset$ DE, D→G, NO es redundante.

$(BE)^+$ respecto a $L^{(2)}$ – {BE→C}=BE. Como C $\not\subset$ BE, BE→C, NO es redundante.

$(CG)^+$ respecto a $L^{(2)}$ – {CG→B}=CGADE. Como B $\not\subset$ CGAD, CG→B, NO es redundante.

$(CG)^+$ respecto a $L^{(2)}$ – {CG→D}=CGAB**D**. Como D $\subset$ CGABD, CG→D, ES redundante.

$L^{(2)}$= {AB→C, C→A, BC→D, D→E, D→G, BE→C, CG→B, CE→G}

$(CE)^+$ respecto a $L^{(2)}$ – {CE→G}=CEA. Como G $\not\subset$ CEA, CE→G, NO es redundante.

$L^{(3)}$= M= {AB→C, C→A, BC→D, D→E, D→G, BE→C, CG→B, CE→G}

VII. Recubrimientos no redundantes (r.n.r) → *Algoritmo*

- *Ejercicio*: Calcular el r.n.r. del esquema de relación R(T,**L**) donde:

  T= {A, B, C, D, E, F}

  L :

  A→BCE

  AB→C

  EB→D

  C→B

  AD→C

  CD→EA

  AC→E

  AD→E

VIII. Algoritmo de simplificación-reducción

- **Problema**: los algoritmos de cálculo de claves son muy complejos desde el punto de vista computacional. Se proponen dos nuevos algoritmos:
  - **Simplificación-reducción**: su objetivo es reducir el conjunto de dependencias funcionales dadas.
  - **Síntesis**: su objetivo es sintetizar las claves a través de un conjunto reducido de atributos.

VIII. Algoritmo de simplificación-reducción → *Definiciones previas*

- Supóngase que ***M*** es el **r.n.r.** que constituye la entrada para el proceso de determinación de clave. Se define:

  - **Atributo Esencial**: se encuentra sólo en el lado izquierdo de cualquier DF de ***M***, o bien no existe en ninguna de ellas.

  - **Atributo Posible**: se encuentra en ambos lados de las dependencias de ***M***.

  - **Atributo No Esencial**: se encuentra sólo en el lado derecho de alguna DF de ***M***.

VIII. Algoritmo de simplificación-reducción → *Definiciones previas*

- Teniendo en cuenta las definiciones anteriores, se establecen las siguientes propiedades:
  - Un atributo **esencial** <u>pertenece</u> a todas las claves.
  - Un atributo **no esencial** <u>no pertenece</u> a ninguna clave.
  - Un atributo **posible** <u>no se puede decir</u> en principio, si pertenece o no a alguna clave.
- Estas propiedades permiten simplificar el proceso de determinación de claves, extrayendo **iterativamente** de ***M*** cualquier ocurrencia de atributos esenciales o no esenciales, y así reducir el tamaño del conjunto de DF's.

VIII. Algoritmo de simplificación-reducción → *Procesos Complementarios*

- Simplificación-reducción: se considera que ***M***, el conjunto de DF's dado, es un **r.n.r.**
  - El objetivo del algoritmo de simplificación-reducción es **obtener un conjunto de DF's minimizado, preservando la completitud y correctitud de las claves calculadas**.

VIII. Algoritmo de simplificación-reducción → *Pasos del algoritmo*

Sea el esquema relacional R = <T, M>, con $M = \{ X_i \rightarrow A_i \}_{i = 1, m}$ r.n.r.

Todas las dependencias funcionales deben ser <u>elementales</u>

- **Paso 1:**

  Calcular i, T_0, M_0 como:

  $i = 0$

  $T_0 = T$

  $M_0 = M$

- **Paso 2:**

  Definir T_i, M_i, E_i, N_i y P_i como:

  T_i conjunto de atributos que quedan

  M_i conjunto de dependencias que quedan

  E_i conjunto de atributos esenciales relacionados a T_i y M_i.

