



# Bases de Datos

(Ingeniería Técnica en Informática de Sistemas)

## 3. Diseño de Bases de Datos: Modelo EER y Normalización



E.T.S.I. Informática

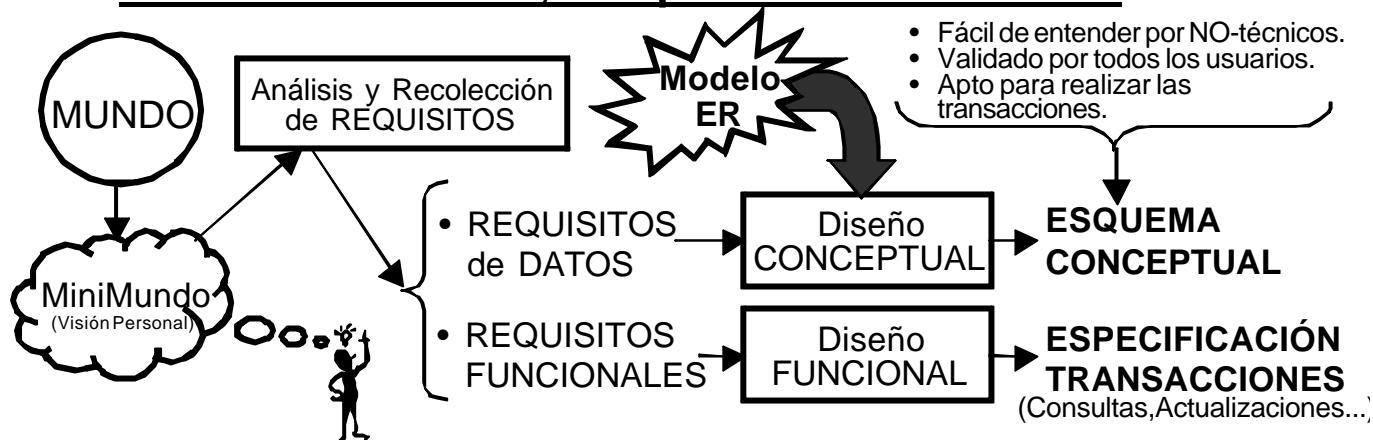
J. Galindo Gómez

### **Modelo ENTIDAD-RELACIÓN: Introducción**

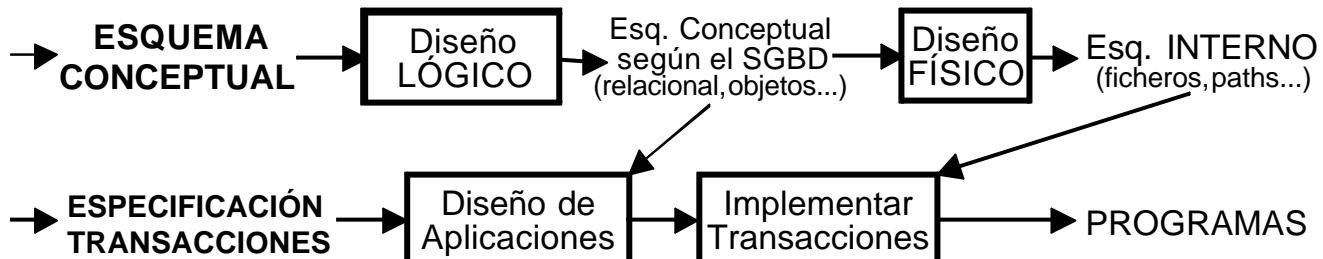
- Es un **modelo conceptual de datos** de alto nivel: Sirve para representar los conceptos del Mundo que nos interesan con sus relaciones y características.
- Es una **herramienta** muy utilizada directamente o a través de otras herramientas o programas (como Data-Architect de Sybase).
- **FASES en el DISEÑO de una BASE de DATOS:**
  - **Fases independientes del SGBD** (Sistema Gestor de Bases de Datos, o DBMS,  *DataBase Management System*), para obtener:
    - Esquema Conceptual: Definición de datos, relaciones...
    - Especificación de funciones necesarias (transacciones).
  - **Fases dependientes del SGBD**, para obtener:
    - Esquema Interno: Ficheros donde almacenar datos, directorios...
    - Programas para efectuar las transacciones.

## Modelo ENTIDAD-RELACIÓN: Introducción

### • FASES del DISEÑO, independientes del SGBD:



### FASES del DISEÑO, dependientes del SGBD:



3

## Modelo ER

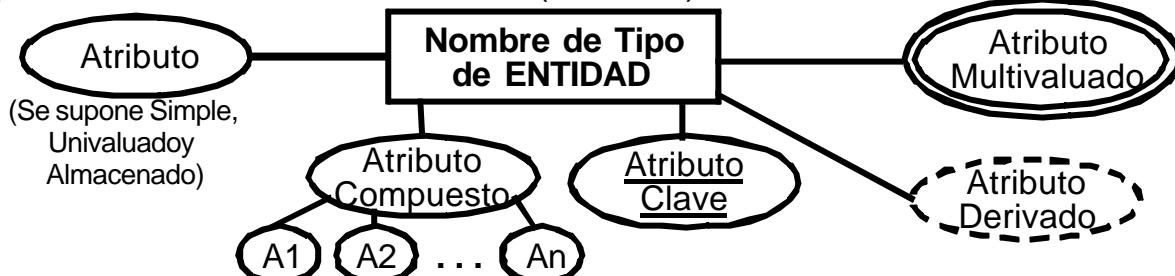
### Conceptos: **ENTIDAD** y **ATRIBUTOS**

- **ENTIDAD:** Concepto, objeto o cosa que existe en el Mundo.
  - **Físicamente:** Ejemplos: Persona, Coche, Cliente...
  - **Lógicamente:** Ejemplos: Empleo, Curso, Compañía...
- **ATRIBUTOS:** Describen las ENTIDADES.
  - Tienen un **DOMINIO**: Conjunto de valores válidos (cadenas, nros...).
  - Valor **NULL**: Ignoramos el valor o el atributo no es aplicable.
  - **Tipos:**
    - **Simples** (indivisibles) y **Compuestos** (divisibles en simples).  
Ejemplo: Dirección=(Calle, Piso, Letra, CP, Ciudad, País).
    - **Univaluados** (Ej: Edad) y **Multivaluados** (Ej: Color en coches, si admitimos que un coche puede tener  $n > 2$ ).
    - **Almacenados** (Ej: Fecha\_Nacimiento) y **Derivados o Calculados** (Ej: Edad).
    - **Complejos:** Son los Compuestos y Multivaluados.

4

## **Conceptos : Tipo de Entidad, Claves...**

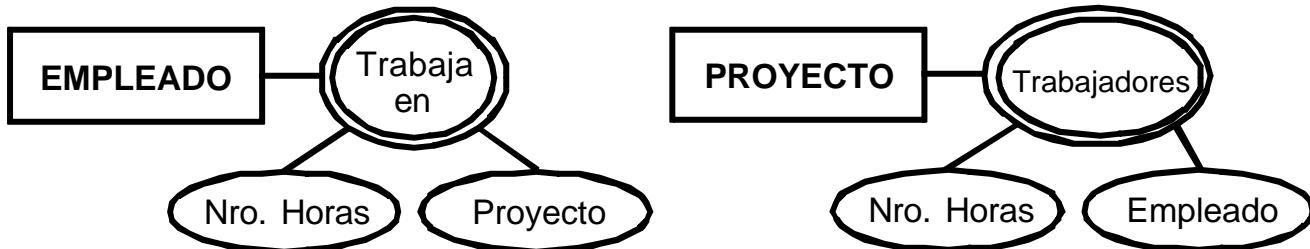
- **TIPO de ENTIDAD**: Conjunto de ENTIDADES con iguales atributos (Ej: Cliente). Al conjunto físico de todas esas entidades (todos los clientes) se le llama **EXTENSIÓN**.
  - **ATRIBUTOS CLAVE o LLAVE (key)**: Aquellos que toman valores únicos y distintos para cada ENTIDAD del mismo tipo.
    - Pueden ser **SIMPLES** o **COMPUESTOS** por varios atributos simples:
      - Si son Compuestos deben ser *mínimos* (sin atributos superfluos o innecesarios).
      - Si tienen atributos superfluos se llamará **SUPERLLAVE**.
      - **Ejemplo:** Atributo DNI será Clave en un tipo de entidad Persona.
  - **INTENSIÓN o ESQUEMA CONCEPTUAL**: Representación de los tipos de entidades, sus atributos (claves...), relaciones entre ellos...



5

## **Conceptos: RELACIONES**

- A veces, existen **distintas formas de representación:**



- Sin embargo, cuando un atributo sirve para relacionar dos entidades, en el **modelo ER** es preferible no representarlo como atributo, sino como **RELACIÓN**.
  - **RELACIÓN:** Relaciona varias entidades (E1, E2... En). Es un subconjunto del producto cartesiano ( $E1 \times E2 \times \dots \times En$ ).
    - Ejemplo de Relación Binaria:



6

## RESTRICCIONES en las RELACIONES

- **RESTRICCIONES ESTRUCTURALES:** De Cardinalidad y de Participación.

