Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga

Administración de Bases de Datos

(Ingeniería Técnica en Informática de Gestión)

Normalización de Sistemas Relacionales

(Dependencias Funcionales)

E.T.S.I. Informática ***J. Galindo Gómez***

Implementar Buenos Esquemas

- Existen muchas formas de **implementar** un Modelo Conceptual en un **esquema de BDR** (tablas, atributos...).
- Hay **dos niveles para evaluar** qué implementación es mejor:
  - **Nivel Lógico:** Un buen esquema relacional hace que los usuarios entiendan mejor la BD y puedan consultarla correctamente.
    - Aquí, las **vistas** ayudan a ver y comprender mejor la BD.
  - **Nivel de Implementación:** Estudia cómo se almacenan y actualizan las tuplas de las relaciones de la BD (no las vistas).
- **Medidas informales** para evaluar la calidad de un esquema:
  - **1. Semántica de los Atributos:** Significado claro.
  - **2. Reducir Valores Redundantes:** Reducir el almacenamiento.
  - **3. Reducir Valores NULL:** Mejor crear una nueva relación.
  - **4. No Permitir Tuplas Falsas:** Reuniones sobre atributos que son llaves primarias o llaves externas.

Implementar Buenos Esquemas

- **1. Semántica de los Atributos:**
  - El esquema será mejor, cuanto más fácil sea **explicar el significado** de una relación y sus atributos.
  - En general, no combinar en una única tabla atributos de varios tipos de entidad o relaciones.
- **2. Reducir Valores Redundantes:** Reducir el almacenamiento.
  - En general, es mejor almacenar dos relaciones independientes relacionadas entre sí por una llave externa que almacenar una Reunión entre ellas, lo cual llevaría a almacenar varias veces los mismos valores en distintas tuplas. Eso conlleva problemas también en las **actualizaciones**, **inserciones** y **borrados**.
- **3. Reducir Valores NULL:**
  - Evitar poner atributos en relaciones base (principales) que tomarán frecuentemente el valor Null. En ese caso es mejor **crear una relación aparte** que tenga tales atributos y la llave primaria de la relación base.
- **4. No Permitir Tuplas Falsas** (*spurious tuples*)**:**
  - Intentar evitar relaciones con atributos (no llave externa) cuya información puede obtenerse por otras vías y, en tal caso, no efectuar operaciones de Reunión sobre dichos atributos.
  - Las **operaciones de Reunión deben hacerse sobre atributos que son o llaves primarias o llaves externas**, para evitar la generación de tuplas falsas.

Dependencias Funcionales

- Sea un esquema **R = {A_1, A_2 , ... A_n}**, con **X** e **Y** siendo dos conjuntos de tales atributos. La **Dependencia Funcional (DF)** **X ® Y** especifica una restricción en **R**:

  X ® Y indica que " **t,u** Î **R: t[X] = u[X]** Þ **t[Y] = u[Y]**

  - **Y depende funcionalmente de X:** En una tupla, los valores de **Y** están determinados por (o dependen de) los valores de **X**.
  - Si **X** es una llave candidata, se cumple **X ® Y**, para cualquier **Y**.
  - **X ® Y**, no implica **Y ® X**.
  - Es una **propiedad de la semántica** de los atributos: No pueden extraerse DFs automáticamente en una relación dada.
  - De un conjunto **F** de DFs pueden deducirse otras DFs. Todas las DFs posibles se llaman **clausura de F**, representada por **F^+**:
    - **1.** Regla Reflexiva: Si **Y** Í **X**, entonces **X ® Y**.
    - **2.** Regla de Aumentación: **{X ® Y}** Þ **{XZ ® YZ}**.
    - **3.** Regla Transitiva: **{X ® Y, Y ® Z}** Þ **{X ® Z}**.

    Reglas de Inferencia de Armstrong: Son coherentes y completas (primitivas).

    - **4.** Regla de Descomposición: **{X ® YZ}** Þ **{X ® Y}** (Aplicar Reglas 1 y 3).
    - **5.** Regla de Unión: **{X ® Y, X ® Z}** Þ **{X ® YZ}** (R. 2 y 6).
    - **6.** Regla Pseudotransitiva: **{X ® Y, WY ® Z}** Þ **{WX ® Z}** (R. 2 y 3).

Dependencias Funcionales

- **Clausura de X en F (X^+):** Conjunto de todos los atributos que son funcionalmente dependientes de **X**. Algoritmo para calcular X^+:

```
X+ := X; (* Regla Reflexiva *)
REPEAT
  viejoX+ := X+;
  PARA cada dependencia en F, Y → Z, HACER
    IF Y ⊆ X+ THEN X+ := X+ ∪ Z;
UNTIL (X+=viejoX+);(* Hasta que no haya modificaciones en X+ *)
```

- Sean **F** y **E** dos conjuntos de Dependencias Funcionales:
  - **F cubre** a **E**, si cada DF de **E** está en F^+**:** En cada DF del tipo $X \rightarrow Y$ en **E,** calcular X^+ con respecto a **F** y comprobar si X^+ incluye los atributos de **Y**. Si eso se cumple para todas las DF de **E** entonces **F** cubre a **E**.
  - **F equivale** a **E**, si $F^+=E^+$ **:** Si **F** cubre a **E** y también **E** cubre a **F**.
- Un conjunto **F** de Dependencias Funcionales es **mínimo** si:
  - **1.** Si cada DF de **F** tiene un único atributo en la parte derecha.
  - **2.** Si $X \rightarrow A$ no puede reemplazarse por $Y \rightarrow A$, donde $Y \subset X$, siendo, con ese cambio, equivalente a **F**.
  - **3.** No puede eliminarse una DF de **F**, manteniendo la equivalencia a **F**.