  N_i conjunto de atributos no esenciales relacionados a T_i y M_i.

  P_i conjunto de atributos posibles relacionados a T_i y M_i.

  Calcular E_i, N_i y P_i como:

  $E_i = T_i - \{ \cup A_k \}$

  $N_i = \{ \cup A_k \} - \{ \cup X_k \}$

  $P_i = T_i - (E_i \cup N_i)$

VIII. Algoritmo de simplificación-reducción → *Pasos del algoritmo*

Sea el esquema relacional R = <T, M>, con $M = \{ X_i \rightarrow A_i \}_{i = 1, m}$ r.n.r.

- **Paso 3:**

  Si ($M_i = \emptyset$) o ($E_i = N_i = \emptyset$)

  Entonces Fin del Algoritmo

  Sino Avanzar al próximo paso

- **Paso 4:**
  - Calcular $\mathbf{M_{i+1}}$ a partir de $\mathbf{M_i}$ haciendo la siguiente simplificación:

    a.- Los atributos que pertenecen al conjunto $(E_i)^+$ se eliminan de su respectiva dependencia funcional, es decir, $\forall B \in (E_i)^+$, substituir $(X_k \rightarrow A_k)$ por $((X_k - B) \rightarrow A_k)$

    b.- Si $\exists$ k tal que $X_k \subseteq (E_i)^+$, entonces eliminar la dependencia $(X_k \rightarrow A_k)$

    c.- Eliminar todas las dependencias $(X_k \rightarrow A_k) \in M_i$, tales que $A_k \in N_i$
  - Calcular $\mathbf{T_{i+1}}$ a partir de $\mathbf{T_i}$ eliminando todos los atributos que pertenecen a los conjuntos E^+_i y N_i
  - Calcular i = i + 1
  - Ir al paso 2

VIII. Algoritmo de simplificación-reducción → *Resultado*

- Como resultado se obtendrá un conjunto de DF’s menor. En el peor caso, el conjunto inicial de DF’s no varía.

- Interpretación del resultado:

  - Si el resultado es un conjunto de **DF’s vacío** (**$M_i=\varnothing$**), quiere decir que la **única clave del esquema** es el **descriptor formado por los atributos pertenecientes a la unión de las series de los diferentes E_i** calculados en el proceso.

  - Cuando el conjunto de **DF’s se reduce** ($Mi\neq\varnothing$), la **estructura de la clave estará dada de la siguiente forma**:
    - $K=\{\cup E_i\} \cup K'$, donde k’ es cualquier clave correspondiente al conjunto de DF’s restantes, obtenidas en el proceso de simplificación, es decir, el último M_i calculado.

  - Cuando el conjunto de **DF’s queda igual que el inicial**, quiere decir que no existe simplificación posible del conjunto de DF’s. Consecuentemente el algoritmo **no aporta ninguna información acerca de la clave**.

VIII. Algoritmo de simplificación-reducción → *Ejemplo 1*

Ejemplo1: Sea R=<T,M>, un esquema relacional

T={A,B,C,D,E,F,G}

M={AB→C, BD→F, BD→G, AE→C}

Paso 1:

$i = 0; T_0=T; M_0=M$

Paso 2:

$E_0 = T_0 - \{ \cup A_k \} = ABCDEFG - CFG = ABDE$

$N_0 = \{ \cup A_k \} - \{ \cup X_k \} = CFG - ABDE = \emptyset$

$P_0 = T_0 - (E_0 \cup N_0) = ABCDEFG - (ABDE \cup \emptyset) = CFG$

Paso 3:

$(M_i \neq \emptyset)$ o $(E_i \neq N_i \neq \emptyset)$

VIII. Algoritmo de simplificación-reducción → *Ejemplo 1*

- $T_0=\{A,B,C,D,E,F,G\}$
  $M_0=\{AB \rightarrow C,\ BD \rightarrow F,\ BD \rightarrow G,\ AE \rightarrow C\}$