- **De Cardinalidad:** Número de Entidades que pueden participar en una relación. En Relaciones Binarias pueden ser de 3 Tipos: **1:1, 1:N y N:M.**
  - Se ponen en los arcos de la Relación.
- **De Participación:** Especifica si la existencia de una entidad depende de estar relacionada con otra o no.
  - **TOTAL** (dependencia de existencia): Si todas las entidades deben relacionarse. Representación: **Doble línea en la Relación:** 
    - **Ejemplo:** Si todo empleado pertenece a un Dpto., no existe ningún empleado sin relacionar con un Dpto. También, todo Dpto. debe tener empleados que trabajen en él:  
 $\forall \text{Empleado } e, \exists \text{Dpto. } d / e \in d,$      $\forall \text{Dpto. } d, \exists \text{Empleado } e / e \in d,$



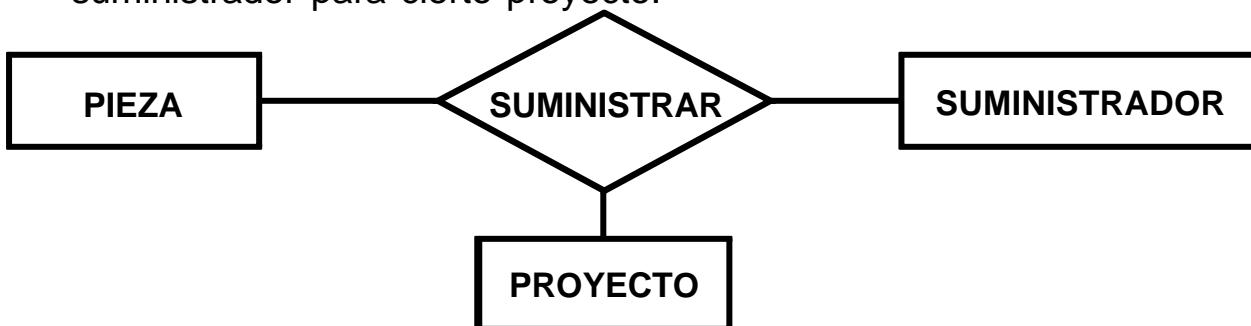
- **PARCIAL:** Si NO todas las entidades tienen que relacionarse.
  - **Ejemplo:** No todo Empleado DIRIGE un Dpto.:  
 $\exists \text{Empleado } e, \exists \text{Dpto. } d / e \in d,$



7

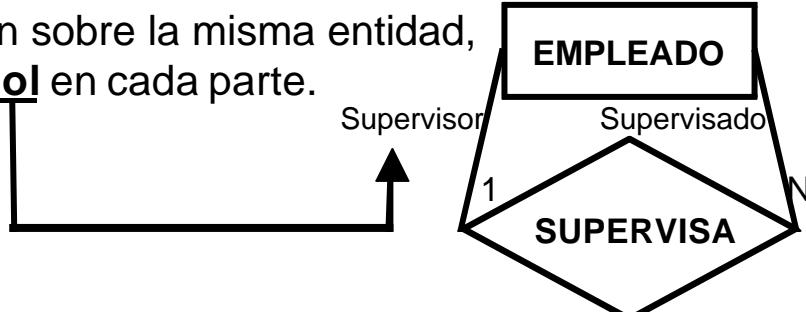
## Tipos de RELACIONES

- **GRADO:** Número de ENTIDADES de la RELACIÓN. Pueden ser Binarias, Ternarias...
  - Ejemplo de Relación Ternaria: Una pieza es suministrada por cierto suministrador para cierto proyecto.



- **RECURSIVA:** Relación sobre la misma entidad, pero jugando distinto **rol** en cada parte.

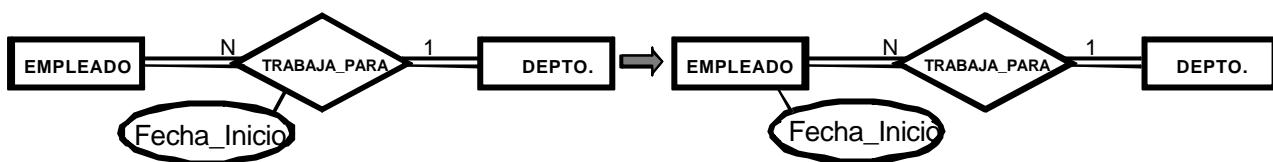
- Ejemplo:



8

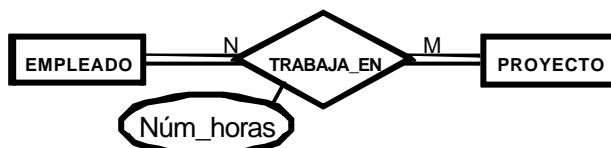
# **ATRIBUTOS en RELACIONES**

- En **Relaciones 1:1 y 1:N**, los Atributos de las Relaciones pueden ponerse también en una entidad.
    - Ejemplo:



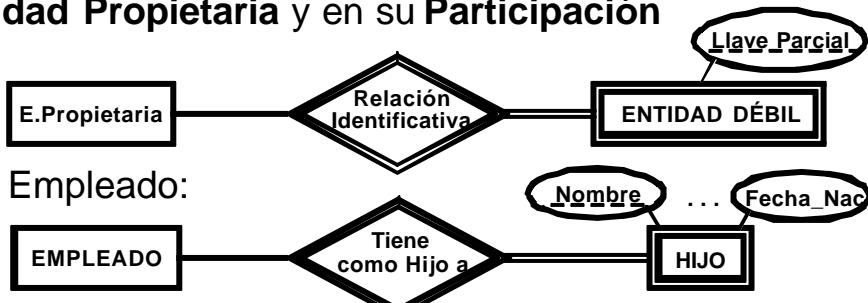
- En **Relaciones N:M** los Atributos dependen de las Entidades que participan en la relación y, por tanto, son forzosamente atributos de la relación y NO pueden ponerse en ninguna de las entidades participantes.

#### - Ejemplo:



## **ENTIDADES DÉBILES (Weak Entities)**

- Son **ENTIDADES SIN LLAVE**: Dependen de otras entidades que las “poseen” (**ENTIDAD PROPIETARIA**) y las identifican inequívocamente.
  - **LLAVE PARCIAL**: Identifica inequívocamente a los elementos de una Entidad Débil, entre otros elementos de la misma **Entidad Propietaria**.
  - Dentro de una **ENTIDAD DÉBIL** pueden existir elementos repetidos, pero no podrán relacionarse con el mismo elemento de la **Entidad Propietaria**.
  - Así, la **LLAVE** de una **ENTIDAD DÉBIL** estará formada por su propia **LLAVE PARCIAL** y la **LLAVE** de la **ENTIDAD PROPIETARIA**.
  - Se representan con **líneas dobles** en la **ENTIDAD DÉBIL**, en la **RELACIÓN** de su **Entidad Propietaria** y en su **Participación** que debe ser **TOTAL**:



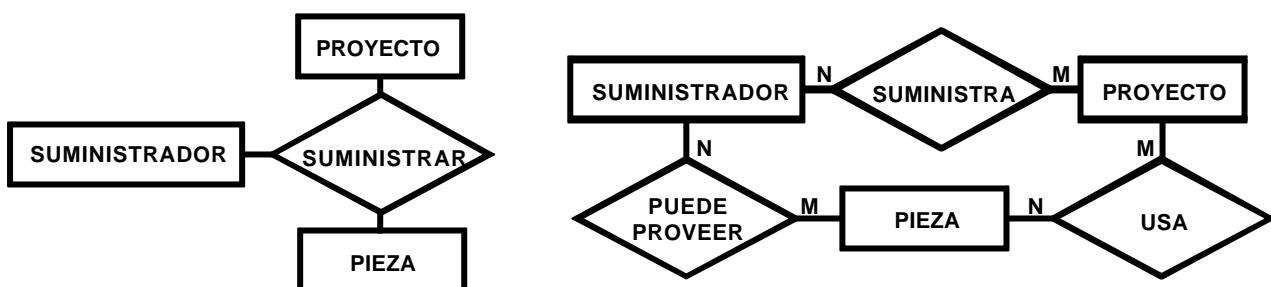
- #### • **Ejemplo:** Hijos de un Empleado:



- También pueden representarse como **ATRIBUTOS COMPLEJOS**.

## RELACIONES con GRADO>2

- Ejemplo:** Relación ternaria y su representación usando 3 relaciones binarias:



- Relación Ternaria:** En general, **representa más información** que 3 binarias.

– **Ejemplo:** El suministrador S provee la pieza P y además suministra al proyecto J. Si además, la pieza P es usada por el proyecto J, eso no significa que la pieza P que usa J sea suministrada por S.

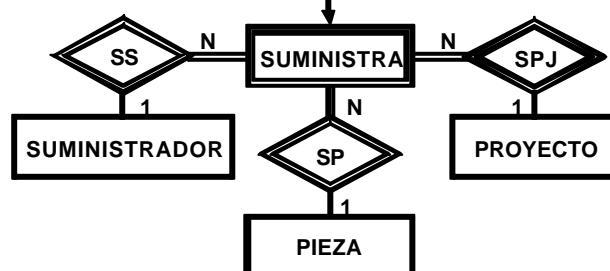
- Solución General:** Incluir la relación ternaria y alguna o algunas de las binarias, según las necesidades. Esto implica algunas restricciones: La existencia de una tupla en la relación ternaria implica tuplas en las binarias, pero no a la inversa.