Normalización

- Es un proceso para conseguir que un esquema relacional cumpla ciertas condiciones. Existen distintos niveles o **Formas Normales:**
  - Existen tres **Formas Normales** básicas (Codd, 1972): **1ª, 2ª y 3ª FN**.
  - Posteriormente, se propuso la tercera **Forma Normal de Boyce-Codd**, más restrictiva que la 3ª FN. Todas se basan en las DF existentes.
  - La **4ª y 5ª FN** se basan en dependencias multivaluadas y de reunión.
- **Ventajas de la Normalización:**
  - Minimizar la **redundancia**.
  - Minimizar los **problemas al insertar, borrar y actualizar**.
- **La normalización no garantiza un buen diseño** de una base de datos. Se necesitan **Características Adicionales** que pueden estudiarse durante la **Normalización por Descomposición:**
  - **Reunión sin pérdidas** (*lossless join*): Garantiza que no ocurrirá el problema de la generación de tuplas falsas tras *descomponer* una tabla en varias. La reunión de esas tablas genera la tabla original.
  - **Conservación de dependencias** (*dependency preservation*): Tras una *descomposición* cada DF debe seguir estando representada.

Formas Normales Básicas: 1FN

- **Primera Forma Normal (1FN):** El dominio de un atributo debe incluir únicamente valores atómicos (simples, indivisibles).
  - **La 1FN es básica** en una relación típica del modelo relacional, pero esta FN se elimina en el modelo relacional dirigido a objetos.
  - **No** se aceptan valores **multivaluados** (ej: varias direcciones en el *mismo* atributo) o **compuestos** (dirección={calle, piso, CP, ciudad...}).
  - **Técnicas** para conseguir la 1FN si existen atributos **multivaluados**:
    - **1.** Poner cada atributo multivaluado **M** en una relación independiente, junto con la llave primaria **K** de la relación de partida. La llave primaria de la nueva relación será la unión de **K** y **M**. Suele ser el mejor método.
    - **2.** Incluir el atributo multivaluado **M** en la llave de la relación. Este método tiene el problema de introducir redundancia en la relación (en el resto de atributos distintos a **K** y **M**), por lo que no debe usarse.
    - **3.** Si sabemos el máximo número de valores **N** del atributo multivaluado, insertar **N** atributos: Atributo1, Atributo2, ... AtributoN. El problema puede ser el alto número de NULL's en las tuplas con menos de **N** valores.
  - **Relaciones Anidadas:** No se acepta una tabla como valor de un atributo. Solución: Crear una nueva tabla con dicho atributo-tabla, incluyendo la llave primaria **K** de la relación original.
    - La llave primaria de la nueva relación incluirá los atributos de **K**.

Formas Normales Básicas: 2FN

- **Segunda Forma Normal (2FN):** Cada atributo **A** de una relación **R** es dependiente funcional y **completamente** de la llave primaria **K** de **R**.
  - **Dependencia Funcional Completa K ® A** (no **parcial**)**:** Si no existe un atributo **B** de **K** tal que, si lo quitamos siga valiendo **{K - B} ® A**.
  - **Normalización a 2FN:** Dividir en relaciones tales que la llave primaria haga depender funcional y completamente al resto de los atributos. Los atributos **B** se quitarán de **K** en la tabla que incluya al atributo **A**.
  - **Ejemplo:** Sea una relación de animales, donde el color es un atributo multivaluado y optamos por utilizar la segunda técnica mostrada, para conseguir una relación en 1FN (Atributo Color sería parte de la llave):
    - **Mascota** (NIFPropietario, Nombre, Color, Especie, Raza);
    - **No está en 2FN**, porque existe un atributo (Color) que está en la llave y del que no dependen atributos como Especie o Raza. Obsérvese, además, la redundancia que se produciría en dichos atributos en las tuplas con igual valor de {NIFPropietario, Nombre}.
    - **Soluciones:**
      - Dividir en dos relaciones: **Mascota** (NIFPropietario, Nombre, Especie, Raza); **Colores** (NIFPropietario, Nombre, Color);
      - Suponer un número máximo de colores:
        Mascota (NIFPropietario, Nombre, Especie, Raza, Color1, Color2);

