

# Modelo Relacional

*Autor: Sergio D'Arrigo*

# Origen

## A Relational Model of Data for Large Shared Data Banks

E. F. CODD  
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

**KEY WORDS AND PHRASES:** data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity  
**CR CATEGORIES:** 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

El Modelo Relacional tiene sus orígenes en el artículo ***"A Relational Model of Data for Large Shared Data Banks."***, publicado en 1970 por 1970 **Edgar F. Codd**

Aquí define el modelo relacional y formas no procedurales de consultar datos en dicho modelo.

Presenta el concepto matemático de relación como su construcción básica, con un fundamento teórico basado en la teoría de conjuntos y la lógica de predicados de primer orden.

# Conceptos Básicos

## Relación

- Cada relación se asemeja a una tabla de filas y columnas.
- Cada fila representa una colección de valores de datos relacionados
- Cada fila representa un hecho que se corresponde con una entidad o una interrelación del mundo real
- Formalmente en el modelo relacional:
  - La tabla es una **Relación**
  - La fila es una **Tupla**
  - Las columnas (o los encabezados de las columnas) son los **Atributos**
  - Los tipos de datos de las columnas son representados por un **Dominio** de posibles valores

# Conceptos Básicos

## Esquema de relación

- Consiste de un nombre y una lista de atributos:  $R(A_1, A_2, \dots A_n)$
- Un **dominio** D es un conjunto de valores atómicos
- Cada atributo es el nombre de un rol sobre algún dominio D en el esquema R
  - Necesitamos roles porque un mismo dominio puede aplicar a más de un atributo.
- El dominio del atributo  $A_i$  se denota como **dom** ( $A_i$ )
- La relación R es un conjunto incluido en el producto cartesiano de sus dominios

$$r(R) \subseteq (\text{dom}(A_1) \times \text{dom}(A_2) \times \dots \times \text{dom}(A_n))$$

- **Aridad o grado de la relación:** Es la cantidad de atributos que tiene el esquema de la relación.

# Conceptos Básicos

- Una esquema de relación  $R(A_1, A_2, \dots, A_n)$  en cada momento particular tiene un **estado** que denominamos  **$r(R)$**
- $r(R)$  es un **conjunto** de  $n$ -tuplas  $r(t_1, t_2, \dots, t_m)$
- Cada tupla  $t_i$  es una **lista ordenada** de  $n$  valores  $t = \langle v_1, v_2, \dots, v_n \rangle$  donde cada  $v_i$ ,  $1 \leq i \leq n$ , pertenece a  $\text{dom}(A_i)$
- El número total de tuplas que puede tener  $r(R)$  es

$$|\text{dom}(A_1)| \times |\text{dom}(A_2)| \times \dots \times |\text{dom}(A_n)|$$

- El **estado  $r(R)$  cambia muy frecuentemente** (acompañando al mundo real)
- El **esquema de relación  $R$  cambia muy infrecuentemente**

# Características de las relaciones

- No hay orden entre las tuplas
  - La relación es un conjunto de tuplas, en el sentido matemático
- La lista de valores de atributos de una tupla tiene orden
  - Las tuplas son una lista ordenada de valores
  - De todos modos, este orden **no es esencial**, podría cambiarse en la medida que se haga consistentemente y se mantenga la correspondencia de atributos y valores
- Interpretación de una relación
  - Afirmación
  - Predicado

# Características de las relaciones

- Valores en las tuplas
  - Cada valor de un **atributo** en una tupla es un **valor atómico**, indivisible en el contexto del modelo relacional
  - Con esta definición, **no son permitidos los atributos multivaluados ni los compuestos.**
    - Se puede reformular su representación:
      - 1 atributo multivaluado → entidad con interrelación 1:N
      - 1 atributo compuesto → n atributos atómicos
- Valores **NULOS**
  - Es un valor especial, que se usa cuando se necesita representar que el valor de un atributo es desconocido o no aplica para una tupla en particular.
  - Es la ausencia de valor.
  - El significado concreto pueda variar según el universo de discurso representado y el modelo planteado.