11

## RELACIONES con GRADO>2

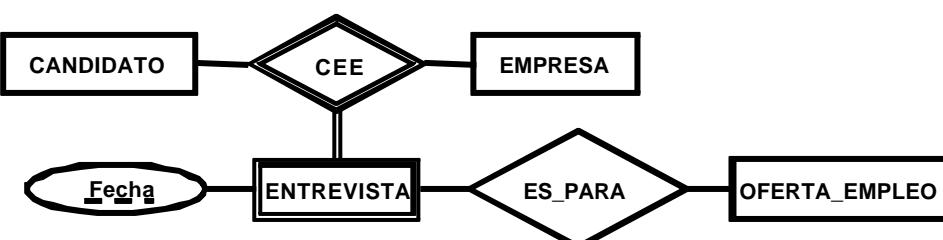
- Si no se permiten relaciones ternarias**, la relación ternaria puede ponerse como **Entidad Débil** sin llave parcial y con tres relaciones que la identifiquen.

– **Ejemplo:**



- También es posible tener una **Entidad Débil** con una relación ternaria que la identifique. En este caso la entidad débil tiene varias entidades propietarias.

– **Ejemplo:**



12

## RELAC. con GRADO>2: Restricciones

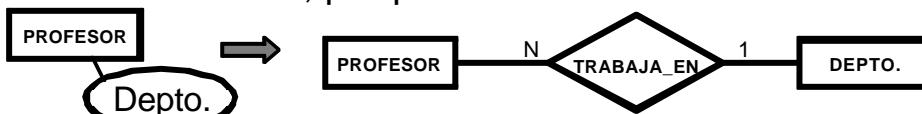
### CARDINALIDAD:

- **Con {1, M , N}:** Se pone 1, M o N en cada arco según la participación de la entidad a la que corresponda.
  - Ejemplo: Si para cada par de valores (proyecto,pieza) sólo puede haber un suministrador, se pone un 1 en el arco de la entidad Suministrador y los otros dos arcos M y N. Esto hace que (proyecto,pieza) sea llave de la relación.
  - Las participaciones que tienen un 1 no serán parte de la llave de la relación.
- **Notación (min,max) Cardinalidad:** Significa que una entidad está relacionada con al menos min y como mucho max instancias en la relación.
  - Aunque en relaciones binarias las restricciones (min,max) pueden determinar la llave de la relación, en relaciones con grado>2 ello no es posible.
  - Sin embargo, especifica un tipo de restricción distinta sobre como muchas instancias de la relación pueden participar en la misma.

13

## OPCIONES de DISEÑO

- A veces es **difícil decidir** si un concepto del “MiniMundo” debe ser una **Entidad**, una **Relación** o un **Atributo**.
- El Esquema debe ser **refinado** en sucesivos pasos, como los siguientes (posteriormente se verán más cuestiones a tener en cuenta):
  - Un concepto puede ponerse como **ATRIBUTO** y luego descubrirse que es una **RELACIÓN**, porque es una referencia a otra **ENTIDAD**:



- Un **ATRIBUTO** de varias **ENTIDADES**, puede convertirse en una **ENTIDAD**:



- Una **ENTIDAD** con pocos atributos y relacionada SÓLO con otra **ENTIDAD**, puede convertirse en un **ATRIBUTO**.
  - **Ejemplo:** Lo inverso del Ej. Anterior, si Dept. sólo interesa de PROFESOR y no tiene muchos atributos.

14

## NOTACIONES ALTERNATIVAS

- **UML** (*Universal Modeling Language*): Modelado Conceptual de Objetos.
- **Notación (min,max)**: Cada **PARTICIPACIÓN** de una Entidad E en una Relación R tiene 2 valores **min** y **max**, con  $0 \leq \text{min} \leq \text{max} \leq 1$ 
  - **min**: Número mínimo de instancias de E que participan en R.
    - $\text{min} = 0$  ® **Participación Parcial**.
    - $\text{min} > 0$  ® **Participación Total**.
  - **max**: Número máximo de instancias de E que participan en R.
    - $\text{max} = 1$  ® **Es una Relación 1:1 o 1:N**.
    - $\text{max} = N$  ® **Es una Relación 1:N o N:M**.
- **Ejemplo de la Notación (min,max): Relación 1:N**
  - Observe como se cambia el orden de **1:N** en los valores max, con esta notación):



15

## RESUMEN de SÍMBOLOS

- **Entidad**
- **Relación**
- **Entidad Débil**
- **Relación identificativa** (de una Entidad Débil)
- **Atributo**
- **Atributo Multivaluado**
- **Atributo Compuesto**
- **Atributo Derivado**
- **Atributo Llave**
- **Atributo Llave Parcial**