Paso 4:
$(E_0)^+ = (ABDE)^+ = ABDECFG = T$

$M_1 = \{\emptyset\}$
$i = 1$
$T_1=\{\emptyset\}$

Paso 2b:
$E_1 = N_1 = P_1 = \emptyset$

Paso 3:
($M_1 = \varnothing$). El algoritmo concluye y la K = ABDE

VIII. Algoritmo de simplificación-reducción → *Ejemplo 2*

- Ejemplo2: Sea R=<T,M>, un esquema relacional

T={A,B,C,D,E,G}

M={AB→C, C→A, BC→D, D→E, D→G, BE→C, CG→B, CG→D, CE→G}

Paso 1:

$i = 0;\ T_0=T;\ M_0=M$

Paso 2:

$E_0 = T_0 - \{ \cup A_k \} = ABCDEG - ABCDEG = \emptyset$

$N_0 = \{ \cup A_k \} - \{ \cup X_k \} = ABCDEG - ABCDEG = \emptyset$

$P_0 = T_0 - (E_0 \cup N_0) = ABCDEG$

Paso 3:

$E_0 = N_0 = \emptyset$. El algoritmo concluye. Este es el peor de los casos. Ni hemos obtenido la clave, ni hemos podido reducir el conjunto M.

VIII. Algoritmo de simplificación-reducción → *Ejemplo 3*

Ejemplo3: Sea R=<T,M>, un esquema relacional

T={A,B,C,D}

M={A→B, BC→D, D→A}

Paso 1:

$i = 0;\ T_0=T;\ M_0=M$

Paso 2:

$E_0 = T_0 - \{ \cup A_k \} = ABCD - ABD = C$

$N_0 = \{ \cup A_k \} - \{ \cup X_k \} = ABD - ABCD = \emptyset$

$P_0 = T_0 - (E_0 \cup N_0) = ABD$

Paso 3:

$(M_0 \neq \emptyset)$ o $(E_0 \neq N_0 \neq \emptyset)$

VIII. Algoritmo de simplificación-reducción → *Ejemplo 3*

Ejemplo3: Sea R=<T,M>, un esquema relacional

T={A,B,C,D}

M={A→B, BC→D, D→A}

Paso 4:

$(E_0)^+ = (C)^+ = C$

$M_1 = \{A \rightarrow B, B \rightarrow D, D \rightarrow A\}$

i = 1

$T_1 = \{A,B,D\}$

Paso 2b:

$E_1 = \varnothing$

$N_1 = \varnothing$

$P_1 = \{A,B,D\}$

Paso 3:

- $E_1 = N_1 = \varnothing$. El algoritmo concluye. La clave K = {C} ∪ K'. Donde K' es la clave del esquema $R(T_1, M_1)$

IX. Algoritmo de síntesis

- Algoritmo para el cálculo de claves que se basa en el hecho de realizar una partición de **T** en 3 conjuntos de atributos:
  - **E**: conjunto de atributos <u>esenciales</u>.
  - **N**: conjunto de atributos <u>no esenciales</u>.
  - **P**: conjunto de atributos <u>posibles</u>.
- El algoritmo sólo usará 2 de estos tres conjuntos en el proceso de cálculo de la clave.
- El método en el que se basa este algoritmo para calcular la clave es bastante intuitivo:
  - Comienza con el conjunto **E** y va agregando grupos de 1,2,…,n atributos pertenecientes a **P** (n es la cardinalidad del conjunto P) comprobando si estos tienen la propiedad de ser una clave, excluyendo aquellos que contienen atributos redundantes, o aquellos previamente se hayan testeado de alguna manera.

IX. Algoritmo de síntesis
→ *Pasos*

Sea R = <T, M>, con **M** = $\{ X_i \rightarrow A_i \}_{i = 1..m}$ r.n.r y **T** = $E \cup P \cup N$.