# Restricciones del modelo relacional

- Las tuplas de las diferentes relaciones usualmente están vinculadas de varias maneras.
- Las **reglas de dominio** del universo de discurso representado usualmente **determinan restricciones** sobre los valores que pueden contener las tuplas de las relaciones de una base de datos, es decir, restricciones sobre los valores actuales del estado de la base de datos completa.
- Se pueden clasificar en
  - Restricciones implícitas o **basadas en el modelo** (x ej, cada relación es un conjunto de tuplas)
  - Restricciones explícitas o **basadas en el esquema**
  - Restricciones semánticas o **basadas en las aplicaciones**
- Veremos las restricciones basadas en el esquema

# Restricciones de dominio

- Los atributos **deben ser atómicos** respecto del dominio al que pertenecen.
- Los dominios tienen tipos de datos asociados
  - Por ej, números enteros, números reales, caracteres, lógicos, strings (longitud fija o variable), fechas, horas, timestamps.
- Un dominio definido en el modelo relacional puede ser descripto:
  - Por un tipo de datos
  - Por un rango de un tipo de datos
  - Por una lista de valores enumerada explícitamente

# Restricciones sobre claves

- Por definición, las relaciones son conjuntos, no hay tuplas repetidas.
  - $R(A_1, A_2, \dots A_n) \Rightarrow t_i[A_1, \dots A_n] \neq t_j[A_1, \dots A_n]$  para todo  $i \neq j$
- Pero en general hay restricciones más fuertes, donde las combinaciones de ciertos subconjuntos de esos atributos tampoco pueden repetirse . Esto es
  - Para un cierto SK subconjunto de  $\{A_1, \dots, A_n\}$ 
    - $T_i[SK] \neq T_j[SK]$  para todo  $i \neq j$
- SK es llamado **Superclave** del esquema de relación R.
- Una superclave especifica una restricción de **unicidad** sobre las tuplas de un estado de la relación.
- Una superclave puede tener atributos redundantes, que no son necesarios para especificar unicidad.

# Restricciones sobre claves

- Llamamos **Clave** a una superclave que no tiene atributos redundantes, es decir, que si se quita cualquiera de sus atributos se pierde la restricción de unicidad.
- Propiedades de las **claves**:
  - **Restricción de unicidad**: dos tuplas distintas de un estado de relación no pueden tener idéntica combinación de todos los atributos que componen la clave.
  - **Minimalidad**: Si se remueve alguno de sus atributos, se pierde la restricción de unicidad

☞ *Toda clave es superclave, pero no todas las superclaves son claves*

# Restricciones sobre claves

- Un esquema de relación puede tener **más de una clave**
    - Por ej, Legajo y CUIL
  - A cada una de las claves, las llamamos **Claves Candidatas** (CK)
  - Llamamos **Clave Primaria** (PK) a una clave candidata seleccionada para identificar las tuplas en una relación
    - ☞ *El criterio general de selección es la simplicidad (atributo único, o menor cantidad de atributos)*
    - ☞ *Spoiler: Al implementar en una BBDD, la clave primaria se especifica como PK y el resto de las claves candidatas se especifican como claves únicas*
  - Las **claves primarias** se denotan subrayando el atributo con **línea continua**.
    - ALUMNO (LU, Apellido, Nombre, Fecha\_Nacimiento, E\_mail, Genero, DNI)
      - PK {LU}
      - CK = {LU, {Genero, DNI}} /\*\* Genero-DNI clave compuesta\*\*/

# Restricciones sobre valores nulos

- Restricciones de **nulidad**
- Para cada atributo especifica si se permite que tome valores nulos o no.

# Restricciones de Integridad

- Restricción de **integridad de clave**:
  - Ningún valor de clave primaria puede ser nulo
- Restricciones de **integridad referencial**:
  - Definen restricciones **entre dos relaciones**
  - Buscan **mantener consistencia** entre las tuplas de ambas relaciones
  - Si una tupla de una relación referencia a otra relación, en la segunda debe existir una tupla que esté referenciada.