### Participación:

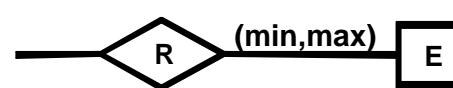
Parcial de E1 en R  
Total de E2 en R



**Cardinalidad 1:N para E1:E2 en R:**

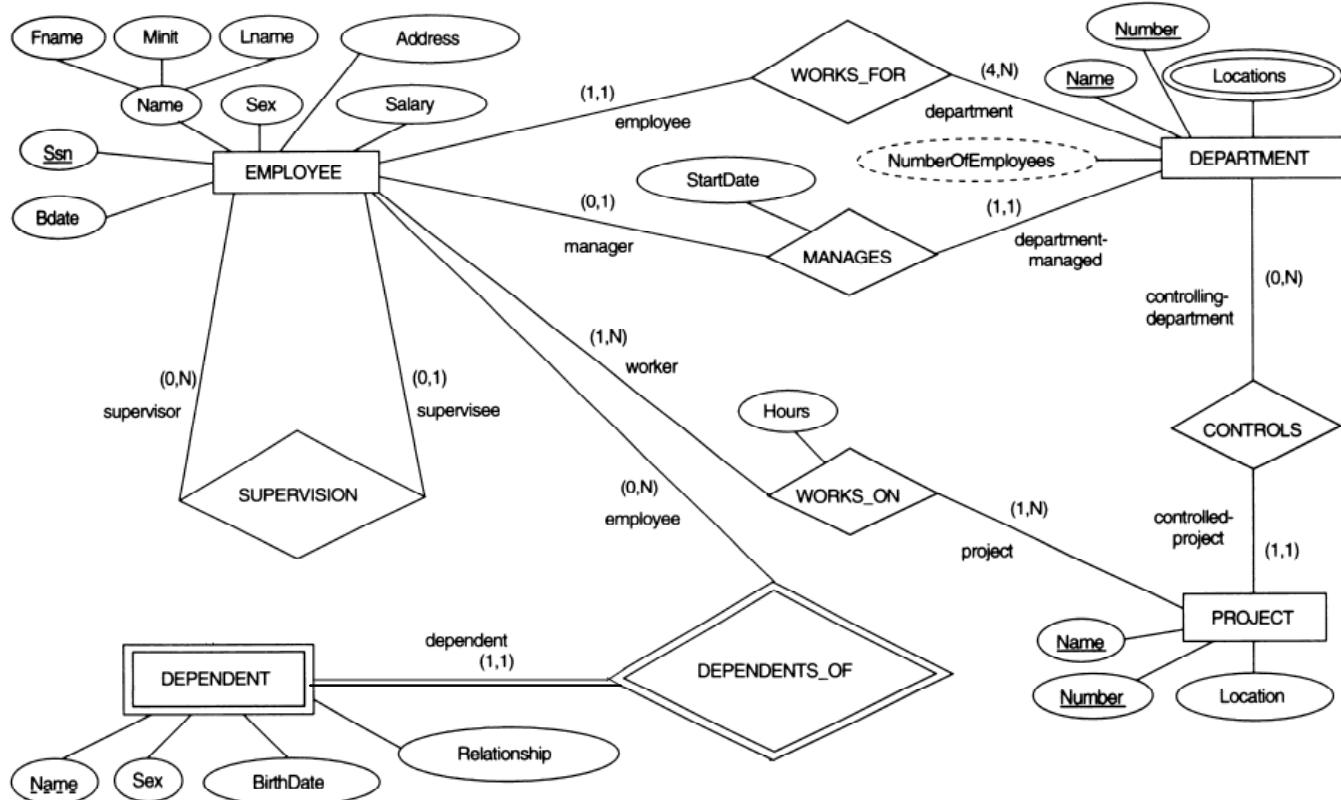


**Notación (min,max) de E en R (Restricción Estructural):**



16

**Ejemplo:** Del libro de Elmasri/Navathe, en inglés, usando la notación (min,max) e incluyendo los roles.



17

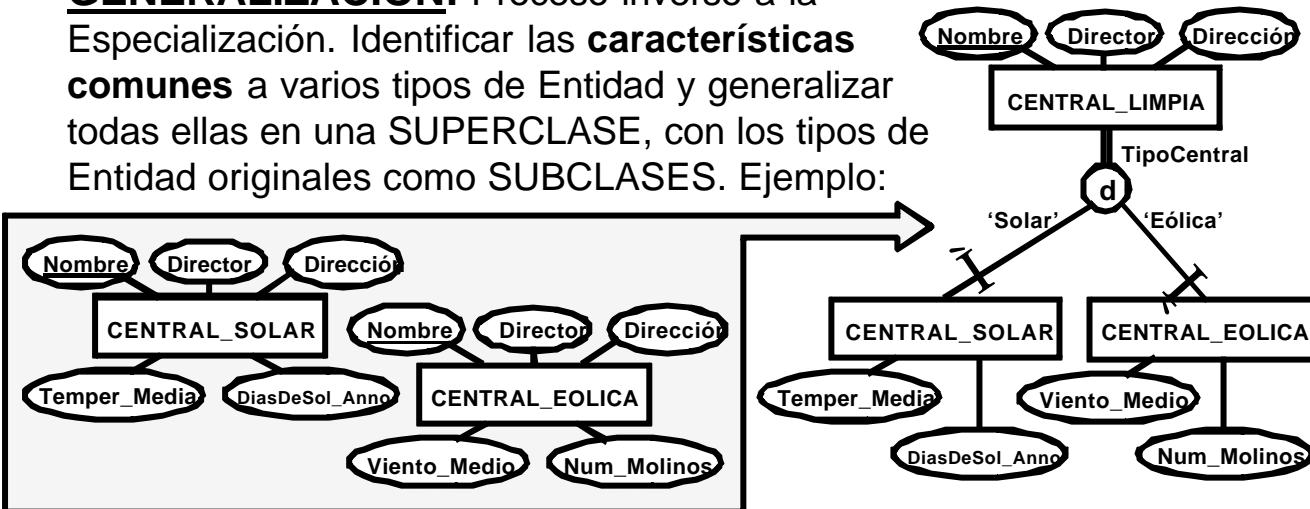
### Modelo ER EXTENDIDO (Extended/Enhanced)

- **SUBCLASE:** Grupo de elementos con algo en común, que pertenecen a una ENTIDAD.
  - **Ejemplo:** Pertenecientes a EMPLEADO, tenemos las subclases INGENIERO, SECRETARIO, SUPERVISOR...
- **SUPERCLASE:** Entidad de la que procede una SUBCLASE.
- **RELACIÓN Clase/Subclase** (o Superclase/Subclase): Es una relación 1:1 en la que ambos elementos son el mismo. Se suele representar por **ES\_UN**. Ejemplo: Ingeniero **ES\_UN** Empleado.
- **CARACTERÍSTICAS:**
  - Una **Entidad** no puede ser sólo miembro de una **SUBCLASE**. Debe ser también miembro de la **SUPERCLASE**.
  - Una **Entidad** puede ser miembro de **varias SUBCLASES**.
    - Ejemplo: Un Empleado puede ser Ingeniero y Supervisor.
  - Una **Entidad** se define por sus **atributos** y sus **relaciones**, los cuales son **HEREDADOS** por sus **SUBCLASES**.
  - Atributos y Relaciones **LOCALES** o **ESPECÍFICAS**: Son aquellas que son propias de una **SUBCLASE** (no de la **SUPERCLASE** a la que pertenece).

18

## ESPECIALIZACIÓN/GENERALIZACIÓN

- **ESPECIALIZACIÓN:** Proceso para definir un conjunto de subclases de un tipo de Entidad (llamada SUPERCLASE).
  - Pueden definirse varias subclases según distintos criterios.
    - Ejemplo: Empleado
      - ® Tipo de Trabajo: Ingeniero, Técnico...
      - ® Tipo de Contrato: Fijo, Por Horas...
  - Deben definirse los atributos y relaciones específicas (si existen).
- **GENERALIZACIÓN:** Proceso inverso a la Especialización. Identificar las **características comunes** a varios tipos de Entidad y generalizar todas ellas en una SUPERCLASE, con los tipos de Entidad originales como SUBCLASES. Ejemplo:



19

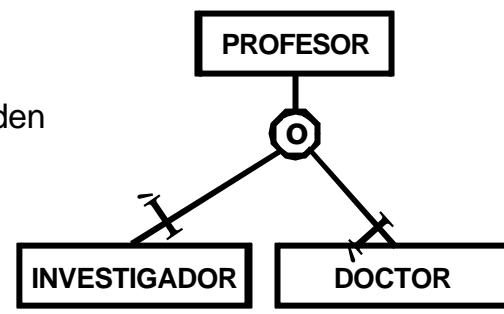
## RESTRICCIONES sobre Espec./Gener.

- En general, una **especialización** tiene **VARIAS** subclases y se representan con la notación del **círculo de especialización**: Como **(d)**
- Pueden también existir especializaciones con **una ÚNICA** subclase a la que **pueden pertenecer o no**, las instancias (o elementos) de la superclase. Ahí, **NO** se usa el círculo.
- **Subclases DEFINIDAS por PREDICADO** (*predicate-defined*) o **por CONDICIÓN** (*condition-defined*): Para determinar si una instancia (o entidad) pertenece a una determinada subclase puede usarse una condición (o predicado) sobre uno o varios atributos de la superclase.
  - **Ejemplo:** La entidad **CENTRAL\_LIMPIA** puede tener el atributo **TipoCentral** con dos valores válidos: **{'Solar', 'Eólica'}**.
  - **RESTRICCIÓN:** Todos los miembros de una subclase **deben** satisfacer la condición establecida en su caso.
    - Ejemplo: Los miembros de la subclase **CENTRAL\_SOLAR**, cumplen el predicado o condición **TipoCentral='Solar'**.
    - Estos predicados se representan escribiendo los valores junto a las líneas de conexión entre superclases y subclases, como en el ejemplo anterior.

20

## RESTRICCIONES sobre Espec./Gener.

- **ESPECIALIZACIÓN DEFINIDA por ATRIBUTO** (*attribute-defined specialization*): Si TODAS las subclases de una especialización tienen la condición de pertenencia, **sobre el mismo atributo** discriminador (como TipoCentral).
- **ESPECIALIZACIÓN DEFINIDA por el USUARIO** (*user-defined specialization*): Si no existe condición para determinar la pertenencia a una subclase y la clasificación se hará individualmente cuando se inserta la entidad.
- **RESTRICCIÓN de DISJUNCIÓN** (*disjointness constraint*):
  - **Espec. DISJUNTA**: Si una entidad puede ser miembro de **una única** subclase en la especialización ® Conjs. disjuntos.
    - Especificación Definida por Atributo (univaluado) ® Espec. Disjunta.
    - Se representa con:
  - **Espec. SOLAPADA** (o coincidente):
    - Si NO son **DISJUNTOS**, las subclases pueden **solaparse o coincidir** (*overlap*) parcial o totalmente: Si una entidad puede pertenecer a varias subclases.
    - Se representa con:



21

## RESTRICCIONES sobre Espec./Gener.

- **RESTRICCIÓN de COMPLETITUD** (*completeness constraint*):
  - **Espec. TOTAL**: Cada entidad de la superclase debe ser miembro de alguna/s subclase/s. Se representa con una doble línea uniendo la superclase con el círculo. Ejemplo: CENTRAL\_LIMPIA.
    - Las superclases que proceden de una generalización suelen ser de este tipo porque la superclase contiene todas las entidades de todas las subclases.
  - **Espec. PARCIAL**: Una entidad puede no pertenecer a ninguna de sus subclases. Se representa con una línea simple. Ejemplo: Un Profesor puede ser Investigador, Doctor, ambas cosas o ninguna.
- **RESUMEN de TIPOS de RESTRICCIONES**: Las restricciones de disjunción y de completitud son **independientes**, por lo que existen **4 tipos** de posibles restricciones sobre Especialización:
  - **Disjunta, total** (*disjoint, total*).
  - **Disjunta, parcial** (*disjoint, partial*).
  - **Solapada, total** (*overlapping, total*).
  - **Solapada, parcial** (*overlapping, partial*).

22

## RESTRICCIONES sobre Espec./Gener.

- **REGLAS de INSERCIÓN y BORRADO:** (E=Entidad, S=Superclase)

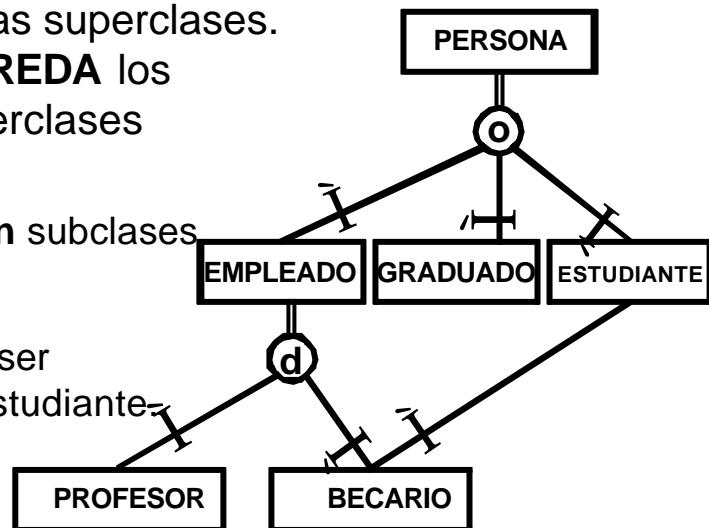
– Borrar entidad $E \in S$	®	Borrar E de las subclases de S.
– Borrar E $\in$ subclase	®	Borrar E de la superclase a la que pertenezca (si procede, ya que puede ser una especialización parcial).
– Insertar E en una S	®	Insertar E en las subclases de S definidas por predicado que cumplan dicho predicado.
– Insertar E en S de Esp. <b>Disjunta-Total</b>	®	Insertar E en <u>una</u> única subclase.
– Insertar E en S de Esp. <b>Disjunta-Parcial</b>	®	Insertar E en <u>una</u> o <u>ninguna</u> subclase.
– Insertar E en S de Esp. <b>Solapada-Total</b>	®	Insertar E en <u>una</u> o <u>más</u> subclases.
– Insertar E en S de Esp. <b>Solapada-Parcial</b>	®	Insertar E en <u>cero</u> , <u>una</u> o <u>más</u> subclases.

