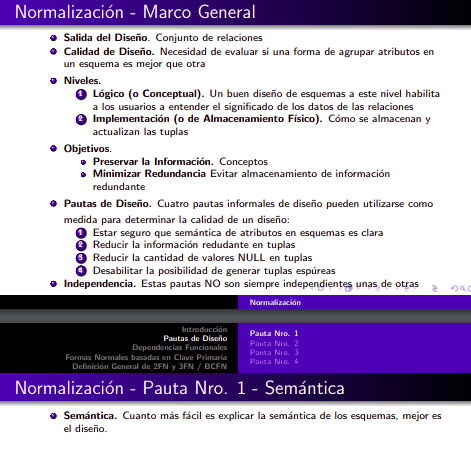
**Marco general**



-Almacenamiento: Minimizar espacio de almacenamiento mediante el diseño, evitando redundancia por ejemplo. No almacenando Natural Joins, ya que pueden introducir además anomalías de inserción, deleción o modificación (tuplas espúreas, borrado de información, etc.).

-Manejo de NULLs:

-Semánticamente, pueden llevar a diversas interpretaciones (no se conoce el dato, el dato no aplica a la tupla)

- Deben ser excepcionales, y cuando los hay, debe ser documentada su interpretación.

-Evitar tuplas espúreas:

-Diseñar esquemas tal que puedan ser relacionados por atributos que se encuentren apropiadamente relacionados por medio de condiciones de igualdad entre ellos (clave primaria, clave foránea), para evitar generación de tuplas espúreas.

-Evitar relaciones que contengan atributos de matching que no sean combinación de claves foránea/clave primaria porque JOINS sobre ellos pueden producir tuplas espúreas

Preguntas de repaso

a. Comente las anomalías de inserción, borrado y modificación. ¿Por qué están consideradas como malas? Ilustre sus comentarios con ejemplos.

b. ¿Por qué deben evitarse en la medida de lo posible los valores NULL en una relación? Comente el problema de las tuplas espúreas y cómo pueden prevenirse.

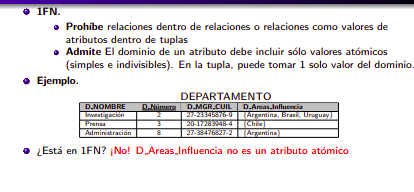
c. ¿Qué es una dependencia funcional? ¿Cuáles son las posibles fuentes de información que definen las dependencias funcionales que se cumplen entre los atributos de un esquema de relación?

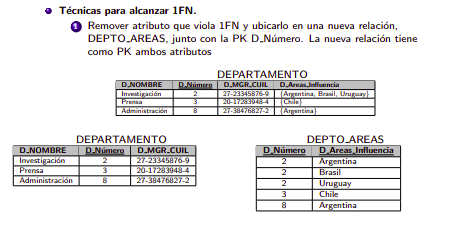
d. ¿Por qué no podemos inferir automáticamente una dependencia funcional de un estado de relación particular?

Ya que hay cuestiones semánticas, entre las relaciones y significados de los atributos, que las define quien diseña el conjunto de relaciones.

**Instancias legales**. r (R) que satisface restricciones de DF se denomina instancia legal, estado legal o extensión legal de R

1FN





e. ¿Qué dependencias no deseables se evitan cuando una relación está en 2FN?

**DF Completa**. Una DF X → Y es Completa si al eliminar algún atributo A de

X la DF deja de existir

**DF Parcial**. Una DF X → Y es Parcial si es posible eliminar algún atributo A

de X y la DF continúa existiendo

**2FN**. Un esquema R está en 2FN si todo atributo no primo A de R depende funcionalmente de manera completa de la PK de R

Ejemplo que queremos evitar: Una tupla con atributos Tipo Documento, DNI y Edad, donde Tipo Documento-DNI es clave primaria, y Edad depende funcionalmente sólo del DNI. Si elimino Tipo Documento, no se eliminará la dependencia funcional, que es algo que queremos evitar.

f. ¿Qué dependencias no deseables se evitan cuando una relación está en 3FN?