Formas Normales Básicas: 3FN

- **Tercera Forma Normal (3FN):** No deben existir dependencias **transitivas** de atributos no llave, respecto de la llave primaria: Atributos no llave no deben depender funcionalmente de otros atributos no llave.
  - **Dependencia Transitiva X ® Y:** Si existe un conjunto de atributos Z, que no son llave candidata ni un subconjunto de ninguna llave de la relación, y que además, cumple que **X ® Z** y **Z ® Y**.
  - **Ejemplo:** La relación siguiente está en 2FN y no en 3FN:
    - **EMP_DEPT**(NIF, NombreEmp, Dirección, DptoNum, NombreDpto).
    - Se cumple que: NIF ® DptoNum, y que DptoNum ® NombreDpto.
  - **Solución:** Separar en dos tablas, dejando en una los atributos **XZ** y en otra los atributos **ZY**, de forma que una reunión natural utilizando los atributos **Z** recuperaría la tabla original. O sea, los atributos no llave (**Y**) que dependen de otros no llave (**Z**) se pondrán en una tabla.
    - **En el ejemplo:**
      - **Emp**(NIF, NombreEmp, Dirección, DptoNum)
      - **Dpto**(DptoNum, NombreDpto).

2FN y 3FN Generalizadas

- Las **dependencias Completas y Transitivas** se pueden considerar respecto a **TODAS las llaves candidatas** (no sólo la primaria).
- **2FN:** Si, en una relación, cada atributo que no pertenece a una llave no es parcialmente dependiente de cualquier llave candidata.
  - Cualquier llave candidata dependerá a la fuerza de **K** (llave primaria). Esa dependencia, junto con la que incumple la 2FN, forman una dependencia transitiva. Entonces, el problema será eliminado al hacer cumplir la 3FN. **K ®** (llave candidata) **A**
- **3FN:** Si en cualquier dependencia no trivial **X ® A** se cumple que **X** es una superllave, o que **A** es miembro de una llave candidata de la relación.
  - Si eso se incumple, resultaría uno de los casos siguientes:
    - **X** no es superllave**: X** no pertenece a una llave (tenemos una d. transitiva), o bien, **X** es parte de una llave (tenemos una dep. parcial, no 2FN).
    - **A** no es miembro de una llave candidata: Tenemos una dependencia transitiva: **K ® X ® A** que debe eliminarse dividiendo en dos tablas. Si **A** es miembro de una llave, la relación no está, como se verá, en **FNBC**.
  - Otra definición de **3FN**: Cada atributo que no es miembro de una llave:
    - Es totalmente dependiente de cada llave candidata, **y**
    - No es transitivamente dependiente de ninguna llave candidata.

FNBC: Forma Normal de Boyce-Codd

- **FNBC:** Si en cualquier dependencia no trivial **X** ® **A** se cumple que **X** es una superllave. Es una **FN más restrictiva que la 3FN**: NO se permite que **A** sea miembro de una llave candidata.
  - **Formato** de una relación **R** en 3FN y NO en FNBC**:** **R(A,B,C)**, en la que ocurre que **{A,B}** ® **C** y además, **C** ® **B**.
  - **Solución:** Dividir en dos tablas: una con los atributos **CB** y otra con los atributos **AC**, donde **AC** será una nueva llave candidata.
    - Otras divisiones en 2 tablas, como (**AB** y **BC**) o (**AB** y **AC**), no son buenas: Producen **tuplas falsas** al efectuar una reunión (sobre el atributo común).
  - En general, **la mayoría de las relaciones en 3FN están también en FNBC:** Eso no se cumple si hay una dependencia **X** ® **A**, en la que **X** no sea superllave y **A** sea miembro de una llave candidata.
    - Cualquier relación con sólo dos atributos está en FNBC.
  - **Es muy deseable** que todas las relaciones estén en FNBC.
    - **La FNBC no garantiza un buen diseño del esquema de la BD:** La descomposición o división en tablas debe ser considerada globalmente, además de exigir cierta normalización a cada tabla individualmente.
    - Las siguientes tablas están en FNBC y al efectuar una reunión sobre ellas se podrían producir tuplas falsas (si hay proyectos en la misma ciudad): **Empleado** (NIF, CiudadProy) y **Proyecto** (Proy#, Nombre, CiudadProy).

FNBC: Ejemplo

- **Relación en 3FN: Curso** (Estudiante, Asignatura, Profesor).
- **DF:** {Estudiante, Asignatura} ® Profesor**,** Profesor ® Asignatura.
  - Si un profesor puede impartir varias asignaturas, no se cumplirá la dependencia Profesor ® Asignatura.
- **Datos en la Relación Curso:**

Estudiante	Asignatura	Profesor
Santiago	B.D.	Pepe
Jesús	B.D.	Pepe
Jesús	S.O.	Juan
María	B.D.	Paco

- **Solución:** Dividir en 2 tablas:
  - **Curso** (Estudiante, Profesor) y **Asig_Prof** (Profesor, Asignatura).

Estudiante	Profesor
Santiago	Pepe
Jesús	Pepe
Jesús	Juan
María	Paco

Profesor	Asignatura
Pepe	B.D.
Juan	S.O.
Paco	B.D.

Resultado de la Reunión entre Curso y Asig_Prof (Reunión **SIN** Pérdidas).