23

## JERARQUÍAS y RETÍCULOS

- Una subclase puede tener, a su vez, otras subclases formando así una **Jerarquía** (*hierarchy*) o un **Retículo** (*lattice*).
- **Espec./Generalización JERÁRQUICA:** Tiene la restricción de que todas las subclases pertenecen sólo a una superclase.
- **Espec./Generalización en RETÍCULO (malla o red):** Una subclase puede serlo de varias superclases. En ese caso, la subclase **HEREDA** los atributos de **TODAS** sus superclases (por todos los caminos).
- **NODO HOJA** (*leaf*): Entidad **sin** subclases
- **Ejemplo:**

- Una persona puede y debe ser Empleado, Graduado **y/o** Estudiante.
- Un Empleado puede y debe ser Profesor **o** Becario.
- Un Becario es **forzosamente** también un Estudiante.



24

## JERARQUÍAS y RETÍCULOS

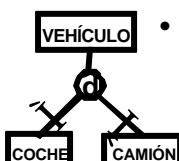
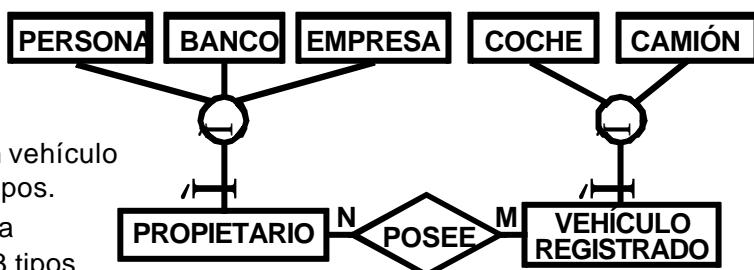
- **SUBCLASE COMPARTIDA** (*shared subclass*): Subclase con más de una superclase. Si existe al menos una subclase compartida, tenemos un **RETÍCULO** y si no existe ninguna tenemos una **JERARQUÍA**.
- **HERENCIA MÚLTIPLE** (*multiple inheritance*): Cuando las subclases compartidas heredan atributos y relaciones de varias clases.
  - Si una subclase compartida hereda características de una superclase por varias vías distintas (en un retículo), dichas características se considerarán sólo una vez.
  - A veces, no se permite herencia múltiple (ni clases compartidas).
    - Solución: Crear subclases para cubrir todas las combinaciones posibles de las subclases de una Especialización Solapada.
    - Ejemplo: Subclase Becario (o Empleado\_Alumno) de la superclase Persona.
- **MODELADO de DATOS CONCEPTUAL:**
  - Proceso de Refinamiento Conceptual **DESCENDENTE** (*top-down*): Sistema de Diseño Conceptual que parte de las entidades básicas y aplica la especialización para ir consiguiendo sucesivamente subclases más específicas. Se obtienen primero diseños Jerárquicos que pueden posteriormente convertirse en Retículos.
  - Síntesis Conceptual **ASCENDENTE** (*bottom-up*): Aplica más la generalización que la especialización, partiendo inicialmente de las entidades más especializadas.
  - En la práctica suele usarse un **Sistema de Diseño MIXTO**.

25

## TIPOS UNION o CATEGORIAS:

- **TIPO UNIÓN o CATEGORIA** (*union type or category*): Subclase que representa una colección de objetos, que son un subconjunto de la **UNION** de distintos tipos de entidad.
  - **Ejemplo:** Sup. 3 entidades Persona, Banco y Empresa.
    - La categoría Propietario de un vehículo incluirá elementos de esos 3 tipos.
    - La categoría Propietario es una subclase de la UNIÓN de los 3 tipos.
    - También hemos creado la categoría VehículoRegistrado a la que **no** tienen porque pertenecer **todos** los vehículos (puede haber vehículos no registrados). Eso no ocurriría si se modela siendo {Coche,Camión} una especialización disjunta de Vehículo. Si esa especialización disjunta es PARCIAL indicaría que un Vehículo puede ser de una subclase distinta a {Coche,Camión}. Sin embargo, como Categoría eso no es posible.
  - Una **CATEGORÍA** siempre tiene **dos o más superclases** (que son distintos tipos de entidad). Una Relación superclase/subclase sólo tiene una única superclase.
  - Una **CATEGORÍA** es similar a una subclase compartida pero:
    - Una **SUBCLASE Compartida** debe pertenecer a **TODAS** sus superclases y **hereda** los atributos de **TODAS** ellas: Es un subconjunto de la **INTERSECCIÓN** de las superclases.
    - Una **CATEGORÍA** es un subconjunto de la **UNIÓN disjunta de varias superclases**: Los miembros de una Categoría deben pertenecer **A UNA** de las superclases (no a todas) y **heredan sólo** los atributos de la superclase a la que pertenezcan.

26

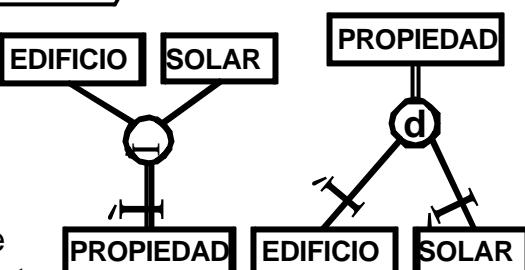


## TIPOS UNION o CATEGORIAS:

- **PARTICIPACIÓN en una CATEGORIA:**

- **TOTAL:** Si todas las superclases de la Categoría deben ser miembros de la Categoría.

- Una Categoría **TOTAL** también puede modelarse como Generalización Disjunta, lo cual es preferible si las entidades tienen muchos atributos/relaciones comunes. Ejemplo: Un Edificio o Solar siempre debe ser una Propiedad.



- **PARCIAL:** Si no todas las superclases deben ser miembros de la Categoría. Ejemplo: No toda Persona tiene que ser Propietaria (de un Vehículo Registrado).

- Así pues, en una **CATEGORIZACIÓN**, la subclase o **Categoría**, debe pertenecer siempre a **UNA y SOLO UNA** de las superclases, pero las superclases no tienen que pertenecer a la Categoría.

- Si las superclases deben pertenecer a la categoría, tenemos una **Categoría TOTAL** y se puede representar también como una Generalización disjunta.
- Recordemos que en toda Generalización todos los miembros de las subclases deben ser también miembros de la superclase. Al revés sólo se cumple si es **TOTAL** (y no se cumple si es **PARCIAL**).

27

## RESUMEN de CARACTERÍSTICAS

- **Clase:** Conjunto de Entidades.
- **Subclase:** Clase cuyas entidades miembros deben siempre ser un subconjunto de otra clase, la Superclase: **subclase ES\_UN/A superclase**.
  - **Definida por predicado**, si existe una condición para determinar los miembros de la superclase que pertenecen a la subclase. Si la condición usa un único atributo con el comparador de igualdad se llama **definida por atributo** (que normalmente será disjunta porque las condiciones serán distintas para cada subclase). Será **definida por usuario**, si no ocurre lo anterior.
- **Especialización/Generalización:** Conjunto de subclases que tienen la misma superclase.
  - **Restricción de COMPLETITUD: TOTAL**, si la unión de todas las subclases genera la superclase. O, lo que es lo mismo, si cada miembro de la superclase debe pertenecer al menos a una subclase. **PARCIAL**, si no ocurre lo anterior.
  - **Restricción de DISJUNCIÓN: DISJUNTA**, si la intersección de dos subclases siempre genera el conjunto vacío. O, lo que es lo mismo, una superclase no puede pertenecer a varias subclases. **SOLAPADA**, si no ocurre lo anterior.
- **Categoría:** Clase, subconjunto de la UNIÓN de n (con n>1) superclases disjuntas ® Cada miembro de una categoría debe pertenecer a sólo una de las superclases.

28

## CORRESPONDENCIA ER/EER-BDR

- Una vez diseñado el **ESQUEMA CONCEPTUAL** usando el **Modelo EER**, debemos traducirlo al **Esquema de BDR**.
  - Muchos sistemas **CASE** (*ComputerAided Software Engineering*) permiten diseñar un **modelo ER/EER** y, automáticamente, obtener las instrucciones en el **lenguaje DDL** del SGBD usado (normalmente **SQL**).
- **Traducir un Modelo ER en un Esquema de BDR:** Algoritmo en 7 pasos, para la **Obtención de las Tablas**:
  - 1. Convertir **cada Entidad en una Tabla** con sus atributos:
    - Los **atributos compuestos** se descomponen en simples.
    - Escoger la **Llave primaria** de entre las llaves candidatas.
  - 2. Igualmente, **cada Entidad Débil será una Tabla**:
    - Se incluirán como atributos las llaves primarias de las **entidades propietarias** (como mínimo existirá una).
    - La **Llave primaria** estará formada por la llave parcial y los atributos anteriores (que también actuarán como llave externa referenciando los entes de la entidad propietaria).