# Restricciones de Integridad

- **Clave Foránea:**

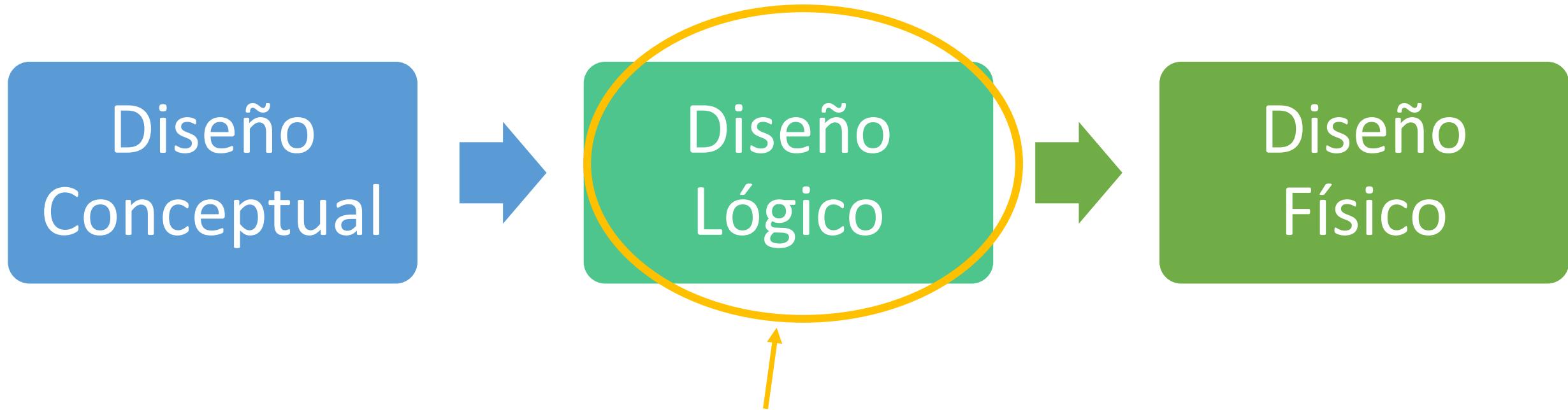
- Un conjunto de atributos FK en un esquema de relación R1 es una clave foránea del esquema R1 que referencia a un esquema R2 si:
  - Los atributos de FK tienen **los mismos dominios** que los atributos de la PK de R2
  - Dado un valor particular de FK en una tupla  $t_1$  de  $r(R1)$ , se cumple una de las siguientes condiciones:
    - Existe una tupla  $t_2$  en  $r(R2)$  cuya PK tiene ese mismo valor
    - El valor FK es nulo
- Decimos que R1 referencia a R2 (o que la clave foránea FK de R1 referencia a la clave primaria PK de R2)
- Las restricciones de integridad referencial **típicamente derivan de interrelaciones** entre entidades representadas por los esquemas de relación

# Esquema de bases de datos relacionales

- Un **esquema de base de datos relacional** S está compuesto de:
  - Un conjunto de esquemas de relación
  - Un conjunto de restricciones de integridad
- El **estado de base de datos relacional** puede ser válido o inválido:
  - Estado **válido**: El estado de la base de datos satisface todas las restricciones de integridad definidas en el esquema.
  - Estado **inválido**: Al menos una restricción de integridad no es satisfacida.

# Pasaje a Modelo Lógico Relacional

En el proceso de diseño, una vez que contamos con un Modelo de Entidad Relación, debemos **generar el correspondiente Modelo Lógico Relacional** (o directamente llamado Modelo Relacional).

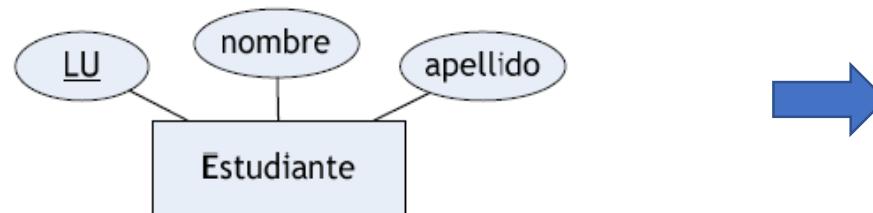


- ✓ Transformaremos el Diagrama de Entidad-Relación en un Modelo Lógico Relacional, mediante un conjunto de reglas de derivación o transformación.

# Transformación del MER a MLR

## Entidades Fuertes

- Las **entidades fuertes** se transforman como **Relaciones**
- Los **atributos simples** de la entidad se convierten en **Atributos** de la Relación
- Los **atributos identificatorios** se convierten en Clave Primaria de la Relación
- Ej