**Súper Clave (SK)**. Una SK de R = {A1, A2, ..., An} es un subconjunto de

atributos S ⊆ R con la propiedad de que no hay dos tuplas t1, t2 en un estado

legal r (R) que cumplan t1(S) = t2(S)

**Clave (K)**. Una clave K es una SK con la propiedad adicional de que al remover

cualquier atributo de K, deja de ser SK. Es decir, K es una SK minimal

**Clave Candidata (CK)**. Si un esquema posee m´as de una clave, cada una de

ellas se denominan clave candidata

**Clave Primaria (PK)**. Una de las CK es designada arbitrariamente como PK

**Clave Secundaria**. CK que no es PK

Atributo primo. Atributo de un esquema R que pertenece a alguna CK de R

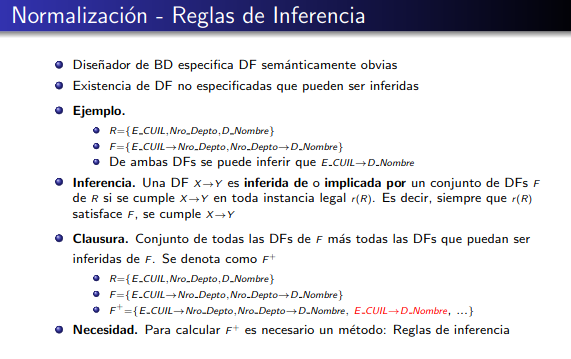
**Requisito**. En la pr´actica, todos los esquemas deben poseer PK

Ejercicio. Proponer un Esquema con todos estos elementos e identificarlos

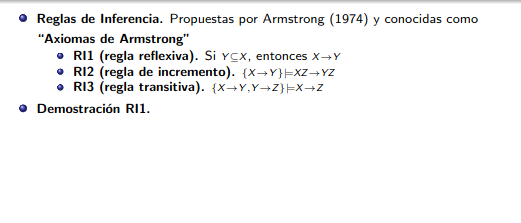
----------------------------------------------------------------------------------------------------------------------

**Reglas de inferencia (19/04)**

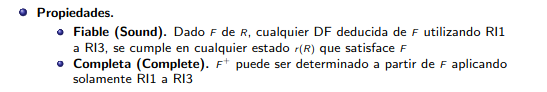
Conceptos (Inferencia de dependencias funcionales, Clausura)



Axiomas de Armstrong

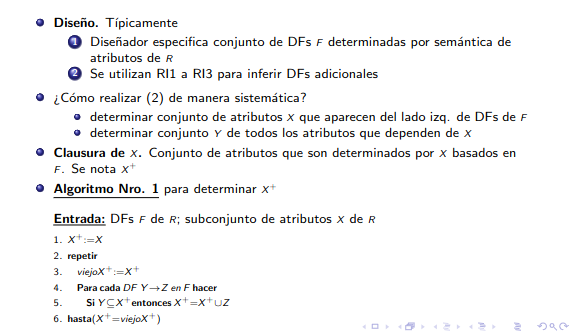


Propiedades:

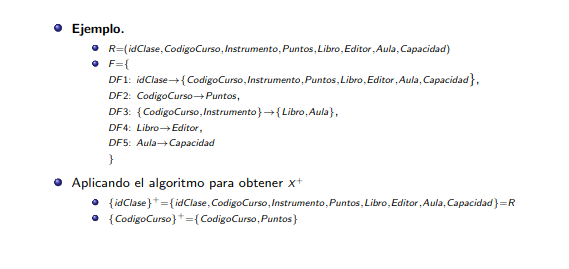


¿Cómo se determinan sistemáticamente la **clausura**? (dependencias funcionales, tanto semánticas como inferidas)

Ver el siguiente algoritmo:



Ejemplo de como funciona el ciclo

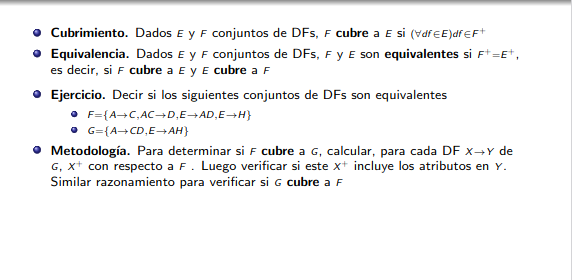


Así, hasta tener un conjunto de atributos del lado derecho para cada elemento del lado izquierdo de las dependencias funcionals.

{CodigoCurso, Instrumento}+ = R-{idClase} => No es clave candidata {CodigoCurso, Instrumento}

Hay que hacer varias pasadas, hasta dejar de agregar atributos a la clausura.

**Cubrimiento y equivalencia de conjuntos de DFs**

****

F+ = {{A}+, {AC}+, {E}+}

G+ = {{A}+, {E}+}

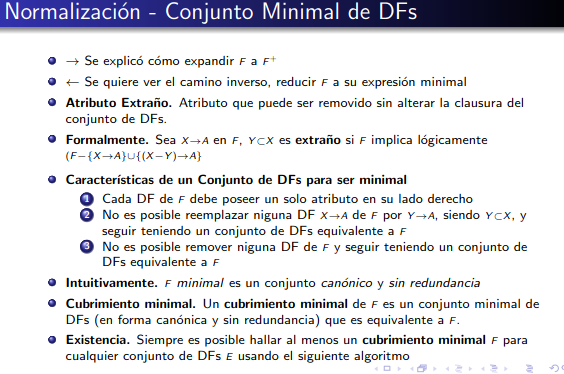
Para que sean equivalentes, debemos determinar si F+=G+

F+={{A, C, D}, {A, C, D}, {E, A, D, C, H}}

G+={{A, C, D}, {E, A, H, C, D}}

**Reducir una clausura a una expresión minimal (cubrimiento minimal)**

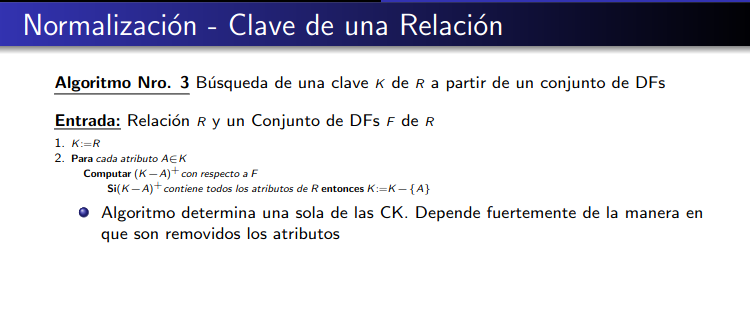
**Def: Atributo extraño**



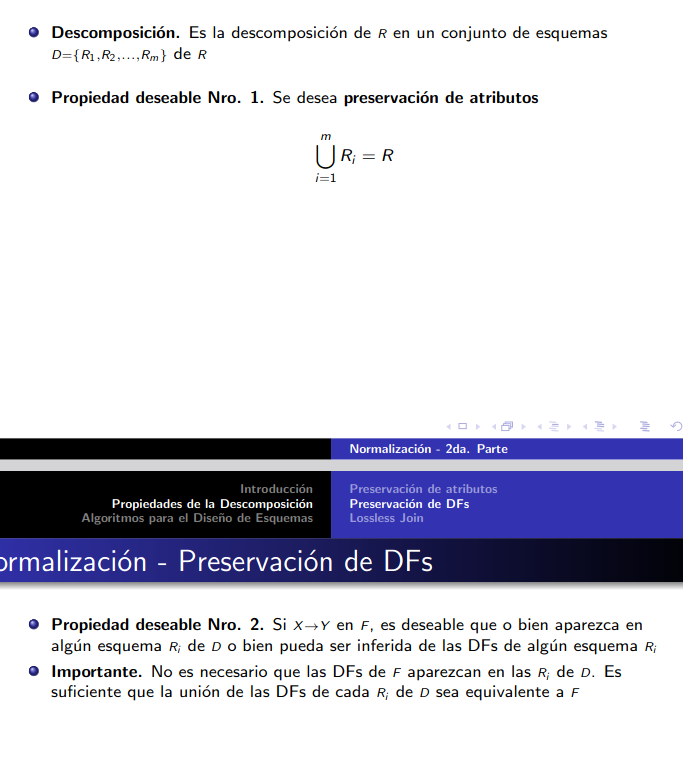
Forma canónica: Todas las partes de lado derecho de las dependencias tienen un solo atributo.