29

## TRADUCCIÓN ER-BDR: Algoritmo

- 3. Para **cada Relación R binaria 1:1** que relacione S y T:
  - Incluir la **Llave primaria de una relación** (por ejemplo la de T) **como atributos (Llave externa)** en la **otra relación** (la de S).
  - Es mejor que S tenga una **participación TOTAL** en R (para evitar valores NULL).
    - **Ejemplo:** Todo Departamento tiene que tener un director.
  - Incluir también en la tabla de S, los **atributos de la relación R**.
- 4. Para **cada Relación R binaria 1:N** que relacione S y T, siendo S la relación del lado de N y T la relación del lado de 1, i.e. cada instancia de S se relaciona como mucho con una de T:
  - Incluir como **Llave externa** de la relación S la **Llave primaria** de T.
  - Incluir también en la tabla de S, los **atributos de la relación R**.
- 5. Para **cada Relación R binaria N:M** que relacione S y T:
  - Crear una **nueva tabla** con los atributos que son llaves primarias de S y T. Juntos serán la nueva llave primaria y separados serán sendas llaves externas.
  - Incluir en la **nueva tabla**, los **atributos de la relación R**.

30

## TRADUCCIÓN ER-BDR: Algoritmo

- Las **relaciones 1:1 y 1:N**, pueden traducirse como las relaciones **N:M** : Esto es preferible cuando hay pocos elementos en la relación R, para evitar valores NULL en las llaves externas. Si R es **1:N** la llave primaria será la del lado de **N** y si es **1:1** será la de participación total (si la hay).
- Para obtener otros atributos, a partir de las llaves externas, se puede usar la operación de **REUNIÓN NATURAL**.

### – 6. Para cada atributo multivaluado de E:

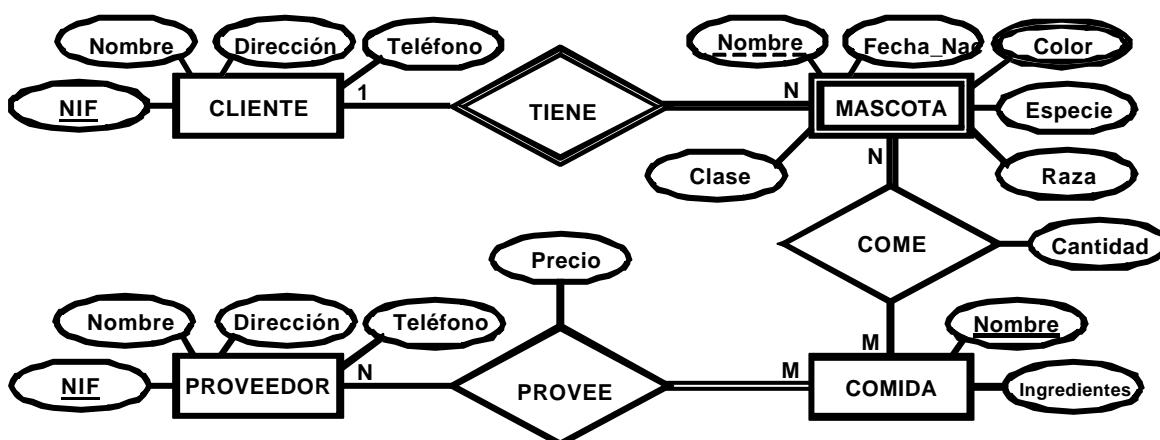
- Crear una **nueva tabla** con dicho atributo y con la **llave primaria** de la entidad a la que pertenece, que será llave externa.
- La **llave primaria** de la nueva tabla serán todos sus atributos.

### – 7. Para cada relación R n-aria (de grado n>2):

- Crear una **nueva tabla** con los atributos que son llaves primarias de las entidades que relaciona R.
  - Eos atributos serán llaves externas a sus respectivas entidades.
- Esas llaves externas formarán su **llave primaria**.
  - Si la cardinalidad de alguna entidad es 1 en R, puede quitarse su llave primaria de la **llave primaria de la nueva tabla** (no de la tabla).
- Incluir en la **nueva tabla**, los **atributos de la relación R**.

31

## EJEMPLO: Hotel de Mascotas



**Cliente** (NIF, Nombre, Dirección, Teléfono);

**Mascota** (NIFPropietario, Nombre, Fecha\_Nac, Especie, Raza, Clase);

**Comida** (Nombre, Ingredientes);

**Proveedor** (NIF, Nombre, Dirección, Teléfono);

**Come** (NIFPropietario, NombreMascota, NombreComida, Cantidad);

**Provee** (NIFProveedor, NombreComida, Precio);

**ColorMascota** (NIFPropietario, Nombre, Color);

32

## TRADUCCIÓN EER-BDR

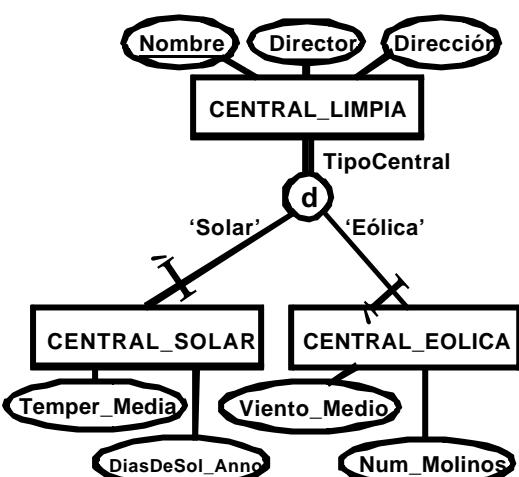
### • Traducir un Modelo EER en un Esquema de BDR:

- Es una extensión del Algoritmo anterior (**EER® BDR**), al que se le añade un paso más (paso 8), para la conversión en tablas de superclases/subclases.
  - Para ello, hay distintas opciones, de las que comentaremos sólo 4.
- **Notación:**
  - **Attrs(R)**: Atributos de la relación (o clase) R (*Attributes*).
  - **PK(R)**: Llave primaria de R (*Primary Key*).
- Sea **C** una superclase con atributos **Attrs(C)= $\{k, a_1, a_2, \dots, a_n\}$**  y con **m** subclases  $\{S_1, S_2, \dots, S_m\}$ :
  - 8. Convertir **Cada Especialización** en esquemas relacionales siguiendo una de las siguientes **4 opciones**, las cuales necesitan crear un número distinto de relaciones en la BDR:
    - **OPCIÓN 8A:  $m+1$  tablas.**
    - **OPCIÓN 8B:  $m$  tablas.**
    - **OPCIÓN 8C: 1 tabla.**
    - **OPCIÓN 8D: 1 tabla.**

33

## TRADUCCIÓN EER-BDR

- **OPCIÓN 8A:** Para cualquier tipo de Especialización:
  - » Crear una tabla **L** para **C**, con sus atributos.
  - » Crear una tabla **Li** para cada **subclase Si**, con  $1 \leq i \leq m$ , tal que: **Attrs(Li) = {k} È Attrs(Si)**. Además, **PK(Li) = k**.
- **OPCIÓN 8B:** Para una Especialización **Disjunta-Total**:
  - » Crear una tabla **Li** para cada **subclase Si**, con  $1 \leq i \leq m$ , tal que: **Attrs(Li) = Attrs(C) È Attrs(Si)**. Además, **PK(Li) = k**.
- **OPCIÓN 8C:** Para una Especialización **Disjunta** definida por el **Atributo t** (si no es definida por Atributo, se crea el atributo **t**):
  - » Crear una única tabla **L** con: **PK(L) = k**.
  - » **Attrs(L) = Attrs(C) È Attrs(S<sub>1</sub>) È ... È Attrs(S<sub>m</sub>) È {t}**.
- **OPCIÓN 8D:** Para Especializaciones **Solapadas** (o **Disjuntas**). Supondremos **m** **atributos lógicos (booleanos) ti**, con  $1 \leq i \leq m$ , tal que cada **ti** indica si la tupla pertenece o no a la subclase **Si**.
  - » Crear una única tabla **L** con: **PK(L) = k**.
  - » **Attrs(L) = Attrs(C) È Attrs(S<sub>1</sub>) È ... È Attrs(S<sub>m</sub>) È {t<sub>1</sub>, ..., t<sub>m</sub>}**.

**EJEMPLO: Energía Limpia** (m=2, k=Nombre)• **OPCIÓN 8A:**

- **Central\_Limpia**(Nombre, Director, Dirección, TipoCentral); → (TipoCentral no es necesario).
- **Central\_Solar**(Nombre, Temper\_Media, DiasDeSol\_Aanno);
- **Central\_Eolica**(Nombre, Viento\_Medio, Num\_Molinos);

• **OPCIÓN 8B:**

- **Central\_Solar**(Nombre, Director, Dirección, Temper\_Media, DiasDeSol\_Aanno);
- **Central\_Eolica**(Nombre, Director, Dirección, Viento\_Medio, Num\_Molinos);

• **OPCIÓN 8C:**

- **Central\_Limpia**(Nombre, Director, Dirección, TipoCentral, Temper\_Media, DiasDeSol\_Aanno, Viento\_Medio, Num\_Molinos); → {t}

• **OPCIÓN 8D:** Si “fuerá” una Especialización Solapada:

- **Central\_Limpia**(Nombre, Director, Dirección, Temper\_Media, DiasDeSol\_Aanno, Viento\_Medio, Num\_Molinos, TCSolar, TCEolica);

{t<sub>1</sub>, t<sub>2</sub>}

35

**EER ® BDR: Observaciones**• **OPCIÓN 8A:** Para cualquier tipo de especialización: **Disjunta o Solapada, Total o Parcial.**

- Cada tabla **Li** de una subclase cumple que:  $p_k(Li) \sqsubseteq p_k(L)$ ;
- Con una **Equi-Reunión** entre **L** y **Li**, se obtienen todos los atributos (específicos y heredados) de cada entidad de la **i**-ésima subclase.
- Si es **definida por atributo**, ese atributo no es necesario en **L**.