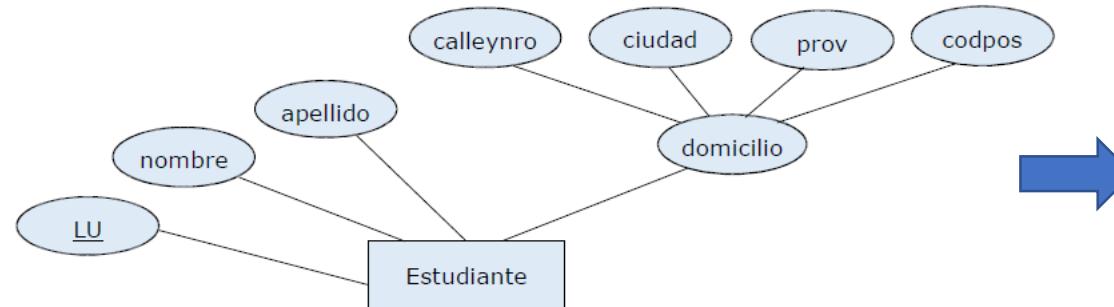
Esquemas resultantes

**Estudiante(LU, nombre, apellido)**

# Transformación del MER a MLR

## Entidades Fuertes

- Las **entidades fuertes** se transforman como **Relaciones**
- Los **atributos simples** de la entidad se convierten en **Atributos** de la Relación
- Los **atributos identificatorios** se convierten en Clave Primaria de la Relación
- Los **atributos compuestos** se “planchan”. Cada **subparte** se convierte en un **atributo simple**. La raíz se descarta.
- Ej



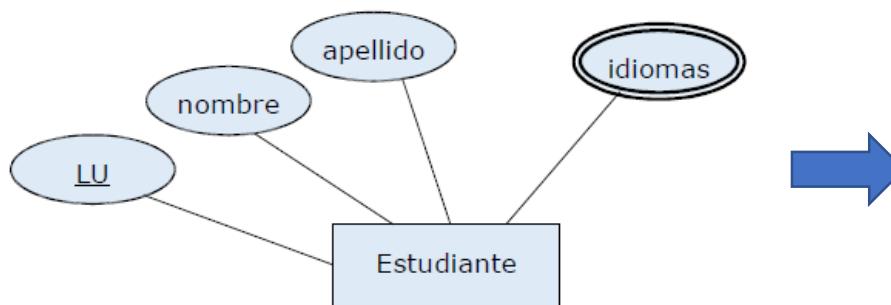
Esquemas resultantes

**Estudiante(LU, nombre, apellido, calleynro, ciudad, prov, codpos)**

# Transformación del MER a MLR

## Entidades Fuertes

- Los **atributos multivaluados** se convierten en una **nueva relación** que tiene un PK compuesta por la PK de la entidad y el propio atributo. La clave parcial de la nueva relación que referencia a la clave primaria de la entidad original se convierte en FK.
- Ej



### Esquemas resultantes

**ESTUDIANTE (LU, nombre, apellido)**

PK = CK = {LU}

**IDIOMA (LU, idioma)**

PK = CK = {{LU, idioma}} notar la doble llave

FK = {LU}

### Restricciones adicionales

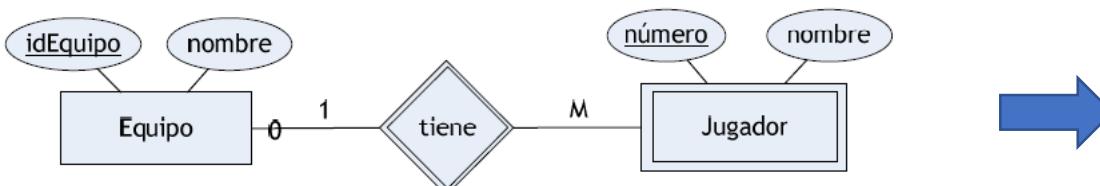
ESTUDIANTE.LU puede no estar en IDIOMA.LU

IDIOMA.LU debe estar en ESTUDIANTE.LU

# Transformación del MER a MLR

## Entidades Débiles

- También se transforman como relaciones
- La transformación de atributos descriptivos sigue las **mismas reglas** que en las entidades fuertes.
- La **clave primaria** de la relación se conforma conjuntamente por el **atributo identificatorio de la entidad padre** y por el **atributo identificatorio parcial de la entidad débil**