Prop: El cubrimiento minimal NO es único.

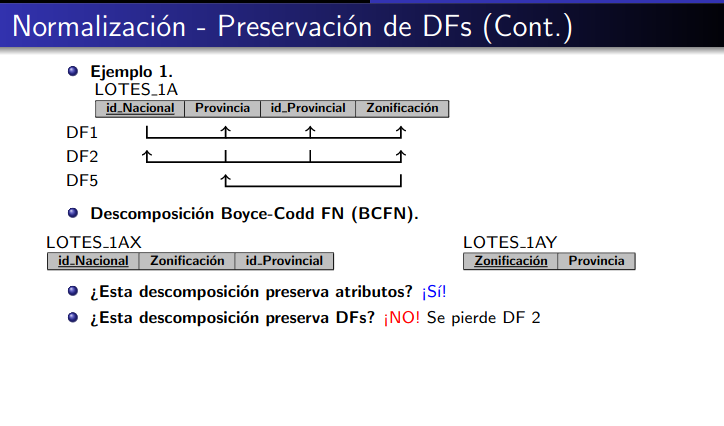
Algoritmo: Determinar clave de una relación (Algoritmo)



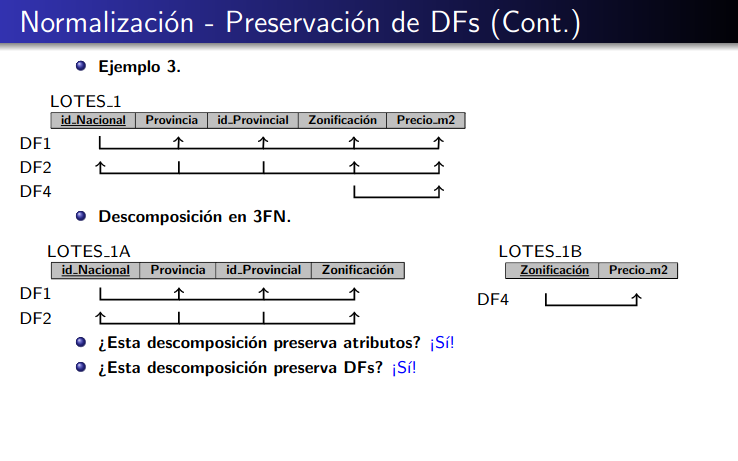
**Descomposición**

****

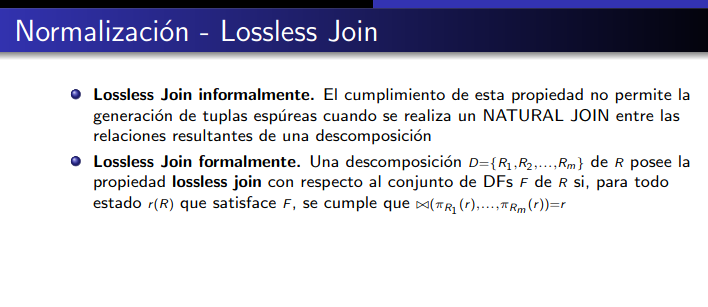
**Ejemplo de Descomposición de Boyce Codd**