- **Tuplas falsas** al hacer una reunión con otras posibles divisiones:
  - (Estudiante, Asignatura) y (Asignatura, Profesor):
    - Falsas tuplas: (Santiago, B.D., Paco), (Jesús, B.D., Paco) y (María, B.D., Pepe).
  - (Estudiante, Asignatura) y (Estudiante, Profesor):
    - Falsa tuplas: (Jesús, S.O., Pepe) y (Jesús, B.D., Juan).

FN de una Relación ya en 1FN

Analizar TODAS las DFs de la relación con la forma X ® A : La FN de la relación será la **menor FN** (1FN < 2FN < 3FN < FNBC).

X Superllave	A Í llave candidata	Resultado
SI	**SI**	**FNBC**
SI	**NO**	**FNBC**
NO (X Ì llave)	**SI**	**1FN (No 2FN)**
NO (X Ë llave)	**SI**	**3FN (No FNBC)**
NO (X Ì llave)	**NO**	**1FN (No 2FN)**
NO (X Ë llave)	**NO**	**2FN (No 3FN)**

Características de un Buen Diseño

- Hemos visto que, para evitar determinados problemas, el esquema de una BD debe **dividirse** o **descomponerse** en m tablas $\mathbf{R_i}$, i=1,...m, (partiendo de la llamada **Relación Universal R**, que sería una tabla con todos los atributos de la BD).
  - **Conservación de Atributos** (*Attribute Preservation*)**:** Todos los atributos deben aparecer en al menos una relación, tras la descomposición en tablas: $\bigcup_{i=1}^{m} R_i = R$
  - **Conservación de Dependencias** (*Dependency Preservation*)**:** Cada DF debe aparecer en alguna relación $\mathbf{R_i}$, o debe poderse inferir de las DF existentes.
    - Cada DF representa una **restricción** en la BD.
      - Si una DF no está representada en una relación individual, para exigir el cumplimiento de dicha restricción será necesario utilizar varias tablas: Efectuar una o varias reuniones y luego comprobar el cumplimiento de la DF. Esto es ineficiente y poco práctico.
    - **Algoritmo de Síntesis Relacional** (*relational synthesis algorithm*)**:** Algoritmo para crear una descomposición en relaciones en 3FN conservando las dependencias, partiendo de una Relación Universal y un conjunto de DF. Este algoritmo de descomposición no garantiza la propiedad de reunión sin pérdidas y no se incluye aquí.

Características de un Buen Diseño

- **Reunión SIN Pérdidas o SIN añadidos** (*Lossless Join or Nonadditive Join*): La reunión natural de las tablas de una descomposición deben dar como resultado la tabla original, sin tuplas falsas.
  - Se supone que los atributos de reunión no admiten el valor NULL
  - Existe un **Algoritmo** para comprobar si una descomposición cumple esta propiedad con respecto a un determinado conjunto de DF.
  - **Propiedad 1:** Una descomposición binaria **{R_1,R_2}** de **R**, cumple la propiedad de reunión sin pérdidas con respecto a un conjunto **F** de DF, si y sólo si cumple una de las siguientes DF está en **F^+** :
    - **($R_1 \cap R_2$) ® (R_1 - R_2)**, o bien **($R_1 \cap R_2$) ® (R_2 - R_1)**. O sea, que los atributos comunes a **R_1** y **R_2** son una llave de una de las dos relaciones.
  - **Propiedad 2:** Sup. **R** con una descomposición sin pérdidas, donde existe una relación **R_i** con otra descomposición sin pérdidas. Entonces, si sustituimos **R_i** en la descomposición de **R**, por su descomposición, el resultado es otra descomposición de **R** también sin pérdidas.
- Existe un **Algoritmo** que partiendo de una relación universal **R** y de un conjunto **F** de DF y utilizando las propiedades anteriores, crea una descomposición sin pérdidas en relaciones en FNBC, pero el resultado no tiene la propiedad de conservación de dependencias.

Características de un Buen Diseño

- **Algoritmo de síntesis relacional con conservación de dependencias y reunión sin pérdidas,** partiendo de una relación universal **R** y de un conjunto **F** de DF **mínimo** (si F no es mínimo puede obtenerse un cubrimiento mínimo mediante un algoritmo: Pueden existir varios y la bondad del resultado depende de él)**:**
  - 1. **PARA** cada **X** que aparezca en la parte izquierda de una DF de **F**:
    - Tomar todas sus DF: **X®A1, X®A2, ... X®Ak**;
    - Crear una relación con llave **X** y atributos **{X, A1, A2... Ak}**;
  - 2. **SI** ninguna de las relaciones de la descomposición anterior tiene una llave de la relación universal **R**

    ENTONCES

    Crear una relación más con los atributos de la llave de **R**;
- **Obtener una posible llave de R:** Se supone que la llave **K** son todos los atributos de **R** y se van eliminando aquellos que no afectan a la condición de llave, es decir, se eliminan aquellos atributos **A** para los que la clausura de **K - A**, representada por **$(K - A)^+$**, incluye todo **R**.
- **La tablas de la descomposición del algoritmo anterior están en 3FN:**
  - Por tanto, podemos comprobar **si alguna relación no está en FNBC** y, en tal caso, escoger entre descomponerla o no, sabiendo que la descomposición para conseguir **la FNBC supone perder una DF**.
  - **El resultado no tiene que ser el mejor posible**, incluso suponiendo que se hallan especificado TODAS las DF, lo cual no es fácil en grandes BD.