Esquemas resultantes

**EQUIPO** (idEquipo, nombre)

PK = CK = { idEquipo }

**JUGADOR** (idEquipo, número, nombre)

PK = CK = { { idEquipo, número } } notar la doble llave

FK = { idEquipo }

Restricciones adicionales

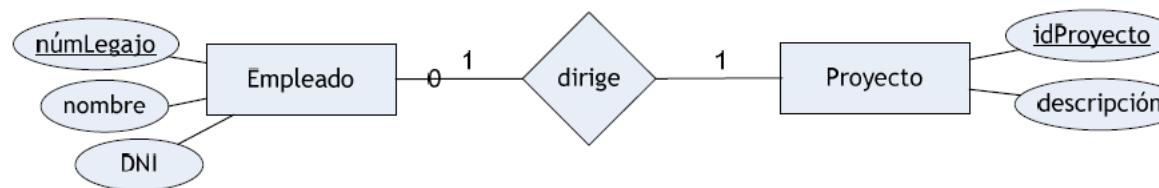
EQUIPO.idEquipo puede no estar en JUGADOR.idEquipo

JUGADOR.idEquipo debe estar en EQUIPO.idEquipo

# Transformación del MER a MLR

## Interrelaciones uno a uno (1:1)

- El atributo identificatorio de una de las entidades se incorpora a la otra relación como clave foránea



- ¿da lo mismo llevar la clave foránea a cualquiera de las dos entidades?
- Siempre priorizar la participación total, para evitar claves foráneas con valores nulos.

### Esquemas resultantes

**EMPLEADO** (númLegajo, nombre, DNI)

PK = {númLegajo}

CK = {númLegajo, DNI}

**PROYECTO** (idProyecto, descripción, númLegajo)

PK = CK = {idProyecto}

FK = {númLegajo}

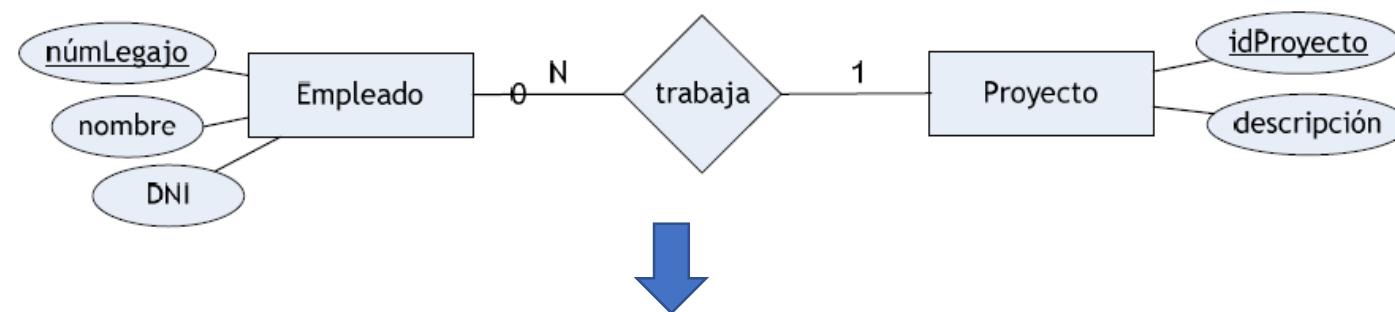
### Restricciones adicionales

EMPLEADO.númLegajo puede no estar en PROYECTO. númLegajo PROYECTO. númLegajo debe estar en EMPLEADO.númLegajo

# Transformación del MER a MLR

## Interrelaciones uno a muchos (1:N)

- El atributo identificatorio de la entidad “UNO” se lleva a la relación “MUCHOS” como clave foránea



### Esquemas resultantes

**EMPLEADO** (númLegajo, nombre, DNI, idProyecto)

PK = {númLegajo}

CK = {númLegajo, DNI}

FK = {idProyecto}

**PROYECTO** (idProyecto, descripción, númLegajo)

PK = CK = {idProyecto}

### Restricciones adicionales

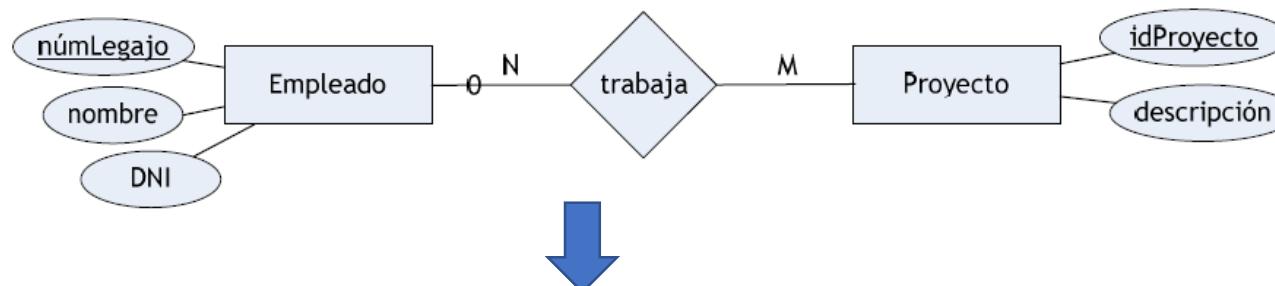
EMPLEADO.idProyecto es nulo o debe estar en PROYECTO.idProyecto