- **Paso 1:**

  Calcular **E** = **T** – $\{ \cup A_i \}$

- **Paso 2:**

  Calcular $\mathbf{E^+}$

  Si ($\mathbf{E^+}$ = **T**)

  Entonces <u>**E** es la **única clave** del esquema</u>. **Fin del Algoritmo**.

  Sino Avanzar al **próximo paso**.

- **Paso 3:**

  Calcular **P** = **T** – ($\mathbf{E^+} \cup \mathbf{N}$), con **N** = $\{ \cup A_i \} - \{ \cup X_i \}$

  (eliminar de P los atributos pertenecientes a $\mathbf{E^+}$)

- **Paso 4:**

  Calcular $\mathbf{U_0} = \{ \mathbf{E} \cup A_k \}$, $\forall A_k \in \mathbf{P}$ y $\notin \mathbf{E^+}$

  (crear descriptores agregando atributos simples de P a E)

  Calcular **V** = $\emptyset$

  Calcular i = 0

IX. Algoritmo de síntesis → *Pasos*

Sea R = <T, M>, con **M** = $\{ X_i \rightarrow A_i \}_{i = 1..m}$ r.n.r y **T** = E ∪ P ∪ N.

- **Paso 5:**

  Calcular $\mathbf{W^+}$, $\forall \mathbf{W} \in \mathbf{U_i}$

  Si $\mathbf{W^+} = \mathbf{T}$

  Entonces incluir a **W** en **V**, y quitarlo de $\mathbf{U_i}$.

  Sino crear $\mathbf{U_{i+1}}$, partiendo de $\mathbf{U_i}$, reemplazando **W** por $\{\mathbf{W} \cup \mathbf{B_k}\}$, $\forall \mathbf{B_k} \in \mathbf{P}$ y $\notin \mathbf{E^+}$ donde:

  $\mathbf{B_k} \cap \mathbf{W^+} = \emptyset$ y ord(B_k)W > ord(A)W, siendo **A** el último atributo agregado a **W**.

- **Paso 6:**

  Si $\mathbf{S} \subseteq \mathbf{R}$, $\forall \mathbf{S,R}$ / $\mathbf{S} \in \mathbf{V}$ y $\mathbf{R} \in \mathbf{U_{i+1}}$

  entonces borrar **R** de $\mathbf{U_{i+1}}$ (se eliminan redundancias)

- **Paso 7:**

  Si $\mathbf{U_{i+1}}$ es vacío

  Entonces **V** contiene **todas las claves** del esquema. **Fin del algoritmo.**

  Sino Hacer **i = i+1**. Volver al paso **5**.

IX. Algoritmo de síntesis → *Resultado*

- Todos los descriptores contenidos en **V** son claves y no existe un subconjunto de ellos con la misma propiedad, es decir sintetiza **todas las claves** del esquema de relación.
- **Reduce el tiempo computacional**, ya que utiliza un conjunto limitado de atributos en el proceso.
- **No necesita ningún otro proceso adicional**.

IX. Algoritmo de síntesis
→ *Ejercicio 1*

- Ejemplo1: Sea R=<T,M>, un esquema relacional

T={A,B,C,D}

M={A→B, BC→D, D→A}

Paso 1:

E = **T** – { ∪ A_i } = ABCD - ABD

Paso 2:

E^+ = (C) $^+$ = C

Si ($\mathbf{E}^+$ = **T**) Entonces **E** es la **única clave** del esquema. **Fin del Algoritmo**.
Sino Avanzar al **próximo paso**.