****

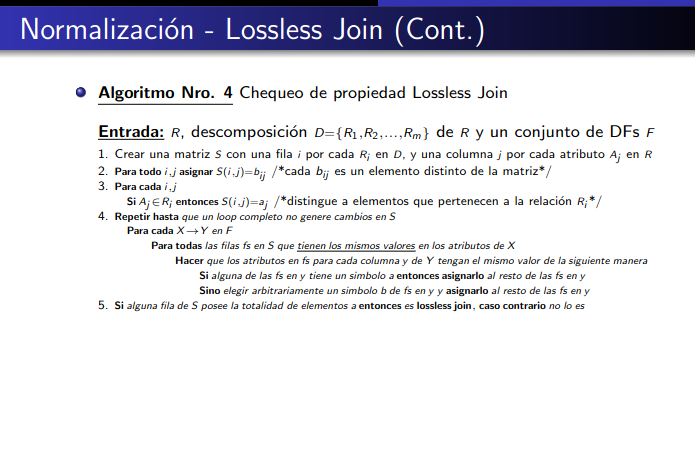
**EN 3FN se garantiza la existencia de una descomposición que preserva atributos y DFs**

****

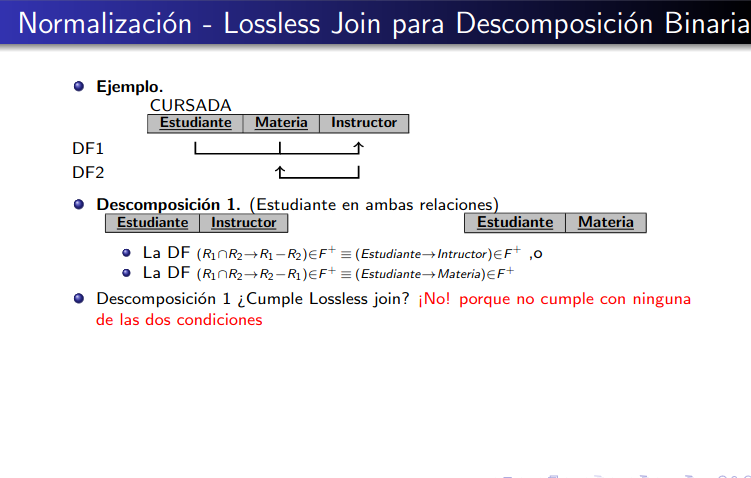
**Lossless Join**

****

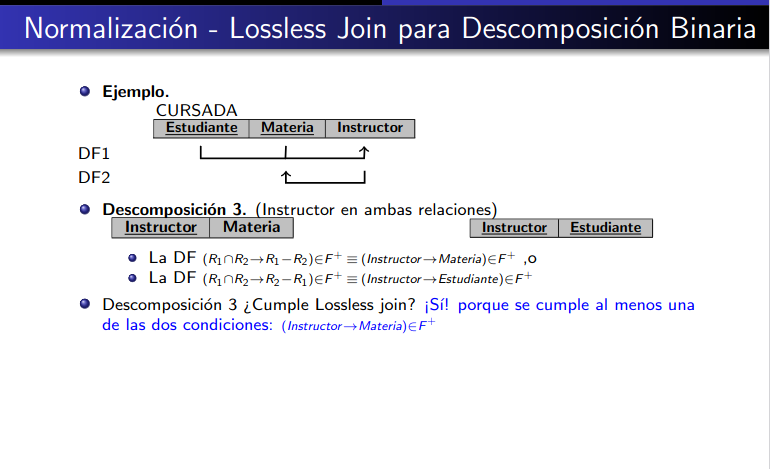
**Algoritmo chequeo Lossless Join (Para cualquier tipo de descomposición)**