Dep. Multivaluadas (DMV) o Plurales

- **DMV X ®® Y en R, con X,Y Î R (X multidetermina a Y):** Si se cumple que, si existen dos tuplas t_1 y t_2 con $t_1[X]=t_2[X]$, entonces deben existir las tuplas t_3 y t_4 con las siguientes propiedades, donde **Z = R - (X ∪ Y):**
  - $t_1[X] = t_2[X] = t_3[X] = t_4[X]$
  - $t_1[Y] = t_3[Y]$ y $t_2[Y] = t_4[Y]$
  - $t_1[Z] = t_4[Z]$ y $t_2[Z] = t_3[Z]$

  t_1, t_2, t_3 y t_4 no son necesariamente distintas
- Por tanto, **X ®® Y, implica que también se da X ®® Z:** **X ®® Y | Z**
- **Interpretaciones de X ®® Y:**
  1. Un valor de **X** puede tener varios valores de **Y**, los cuales se relacionarán con **TODOS** los valores de **Z** que se relacionen con dicho **X**.
  2. A cada valor de **X** pueden corresponder varios valores de **Y** y varios de **Z**, y que éstos son independientes de aquellos.
- **DMV Trivial:** Si **Y Ì X**, o si **R = X ∪ Y (Z=Æ).**
- En general, el conj. de valores **Y** y el de **Z dependen SÓLO de X.**
  - **Y es un atributo multivaluado.**
  - Para evitar una falsa relación entre **Y** y **Z**, debe repetirse cada valor de **Y** con **todos** los valores de **Z:**
    - Esta **redundancia** es indeseable, pero la relación está en **FNBC** (no hay DF).
    - **Solución:** Aplicar la **4FN** (Cuarta Forma Normal).

DMV: Ejemplo y Reglas de Inferencia

- **Ejemplo: Empleado**(Nombre, Proyecto, Hijo),
  - donde **Nombre** tiene dos relaciones **1:N** (con Proyecto y con Hijo):
    - Nombre ®® Proyecto | Hijo
    - Esa relación está en **FNBC** y tiene problemas de redundancia (no está en **4FN**):

Nombre	Proyecto	Hijo
Príamo	Caballo	Casandra
Príamo	Troya	Paris
Príamo	Caballo	Paris
Príamo	Troya	Casandra

- **Reglas de Inferencia:** A las 3 reglas de Inferencia de Armstrong podemos añadir otras sobre DMV:
  - **4.** Regla de Complementación de DMV: **{X ®® Y}** Þ **{X ®® (R - (X ∪ Y))}**.
  - **5.** Regla de Aumentación para DMV: **{X ®® Y, Z Í W}** Þ **{WX ®® YZ}**.
  - **6.** Regla Transitiva para DMV: **{X ®® Y, Y ®® Z}** Þ **{X ®® (Z - Y)}**.
  - **7.** Regla de Paso de DF a DMV: **{X ® Y}** Þ **{X ®® Y}** (implica que **FNBC Ì 4FN**).
  - **8.** Regla de Fusión de DF y DMV: **{X ®® Y, W∩Y=Æ, W ® Z, Z Í Y}** Þ **{X ® Z}**.
    - Ejemplo: {Nombre $\twoheadrightarrow$ Proyecto, Hijo $\rightarrow$ Proyecto} $\Rightarrow$ {Nombre $\rightarrow$ Proyecto}.

4FN (Cuarta Forma Normal)

- **Cuarta Forma Normal (4FN):** Si para cada dependencia multivaluada no trivial **X ®® Y,** que hay en la clausura de su conjunto de dependencias **F** (que incluye DF y DMV), **X** es una **superllave.**
  - **Solución:** Dividir en dos relaciones (con **X** en la llave): **{X, Y}** y **{R - Y}.** **{X, Z}**
    - Al dividir, las **DMV** no se pierden, sino que se convierten en **triviales**.
    - **Del ejemplo:**

Nombre	Proyecto
Príamo	Caballo
Príamo	Troya

Nombre	Hijo
Príamo	Casandra
Príamo	Paris

  - **Ventajas:**
    - **Ahorrar Espacio** de Almacenamiento.
      - **Ej.:**No duplicar el nombre de cada hijo.
    - **Eliminar la Redundancia**.
    - **Eliminar Problemas al Insertar, Borrar o Modificar**.
      - **Ej.:** Modificar el nombre de un hijo supone modificarlo en todos los lugares donde aparezca.
      - **Ej.:** Si insertamos un proyecto, debemos insertar tantas tuplas como hijos tenga el empleado, para evitar violar la DMV y que se pueda establecer una relación inexistente entre los atributos de Proyecto e Hijo.