PROYECTO.idProyecto debe estar en EMPLEADO.idProyecto

# Transformación del MER a MLR

## Interrelaciones muchos a muchos (M:N)

- Se crea una **nueva relación** para la interrelación, con una PK compuesta por los atributos identificatorios de las dos entidades interrelacionadas.



### Esquemas resultantes

**EMPLEADO** (númeroLegajo, nombre, DNI)

PK = {númeroLegajo}

CK = {númeroLegajo, DNI}

**PROYECTO** (idProyecto, descripción, númeroLegajo)

PK = CK = {idProyecto}

**TRABAJA** (númeroLegajo, idProyecto)

PK = CK = {{númeroLegajo, idProyecto}}

FK = {númeroLegajo, idProyecto}

### Restricciones adicionales

EMPLEADO.NúmeroLegajo puede no estar en TRABAJA.númeroLegajo

TRABAJA.númeroLegajo debe estar en EMPLEADO.NúmeroLegajo

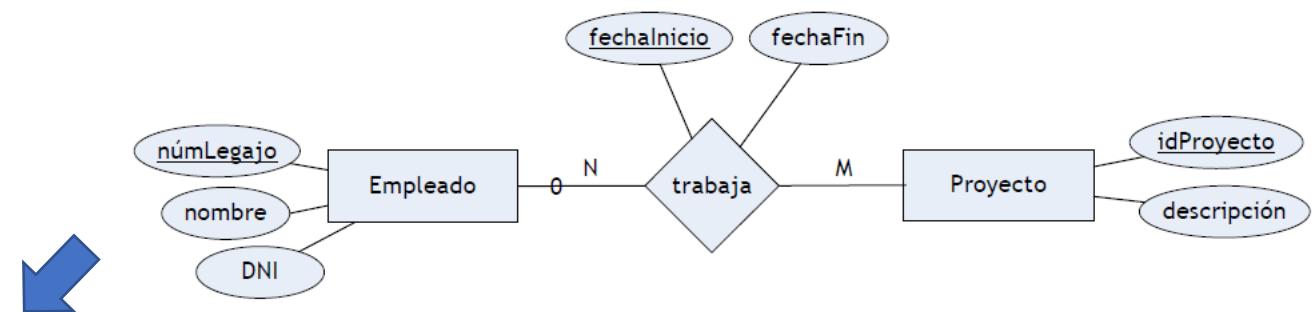
PROYECTO.idProyecto debe estar en TRABAJA.idProyecto

TRABAJA.idProyecto debe estar en PROYECTO.idProyecto

# Transformación del MER a MLR

## Interrelaciones muchos a muchos (M:N) con atributos

- Similar a lo visto, con la diferencia que el atributo identificatorio de la interrelación se incorpora en la relación correspondiente como parte de la PK, y los atributos descriptivos se incorporan como atributos descriptivos