• **OPCIÓN 8B:** Para una Especialización **Disjunta-Total**.

- Si no fuera **Total**, la entidad que no fuera de ninguna subclase se perdería.
- Si no fuera **Disjunta**, la entidad que perteneciera a varias subclases tendría sus atributos heredados **redundantes** ® Grave problema.
- Si es **definida por atributo**, ese atributo no se incluye en las **Li**.

• **OPCIÓN 8C (8D):** Usa el Atributo **t** (o **m** atributos **ti**).

- Si es **Parcial**, el atributo **t** (o todos los atributos **ti**) podrá tomar el valor **NULL**.
- Cada fila tendrá a **NULL** los atributos que no le correspondan.
- **No recomendables** si existen muchos atributos específicos en las subclases. En otro caso, son mejores que **8A** y **8B**, pues evitan realizar Equi-Reuniones.

Por supuesto, en un mismo **Modelo EER** pueden usarse distintas opciones.

36

- **Subclase Compartida:**

- Es una subclase de varias superclases: Deben tener **los mismos atributos Llave**. Si no ocurre así, la subclase compartida debería ser modelada como **Categoría**.
- Puede usarse cualquier opción del **paso 8**: Suele usarse **8A**.

- **Categorías:**

- Es una subclase de la unión de 2 ó más superclases, que pueden tener **distintas Llaves** (porque pueden ser de distintos tipos de entidad).
  - **Ejemplo:** Propietario ∈ {Persona,Banco,Empresa}.
- **Si tienen distintas llaves**, hay que hacer lo siguiente:
  - **Crear una LLAVE SUPLENTE (surrogate key)**: Nueva **Llave primaria** para la tabla de la Categoría. En esa tabla se incluirán todos los atributos de la Categoría.
  - Incluir la **Llave Suplente como Llave externa de las superclases**, para especificar la correspondencia entre la llave suplente y la llave de cada superclase.

37

## 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.

38

## 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.

39

## Dependencias Funcionales

- Sea un esquema  $R = \{A_1, A_2, \dots, A_n\}$ , con  $X$  e  $Y$  siendo dos conjuntos de tales atributos.
- La Dependencia Funcional (DF)  $X \xrightarrow{\circ} Y$  especifica una restricción en  $R$ :  
$$X \xrightarrow{\circ} Y \text{ indica que " } t, u \in R: t[X] = u[X] \models t[Y] = u[Y]$$

- **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  $X \xrightarrow{\circ} Y$ , para cualquier  $Y$ .
- $X \xrightarrow{\circ} Y$ , no implica  $Y \xrightarrow{\circ} X$ .
- 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  $Y \subseteq X$ , entonces  $X \xrightarrow{\circ} Y$ .
- 2. Regla de Aumentación:  $\{X \xrightarrow{\circ} Y\} \models \{XZ \xrightarrow{\circ} YZ\}$ .
- 3. Regla Transitiva:  $\{X \xrightarrow{\circ} Y, Y \xrightarrow{\circ} Z\} \models \{X \xrightarrow{\circ} Z\}$ .
- 4. Regla de Descomposición:  $\{X \xrightarrow{\circ} YZ\} \models \{X \xrightarrow{\circ} Y, X \xrightarrow{\circ} Z\}$  (Aplicar Reglas 1 y 3).
- 5. Regla de Unión:  $\{X \xrightarrow{\circ} Y, X \xrightarrow{\circ} Z\} \models \{X \xrightarrow{\circ} YZ\}$  (R. 2 y 6).
- 6. Regla Pseudotransitiva:  $\{X \xrightarrow{\circ} Y, WY \xrightarrow{\circ} Z\} \models \{WX \xrightarrow{\circ} Z\}$  (R. 2 y 3).

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

40

## 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 Í 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  $X®Y$  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  $X®A$  no puede reemplazarse por  $Y®A$ , donde  $Y \not\subseteq X$ , siendo, con ese cambio, equivalente a  $F$ .

41

## Normalización

- Es un proceso para conseguir que un esquema relacional cumpla ciertas condiciones que eliminan ciertos problemas.
- Existen distintos niveles o **Formas Normales**:
  - Existen tres **Formas Normales** básicas (Codd, 1972): **1<sup>a</sup>, 2<sup>a</sup> y 3<sup>a</sup> FN**.
  - Posteriormente, se propuso la tercera **Forma Normal de Boyce-Codd**, más restrictiva que la 3<sup>a</sup> FN. Todas se basan en las DF existentes.
  - La **4<sup>a</sup> y 5<sup>a</sup> 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.

42

## 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**.

43

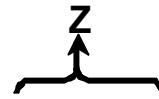
## Formas Normales Básicas: 2FN

- **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  $\{K - B\} \rightarrow A$ .
  - **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);

44

## Formas Normales Básicas: 3FN

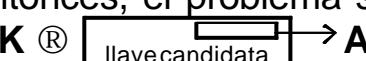
- **Tercera Forma Normal (3FN):** No deben existir dependencias **transitivas** de atributos no llave, respecto de la llave primaria: Atributos no llave no deben depender funcionalmente de otros atributos no llave.
  - **Dependencia Transitiva X®Y:** Si existe un conjunto de atributos Z, que no son llave 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).



45

## 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.
- **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 df. 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.



46

## **FNBC: Forma Normal de Boyce-Codd**

- **FNBC:** Si en cualquier dependencia no trivial  $X \xrightarrow{*} A$  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:  
 $R(\underline{A}, \underline{B}, C)$ , en la que ocurre que  $\{A, B\} \xrightarrow{*} C$  y además,  $C \xrightarrow{*} B$ .
  - **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  $(AB \text{ y } BC)$  o  $(AB \text{ y } AC)$ , 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  $X \xrightarrow{*} A$ , 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)**.

47

## **FNBC: Ejemplo**

- **Relación en 3FN: Curso (Estudiante, Asignatura, Profesor).**
- **DF:**  $\{\text{Estudiante, Asignatura}\} \xrightarrow{*} \text{Profesor}$ ,  $\text{Profesor} \xrightarrow{*} \text{Asignatura}$ .
  - Si un profesor puede impartir varias asignaturas, no se cumplirá la dependencia  $\text{Profesor} \xrightarrow{*} \text{Asignatura}$ .
- **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):
    - Tuplas falsas: (Santiago, B.D., Paco), (Jesús, B.D., Paco) y (María, B.D., Pepe).
  - (Estudiante, Asignatura) y (Estudiante, Profesor):
    - Tuplas falsas: (Jesús, S.O., Pepe) y (Jesús, B.D., Juan).

48

## FN de una Relación ya en 1FN

- **Analizar TODAS las DFs de la relación con la forma  $X \xrightarrow{R} A$**  : 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 È Llave)	SI	3FN (No FNBC)
NO (X Ì Llave)	NO	1FN (No 2FN)
NO (X È Llave)	NO	2FN (No 3FN)

49

## Características de un Buen Diseño

- Hemos visto que, para evitar determinados problemas, el esquema de una BD debe **dividirse o descomponerse** en  $m$  tablas  $R_i$ ,  $i=1,...,m$ , (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  $R_i$ , 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 (el algoritmo de descomposición no garantiza la propiedad de reunión sin pérdidas).

50

## 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 conj. 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  $F^+$  :
    - $(R_1 \cap R_2) \xrightarrow{\circ} (R_1 - R_2)$ , o bien  $(R_1 \cap R_2) \xrightarrow{\circ} (R_2 - R_1)$ . 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.

51

## 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:  $x \xrightarrow{\circ} A_1, x \xrightarrow{\circ} A_2, \dots, x \xrightarrow{\circ} A_k$ ;
  - Crear una relación con llave  $x$  y atributos  $\{x, A_1, A_2, \dots, A_k\}$ ;
- 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.

52

## Dep. Multivaluadas (DMV) o Plurales

- **DMV  $X \bowtie Y$  en R, con  $X, Y \subset R$  ( $X$  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  $Z = R - (X \cup Y)$ :

- $t_1[X] = t_2[X] = t_3[X] = t_4[X]$
- $t_1[Y] = t_3[Y]$  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,  $X \bowtie Y$ , implica que también se da  $X \bowtie Z$ :  $X \bowtie Y | Z$
- **Interpretaciones de  $X \bowtie Y$ :**

- 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  $X$ .
- 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  $Y \subset X$ , o si  $R = X \cup Y$  ( $Z = \emptyset$ ).

- 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).