DMV y 4FN: Ejemplo de una Fábrica

- **Fábrica (**Empleado, Máquina, Producto**),** con las siguientes dependencias**:**
  - **D1:** Empleado ® Máquina (cada empleado maneja sólo una máquina).
  - **D2:** Máquina ®® Producto (cada máquina puede fabricar varios productos y esos productos podrán ser fabricados por todos los empleados que la manejen).
    - Y cada máquina puede ser manejada por varios empleados y esos empleados podrán fabricar todos los productos de dicha máquina: Máquina ®® Empleado.
    - **No** se cumple Empleado ®® Producto: Aunque un empleado puede fabricar varios productos, sólo son aquellos que se fabrican con la misma máquina.
  - **D1** hace que la relación **NO** esté en **FNBC** (Emp. no superllave). Solución: Dividir:
    - **Maneja (**Empleado, Máquina**)** y **Produce (**Máquina, Producto**)** [ambas en 4FN]
- **Si añadimos otra DF:**
  - **D3:** Empleado ® Producto (cada empleado sólo fabrica un producto).
  - Resultado: **Fábrica (**Empleado, Máquina, Producto**)**
    - Por la **Regla de Fusión**, tenemos que Máquina ® Producto (DF, no DMV).
  - **NO** está en **FNBC** (Máquina no superllave)**:** Dividimos en 2 (cambia una llave primaria):
    - **Maneja (**Empleado, Máquina**)** y **Produce (**Máquina, Producto**)** [ambas en 4FN]

Ejemplo y Dependencias Embebidas

- **Fábrica Sin DF:** Un empleado puede usar varias máquinas y fabricar todos los productos que se puedan con dichas máquinas.
  - Se cumple sólo la **DMV:** Máquina ®® Producto (y Máquina ®® Empleado).
  - Tenemos pues, lo siguiente: **Fábrica (**Empleado, Máquina, Producto**)**.
  - **NO** está en **4FN** (sí está en FNBC)**:** Dividimos usando la DMV:
    - **Maneja (**Empleado, Máquina**)** y **Produce (**Máquina, Producto**)**
    - **Ambas DMV no han desaparecido:** Ahora son **DMV triviales.**
    - **Relación Produce:**
      - **Aparece la DMV:** Producto ®® Máquina (que no existía en la relación **Fábrica**).
      - No genera **ningún problema** porque es una **DMV trivial,** pero ¿qué pasaría si al descomponer surgiera una DVM que no fuera trivial? Eso es una **DEPENDENCIA_EMBEBIDA**.
- Sea **R** una relación con dependencias funcionales **F**, y sea **S** una proyección de **R** (o una descomposición) con dependencias **G**:
  - Si **X** ® **Y** está en **G+**, también estará en **F+**: Eso **NO** ocurre con **DMV**.
  - Al descomponer siempre hay alguna DMV que no se transmite completa:
    - Si **X** ®® **Y** se transmitiera completa, no se transmitiría completa su complementaria.

Dependencias Embebidas

- Sea una relación **R,** con un cj. **F** de dependencias (funcionales o no), que se descompone en **S** y **T**. La Reunión **S * T** puede no cumplir **F**.
  - Para dos tablas cualesquiera **S(A,B)** y **T(A,C)**, su reunión **S * T** verifica las DMV: **A** ®® **B** y **A** ®® **C.**
- **Dep. EMBEBIDA:** Es una DMV no trivial, aplicable a una relación de una descomposición (**S**) y no aplicable a la relación original (**R**).
- **Ejemplo: Matricula_Requi (**Alumno, Asignatura, Req, Fecha**);**
  - **Significado: Alumno** matriculado en una **Asignatura**, la cual tiene como Requisito tener aprobada la asignatura **Req**, aprobada el día de **Fecha**.
  - Sólo hay una DF: {Alumno, Req} ® Fecha.
    - La relación no está en **2FN**, porque Fecha depende parcialmente de la llave.
    - NO se cumple la DMV: Asignatura ®® Alumno (aunque una **Asignatura** puede tener varios **Alumno**s matriculados todos estos alumnos no tienen que tener las asignaturas **Req**ueridas aprobadas en las mismas **Fecha**s).
  - Descomposición para **FNBC**:
    - **Aprobadas (**Alumno, Req, Fecha**)**.
    - **Matriculadas_con_Requi (**Alumno, Asignatura, Req**)**.