### Esquemas resultantes

**EMPLEADO** (númLegajo, nombre, DNI)

PK = {númLegajo}

CK = {númLegajo, DNI}

**PROYECTO** (idProyecto, descripción, númLegajo)

PK = CK = {idProyecto}

**TRABAJA** (númLegajo, idProyecto, fechalinicio, fechaFin)

PK = CK = {númLegajo, idProyecto, fechalinicio}

FK = {númLegajo, idProyecto}

### Restricciones adicionales

EMPLEADO.NúmLegajo puede no estar en TRABAJA.númLegajo

TRABAJA.númLegajo debe estar en EMPLEADO.NúmLegajo

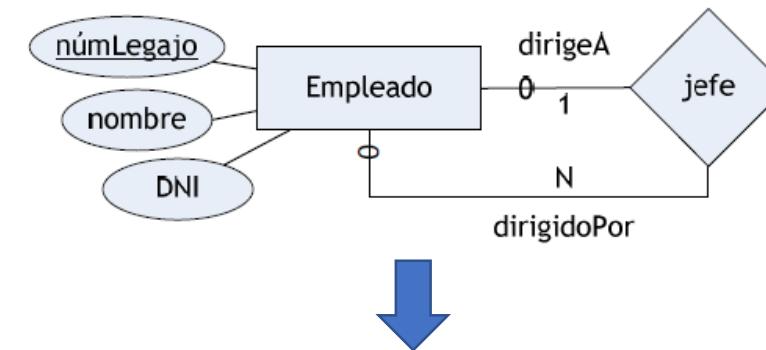
PROYECTO.idProyecto debe estar en TRABAJA.idProyecto

TRABAJA.idProyecto debe estar en PROYECTO.idProyecto

# Transformación del MER a MLR

## Interrelaciones unarias

- Similar a lo visto para las binarias, sólo que ahora hay sólo una entidad para transformar.



## Esquemas resultantes

**EMPLEADO** (númLegajo, nombre, DNI, númLegajoJefe)

PK = {númLegajo}

CK = {númLegajo, DNI}

FK = {númLegajoJefe}

## Restricciones adicionales

EMPLEADO.NúmLegajoJefe es nulo o debe estar en EMPLEADO.númLegajo

EMPLEADO.NúmLegajo puede no estar en EMPLEADO.NúmLegajoJefe

# Transformación del MER a MLR

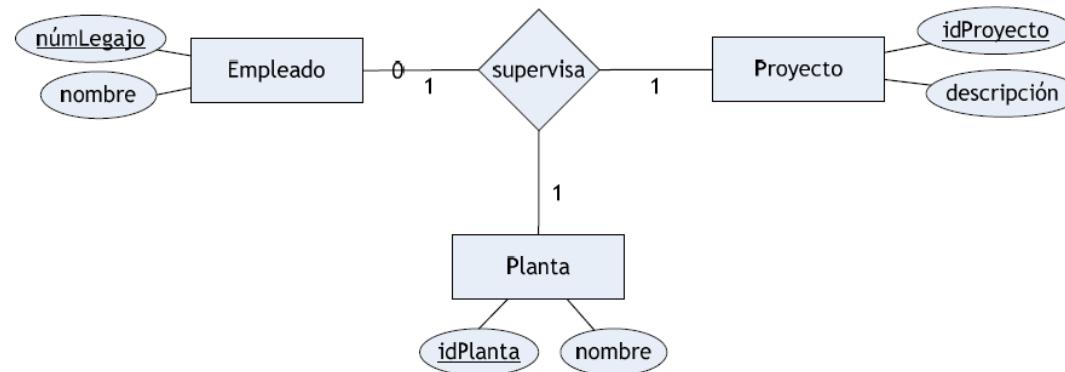
## Interrelaciones ternarias

- Se crea un nuevo esquema de relación para la interrelación.
- La definición de las **claves** del esquema de relación generado por la interrelación **dependerá de la cardinalidad** de la interrelación.
- Criterio:
  - Si **un par de entidades determina a la restante** (es decir, tiene cardinalidad “1”), las claves identificatorias de esas dos entidades serán una clave candidata de la relación en cuestión.
  - Si **ninguno de los 3 pares de entidades determina a la restante**, entonces la clave candidata estará compuesta por los atributos identificatorios de las 3 entidades

# Transformación del MER a MLR

## Interrelaciones ternarias 1:1:1

- Los 3 pares de entidades determinan a la restante. **Habrá 3 claves candidatas compuestas.**