53

## DMV: Ejemplo y Reglas de Inferencia

- **Ejemplo: Empleado(Nombre, Proyecto, Hijo),**

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

Nombre	Proyecto	Hijo
Príamo	Caballo	Cassandra
Príamo	Troya	Paris
Príamo	Caballo	Paris
Príamo	Troya	Cassandra

- **Reglas de Inferencia:** A las 3 reglas de Inferencia de Armstrong podemos añadir otras sobre DMV:

- 4. Regla de Complementación de DMV:  $\{X \bowtie Y\} \vdash \{X \bowtie (R - (X \cup Y))\}$ .
- 5. Regla de Aumentación para DMV:  $\{X \bowtie Y, Z \subset W\} \vdash \{WX \bowtie YZ\}$ .
- 6. Regla Transitiva para DMV:  $\{X \bowtie Y, Y \bowtie Z\} \vdash \{X \bowtie (Z - Y)\}$ .
- 7. Regla de Paso de DF a DMV:  $\{X \bowtie Y\} \vdash \{X \bowtie Y\}$  (implica que **FNBC**  $\vdash$  **4FN**).
- 8. Regla de Fusión de DF y DMV:  $\{X \bowtie Y, W \cap Y = \emptyset, W \bowtie Z, Z \subset Y\} \vdash \{X \bowtie Z\}$ .
  - Ejemplo: {Nombre  $\rightarrow\!\!\!-\!\!\!$  Proyecto, Hijo  $\rightarrow\!\!\!-\!\!\!$  Proyecto}  $\Rightarrow$  {Nombre  $\rightarrow\!\!\!-\!\!\!$  Proyecto}.

54

## 4FN (Cuarta Forma Normal)

- **Cuarta Forma Normal (4FN):** Si para cada dependencia multivaluada no trivial  $X \text{ } \textcircled{R} \text{ } Y$ , 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  $\underbrace{\{R - Y\}}_{\{X, Z\}}$ .

- Al dividir, las **DMV** no se pierden, sino que se convierten en **triviales**.

- **Del ejemplo:**

Nombre	Proyecto
Príamo	Caballo
Príamo	Troya

Nombre	Hijo
Príamo	Cassandra
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.

55

## DMV y 4FN: Ejemplo de una Fábrica

- **Fábrica (Empleado, Máquina, Producto)**, con las siguientes dependencias:

- **D1:** Empleado  $\textcircled{R}$  Máquina (cada empleado maneja sólo una máquina).
- **D2:** Máquina  $\textcircled{R} \text{ } \textcircled{R}$  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  $\textcircled{R} \text{ } \textcircled{R}$  Empleado.
  - No se cumple Empleado  $\textcircled{R} \text{ } \textcircled{R}$  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 en 2:

- **Maneja (Empleado, Máquina)** y **Produce (Máquina, Producto)** [ambas en 4FN]

- **Si añadimos otra DF:**

- **D3:** Empleado  $\textcircled{R}$  Producto (cada empleado sólo fabrica un producto).
- Resultado: **Fábrica (Empleado, Máquina, Producto)**
  - Por la **Regla de Fusión**, tenemos que Máquina  $\textcircled{R}$  Producto (DF, no DMV).
- **NO** está en **FNBC** (<sup>Máquina no</sup><sub>es superllave</sub>): Dividimos en 2 (cambia una llave primaria):
  - **Maneja (Empleado, Máquina)** y **Produce (Máquina, Producto)** [ambas en 4FN]

56

## 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  $\bowtie$  Producto (y Máquina  $\bowtie$  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  $\bowtie$  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**  $\circledR$  **Y** está en **G<sup>+</sup>**, también estará en **F<sup>+</sup>**: Eso **NO** ocurre con **DMV**.
  - Al descomponer siempre hay alguna DMV que no se transmite completa:
    - Si **X**  $\bowtie$  **Y** se transmitiera completa, no se transmitiría completa su complementaria.

57

## 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**  $\bowtie$  **B** y **A**  $\bowtie$  **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 d<sup>a</sup> de **Fecha**.
  - Sólo hay una DF: {Alumno, Req}  $\circledR$  Fecha.
    - La relación no está en **2FN**, porque Fecha depende parcialmente de la llave.
    - NO se cumple la DMV: Asignatura  $\bowtie$  Alumno (aunque una **Asignatura** puede tener varios **Alumnos** matriculados todos estos alumnos no tienen que tener las asignaturas **Requeridas** aprobadas en las mismas **Fechas**).
  - Descomposición para **FNBC**:
    - **Aprobadas (Alumno, Req, Fecha)**.
    - **Matriculadas\_con\_Requi (Alumno, Asignatura, Req)**.

58

## 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  $\textcircled{R}$  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**  $\Rightarrow$  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)  $\Rightarrow$  Asignaturas Aprobadas y su Fecha.
    - **Matriculadas** (Alumno, Asignatura)  $\Rightarrow$  Alumnos Matriculados y sus Asignat.
    - **Requisitos** (Asignatura, Req)  $\Rightarrow$  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.

59

## Dependencias Jerárquicas (DMV Múltiple)

- Si se dice que la DMV ( $X \textcircled{R} Y | Z$ ) 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  $\textcircled{R}$  Req | Alumno, es embebida en **Matricula\_Requi** porque falta el atributo Fecha, y es explícita en **Matriculadas\_con\_Requi**.
- **Dependencia Jerárquica**  $X \textcircled{R} Y_1 | Y_2 | \dots | Y_n$ , o DMV Múltiple:  
Si en una relación **R** con los siguientes conjuntos disjuntos de atributos ( $X, Y_1, Y_2, \dots, Y_n, Z$ ), existen asociaciones independientes entre los valores ( $X, Y_1$ ), ( $X, Y_2$ ), ..., ( $X, Y_n$ ).
  - **Explícita:** Si  $Z = \emptyset$ , es decir, si están incluidos todos los atributos de **R**.
    - Si  $Z \neq \emptyset$ , será una dependencia embebida o implícita en **R**, que será explícita en **S**, siendo **S** la proyección de **R** sobre los atributos ( $X, Y_1, Y_2, \dots, Y_n$ ).
  - Existe ( $X \textcircled{R} Y_1 | Y_2 | \dots | Y_n$ ), si para todo valor  $x$  del atributo **X** se cumple:  $S_{X=x}(S) = p_{X,Y_1}(S_{X=x}(S)) * p_{X,Y_2}(S_{X=x}(S)) * \dots * p_{X,Y_n}(S_{X=x}(S))$ ;
  - **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: ( $X, Y_1$ ), ( $X, Y_2$ ), ..., ( $X, Y_n$ ).

60

## Dependencias Jerárquicas: Ejemplo

- La siguiente relación (**S**) en **FNBC** cumple la **Dep. Jerárquica**:
  - Entidad  $\bowtie$  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  $x = \text{'Chicho Méndez'}$ :

$$S_{X=x}(S) = p_{X, \text{Acción}}(S_{X=x}(S)) * p_{X, \text{Cualidad}}(S_{X=x}(S)) * p_{X, \text{Herramienta}}(S_{X=x}(S)) = \\ = (\text{Chicho Méndez, Defender Amazonas}) * \begin{cases} \text{Chicho Méndez, Amor} \\ \text{Chicho Méndez, Sensatez} \end{cases} * \begin{cases} \text{Chicho Méndez, Palabra} \\ \text{Chicho Méndez, Reforestación} \end{cases}$$

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

61

## Dependencias de Reunión y 5FN

- Dependencia de Reunión** (funcional o composicional) en **R**:  $*(X_1, X_2, \dots, X_n)$   
Si la reunión entre las proyecciones sobre los conj. de atributos **X** generan la relación original **R**. Si  $R_i = R[X_i]$ :  $R = R_1 * R_2 * \dots * R_n$ ;
  - Lógicamente:  $\bigcup_{i=1}^n X_i = R$ ;
  - Una DMV es una Dependencia de Reunión con  $n = 2$ :
    - $*(X_1, X_2)$  implica que  $(R_1 \cap R_2) \bowtie (R_1 - R_2)$ ; (A la fuerza  $R_1 \cap R_2 \neq \emptyset$ )
    - Dependencia de Reunión Trivial: Si una **R<sub>i</sub>** 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<sup>+</sup>**, denotada por  $*(X_1, X_2, \dots, X_n)$ , cada **X<sub>i</sub>** 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)

62

## Dep. de Reunión: Ejemplo

- Solución:** Dividir  $R$  en  $R_1, R_2, \dots, R_n$ .
  - En el Ej., la Dep. de Reunión  $*(X_1, X_2, X_3)$ , se divide en las 3 relaciones:
    - $R_1$  (ONG, Proyecto)
    - $R_2$  (Proyecto, Lugar)
    - $R_3$  (ONG, 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

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



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

63

## 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:  $p_X(R) \sqsubseteq p_Y(S)$ 
  - $X$  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:**
      - $CENTRAL_SOLAR.Nombre < CENTRAL_LIMPIA.Nombre$ ,  $CENTRAL_EOLICA.Nombre < CENTRAL_LIMPIA.Nombre$
      - $Persona.NIF < Empleado.NIF$ ,  $Persona.NIF < Estudiante.NIF...$
  - Reglas de Inferencia:**
    - Reflexiva:  $R.X < R.X$ .
    - Correspondencia de Atributos:  $R.\{A_1, A_2, \dots, A_n\} < S.\{B_1, B_2, \dots, B_n\} \models R.A_i < S.B_i$  para  $1 \leq i \leq n$ .
    - Transitiva:  $\{R.X < S.Y, S.Y < T.Z\} \models R.X < T.Z$ .

64