Dependencias Embebidas: Ejemplo

- La **descomposición** anterior está en **FNBC**:
  - **Aprobadas (**Alumno, Req, Fecha**)**.
  - **Matriculadas_con_Requi (**Alumno, Asignatura, Req**)**.
- **Parece** un diseño correcto, pero **debido al significado** de los atributos, **aparece una DMV:** Asignatura ®® Req **|** Alumno.
  - **Interpretaciones:**
    A)Una Asignatura puede tener varios Requisitos, y todos ellos deben ser cumplidos por todos los alumnos matriculados en esa Asignatura.
    B) Una Asignatura puede tener varios Requisitos y varios Alumnos matriculados en ella, siendo esos Requisitos independientes de esos Alumnos.
  - Esas DMV son dependencias embebidas en la relación original **Matricula_Requi** (no se cumplen en ella), que sí aparecen en la relación **Matriculadas_con_Requi** ⇒ esta relación **NO** está en **4FN**.
  - **Solución:** Dividir **Matriculadas_con_Requi** en dos relaciones, quedando ambas en **4FN.** El esquema completo obtenido es:
    - **Aprobadas (**Alumno, Req, Fecha**)** ⇒ Asignaturas Aprobadas y su Fecha.
    - **Matriculadas (**Alumno, Asignatura**)** ⇒ Alumnos Matriculados y sus Asignat.
    - **Requisitos (**Asignatura, Req**)** ⇒ Requisitos para cada Asignatura.
- **Conclusión:** Al normalizar hay que estar atento por si aparecen Dependencias Multivaluadas no explícitas en la relación original, de acuerdo al **significado** de los atributos.

Dependencias Jerárquicas (DMV Múltiple)

- Si se dice que la DMV **(X ®® Y | Z)** se cumple en **R**, esta DMV será **embebida** si entre **X**, **Y** y **Z** no aparecen todos los atributos de **R**. En otro caso, la DMV será **explícita** (no embebida). Ejemplo:
  - Asignatura ®® Req **|** Alumno, es embebida en **Matricula_Requi** porque falta el atributo Fecha, y es explícita en **Matriculadas_con_Requi.**
- **Dependencia Jerárquica** **X ®® Y_1| Y_2| ...| Y_n**, o **DMV Múltiple:** Si en una relación **R** con los siguientes conjuntos disjuntos de atributos **(X, Y_1, Y_2, ..., Y_n, Z)**, existen asociaciones independientes entre los valores **(X, Y_1), (X,Y_2), ..., (X, Y_n).**
  - **Explícita:** Si **Z=**Æ, es decir, si están incluidos todos los atributos de **R**.
    - Si Z≠Æ, será una dependencia embebida o implícita en **R**, que será explícita en **S**, siendo **S** la proyección de **R** sobre los atributos **(X, Y_1, Y_2, ..., Y_n)**.
  - Existe **(X ®® Y_1| Y_2| ...| Y_n),** si para todo valor **x** del atributo **X** se cumple: $s_{X=x}(S) = p_{X,Y1}(s_{X=x}(S)) * p_{X,Y2}(s_{X=x}(S)) * ... * p_{X,Yn}(s_{X=x}(S))$;
  - **Como una Dep. Jerárquica equivale a varias DMVs, si X no es superllave de R entonces esta relación no estará en 4FN.**
    - **Solución:** Descomponer **R** en las proyecciones: **(X, Y_1), (X,Y_2), ..., (X,Y_n).**

$s_C(R)$: Selección con la condición C
$p_A(R)$: Proyección de los atributos A

Dependencias Jerárquicas: Ejemplo

- La siguiente relación (**S**) en **FNBC** cumple la **Dep. Jerárquica:**
  - Entidad ®® Acción | Cualidad | Herramienta;
    - Si cada **Entidad** debe usar en **TODAS** sus **Acciones**, **TODAS** sus **Cualidades** con **TODAS** sus **Herramientas**.

Entidad	Acción	Cualidad	Herramienta
GreenPeace	Defender Amazonas	Constancia	Reforestación
Plinio el Viejo	Reportaje Vesubio	Curiosidad	Papel
Plinio el Viejo	Reportaje Vesubio	Curiosidad	Lápiz
Chicho Méndez	Defender Amazonas	Amor	Palabra
Chicho Méndez	Defender Amazonas	Sensatez	Reforestación
Chicho Méndez	Defender Amazonas	Sensatez	Palabra
Chicho Méndez	Defender Amazonas	Amor	Reforestación

No está en **4FN** (Entidad no es superllave)

Estas 2 tuplas implican la inserción de las 2 últimas tuplas:

– Por ejemplo, con **X** siendo el atributo **Entidad** y **x = 'Chicho Méndez'**:

$\sigma_{X=x}(S) = \pi_{X,Acción}(\sigma_{X=x}(S)) * \pi_{X,Cualidad}(\sigma_{X=x}(S)) * \pi_{X,Herramienta}(\sigma_{X=x}(S)) =$

$= (\text{Chicho Méndez, Defender Amazonas}) * \begin{pmatrix}\text{Chicho Méndez, Amor}\\ \text{Chicho Méndez, Sensatez}\end{pmatrix} * \begin{pmatrix}\text{Chicho Méndez, Palabra}\\ \text{Chicho Méndez, Reforestación}\end{pmatrix}$

- Si quisiéramos **Insertar** una acción más para Chicho Méndez, habría que insertar **4** tuplas más. Para resolver ese problema: **Dividir en 3 Relaciones**.