****

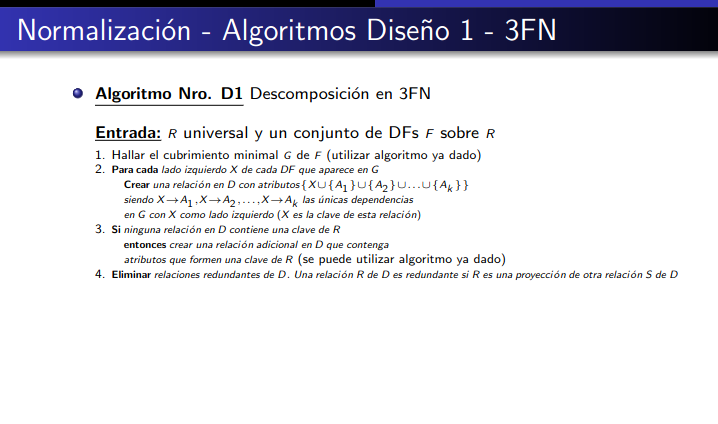
**Algoritmo chequeo Lossless Join para descomposiciones binarias (dos tablas, más sencillo)**

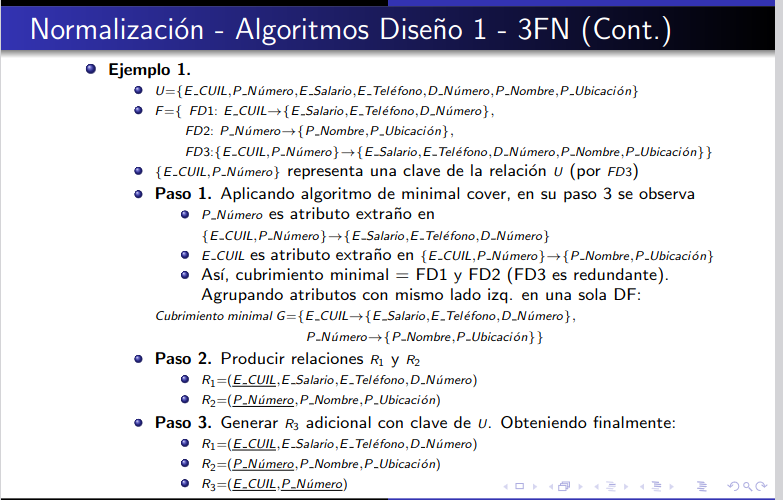
****

**Una forma que cumple**

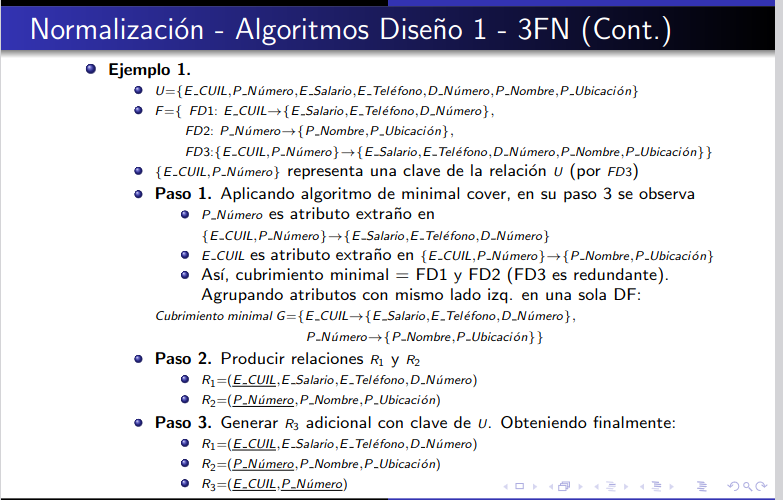
****

**Algoritmo para encontrar descomposición en 3FN que cumpla Lossless Join**

****

****

**Prop: Para BCFN también se puede garantizar la existencia de una descomposición.**

****