Paso 3:

N = { ∪ A_k } – { ∪ X_k } = ABD – ABCD = ∅

P = T – ($\mathbf{E}^+$ ∪ N) = ABD (eliminar de P los atributos pertenecientes a $\mathbf{E}^+$)

Paso 4:

$\mathbf{U}_0$ = { **E** ∪ A_k }, ∀A_k ∈ **P** y ∉ $\mathbf{E}^+$; $\mathbf{U}_0$ = { CA, CB, CD },

V = ∅

i = 0

IX. Algoritmo de síntesis
→ *Ejercicio 1*

- Ejemplo1: Sea R=<T,M>, un esquema relacional

T={A,B,C,D}

M={A→B, BC→D, D→A}

Paso 5:

$(CA)^+ = CABD$

$(CB)^+ = CBDA$

$(CD)^+ = CDAB$

V= {CA, CB, CD}

$\mathbf{U}_1 = \{\emptyset\}$

Paso 6- 7:

Como U_1 es vacío, entonces V contiene todas las claves del esquema. Fin del algoritmo.

IX. Algoritmo de síntesis → *Ejercicio 2*

- Ejemplo2: Sea R=<T,M>, un esquema relacional

T={A,B,C,D,E,F,G}

M={AB→C, BD→F, BD→G, AE→C}

Paso 1:

E = **T** – { $\cup$ A_i } = ABCDEFG – CFG = ABDE

Paso 2:

E^+ = (ABDE) $^+$ = ABDECFG

Si (**E^+** = **T**) Entonces **E** es la **única clave** del esquema. **Fin del Algoritmo**.

IX. Algoritmo de síntesis

→ *Ejercicio 3*

- Ejemplo3: Sea R=<T,M>, un esquema relacional con T={A,B,C,D,E,F} y M={BF→E, BE→F, AF→D, D→A ,D→B, D→F , C→A}

Paso 1:

$\mathbf{E} = \mathbf{T} - \{ \cup A_i \} = ABCDEF - ABDEF = C$

Paso 2:

$E^+ = (C)^+ = CA$

Si ($\mathbf{E^+ \neq T}$) Avanzar al **próximo paso**.

Paso 3:

$N = \{ \cup A_k \} - \{ \cup X_k \} = ABDEF - ABCDEF = \emptyset$

$P = T - (E^+ \cup N) = BDEF$

Paso 4:

$\mathbf{U_0} = \{ \mathbf{E} \cup A_k \}, \forall A_k \in \mathbf{P}$ y $\notin \mathbf{E^+}$; $\mathbf{U_0} = \{ CB, CD, CE, CF \}$,

$\mathbf{V} = \emptyset$

$i = 0$

IX. Algoritmo de síntesis

→ *Ejercicio 3*

- Ejemplo3: Sea R=<T,M>, un esquema relacional con T={A,B,C,D,E,F} y M={BF→E, BE→F, AF→D, D→A ,D→B, D→F , C→A}

Paso 5:

$(CB)^+ = CBA$

$(CD)^+ = CDABFE = T$

$(CE)^+ = CEA$

$(CF)^+ = CFADBE$

V= {CD,CF}

$U_1 = \{CBD, CBE, CBF, CEF\}$

Paso 6:

$U_1 = \{CBE\}$

Paso 7:

Como $U_1 \neq \emptyset$, ir al paso 5

IX. Algoritmo de síntesis
→ *Ejercicio 3*

- Ejemplo3: Sea R=<T,M>, un esquema relacional con T={A,B,C,D,E,F} y M={BF→E, BE→F, AF→D, D→A ,D→B, D→F , C→A}

Paso 5b:
$(CBE)^+$ = CBEFAD

V= {CD,CF, CBE}
U_1= {∅}

Paso 6b- 7b:
Como U_1 es vacío, entonces V contiene todas las claves del esquema. Fin del algoritmo

Bibliografía

- *Ramez A. Elmasri, Shamkant B. Navathe*. ***Fundamentos de Sistemas de Bases de Datos*** *(5ª edic.).* Prentice-Hall. 2008. *[cap. 10]*
- De Miguel, A., Piattini. ***Fundamentos y modelos de bases de datos*** *(2ª edic.).*, Rama. [*cap. 8*]
- A. Silberschatz, Korth, Sudarshan. ***Fundamentos de Bases de Datos*** *(5ª edic.).* McGraw-Hill [*cap. 7*]