Dependencias de Reunión y 5FN

- **Dependencia de Reunión** (yuncional o composicional) en **R:** ***(X1, X2, ..., Xn)** Si la reunión entre las proyecciones sobre los conj. de atributos X_i generan la relación original **R.** Si $R_i=R[X_i]$: **R= R1 * R2 *...* Rn;**
  - **Lógicamente:** $\bigcup_{i=1}^{n} X_i = R$ **;**
  - **Una DMV es una Dependencia de Reunión con n = 2:**
    - ***(X1, X2)** implica que **(R1 ∩ R2) ®® (R1 - R2);** (A la fuerza $R_1 \cap R_2 \neq \emptyset$)
  - **Dependencia de Reunión Trivial:** Si una R_i es igual a **R**.
    - Una Dependencia de Reunión no trivial es **muy rara** en la práctica.
- **Quinta Forma Normal (5FN) o Forma Normal de Proyección-Reunión** con respecto al conj. **F** (de dependencias funcionales, multivaluadas y de reunión)**:** Si en cada dependencia de reunión no trivial de **F+**, denotada por ***(X1, X2, ..., Xn)**, cada X_i es una **superllave de R.**
  - **Ejemplo** de relación no en **5FN: Solidarios(ONG, Proyecto, Lugar)**, suponiendo que siempre que una ***ONG*** tenga un proyecto ***P,*** dicho proyecto ***P*** esté aplicado en el lugar ***L*** y la ***ONG*** tenga al menos un proyecto en dicho Lugar ***L***, entonces, forzosamente, tal ***ONG*** tendrá ***P*** en ***L***. Así, las tuplas de la izda. deben suponer la inserción de la tupla de la dcha.:
    - (WWF/Adena, Árboles, Mundo)
    - (GreenPeace, Árboles, España)
    - (WWF/Adena, Aire, España)

    ⇒ (WWF/Adena, Árboles, España)

Dep. de Reunión: Ejemplo

NOTA

Una dep. **Jerárquica** es una dep. de **Reunión** en la que todas las relaciones de la descomposición (**Ri**), tienen un mismo grupo de atributos comunes (**X**), cumpliendo que: $Y_i = X_i - X$.

- **Solución:** Dividir **R** en $R_1, R_2, ..., R_n$.
  - En el Ej., la Dep. de Reunión ***(X1,X2,X3)**, se divide en las 3 relaciones:
    - **R1** (ONG, Proyecto)
    - **R2** (Proyecto, Lugar)
    - **R3** (ONG, Lugar)

    X2 = {Proyecto, Lugar}
  - Si aplicamos una **Reunión Natural a dos** de esas tres relaciones, se producen **tuplas falsas**, pero si aplicamos una **Reunión Natural a las tres NO.**
- **Ventajas:** Eliminar Problemas al Insertar, Borrar o Modificar.

ONG	Proyecto	Lugar
WWF/Adena	Árboles	Mundo
GreenPeace	Árboles	España
WWF/Adena	Aire	España
WWF/Adena	Árboles	España
WWF/Adena	Biodiversidad	Mundo
Ing. Sin Fronteras	Desarrollo	Mundo

Proyección →
← Reunión

ONG	Proyecto
WWF/Adena	Árboles
GreenPeace	Árboles
WWF/Adena	Aire
WWF/Adena	Biodiversidad
Ing. Sin Fronteras	Desarrollo

Proyecto	Lugar
Árboles	Mundo
Árboles	España
Aire	España
Biodiversidad	Mundo
Desarrollo	Mundo

ONG	Lugar
WWF/Adena	Mundo
GreenPeace	España
WWF/Adena	España
Ing. Sin Fronteras	Mundo

Dependencias de Inclusión

- **Dep. de Inclusión** **R.X < S.Y** **entre los conjuntos de atributos X e Y de las relaciones R y S respectivamente:** Especifica una restricción sobre la proyección en dichos atributos: $\pi_X(R) \subseteq \pi_Y(S)$
  - **X** e **Y** deben ser compatibles respecto la Unión: Tener igual número y tipo de atributos.
  - Expresa una **restricción entre dos relaciones distintas.**
  - Se usan para **especificar:**
    - **Restricciones de integridad referencial (llaves externas). Ejemplos:**
      - **Dpto.**NIFDirector **< Empleado.**NIF**, Empleado.**Dpto **< Dpto.**NumDpto**...**
    - **Relaciones Clase/Subclase a través de la llave común. Ejemplos:**
      - **CENTRAL_SOLAR.**Nombre **< CENTRAL_LIMPIA.**Nombre**, CENTRAL_EOLICA.**Nombre **< CENTRAL_LIMPIA.**Nombre
      - **Empleado.**NIF **< Persona.**NIF**, Estudiante.**NIF **< Persona.**NIF**...**
  - **Reglas de Inferencia:**
    - **1.** Reflexiva: **R.X < R.X**.
    - **2.** Correspondencia de Atributos: **R.{$A_1, A_2, ... A_n$} < S.{$B_1, B_2, ... B_n$}** ⇒ **R.A_i < S.B_i** para **1 ≤ i ≤ n.**
    - **3.** Transitiva: **{R.X < S.Y, S.Y < T.Z}** ⇒ **R.X < T.Z.**