### Esquemas resultantes

**EMPLEADO** (númLegajo, nombre)

PK = CK = {númLegajo}

**PROYECTO** (idProyecto, descripción)

PK = CK = {idProyecto}

**PLANTA** (idPlanta, nombre)

PK = CK = {idPlanta}

**SUPERVISA** (númLegajo, idProyecto, idPlanta)

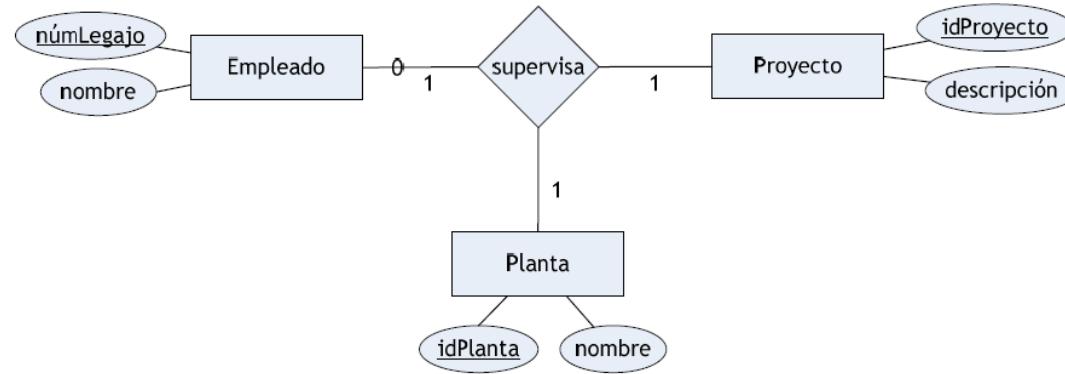
FK = {númLegajo, idProyecto, idPlanta}

¿Y cuales son las claves?

# Transformación del MER a MLR

## Interrelaciones ternarias 1:1:1

- Los 3 pares de entidades determinan a la restante. **Habrá 3 claves candidatas compuestas.**



### Esquemas resultantes

**EMPLEADO** (númeroLegajo, nombre)

PK = CK = {númeroLegajo}

**PROYECTO** (idProyecto, descripción)

PK = CK = {idProyecto}

**PLANTA** (idPlanta, nombre)

PK = CK = {idPlanta}

**SUPERVISA** (númeroLegajo, idProyecto, idPlanta)

FK = {númeroLegajo, idProyecto, idPlanta}

CK = { {númeroLegajo, idProyecto}, {idProyecto, idPlanta}, {númeroLegajo, idPlanta} }

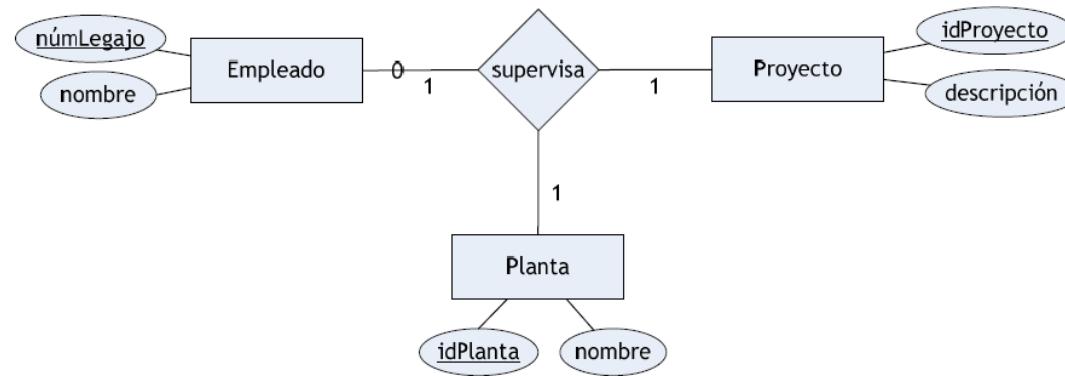


Elegiremos una de ellas como PK

# Transformación del MER a MLR

## Interrelaciones ternarias 1:1:1

- Los 3 pares de entidades determinan a la restante. **Habrá 3 claves candidatas compuestas.**



### Restricciones adicionales

SUPERVISA.númeroLegajo debe estar en Empleado.númeroLegajo

SUPERVISA.idProyecto debe estar en Proyecto.idProyecto

SUPERVISA.idPlanta debe estar en Planta.idPlanta

Empleado.númeroLegajo puede no estar en SUPERVISA.númeroLegajo

Proyecto.idProyecto debe estar en SUPERVISA.idProyecto

Planta.idPlanta debe estar en SUPERVISA.idPlanta

### Esquemas resultantes

**EMPLEADO** (númeroLegajo, nombre)

PK = CK = { númeroLegajo }

**PROYECTO** (idProyecto, descripción)

PK = CK = { idProyecto }

**PLANTA** (idPlanta, nombre)

PK = CK = { idPlanta }

**SUPERVISA** (númeroLegajo, idProyecto, idPlanta)

FK = { númeroLegajo, idProyecto, idPlanta }

CK = { { númeroLegajo, idProyecto }, { idProyecto, idPlanta }, { númeroLegajo, idPlanta } }

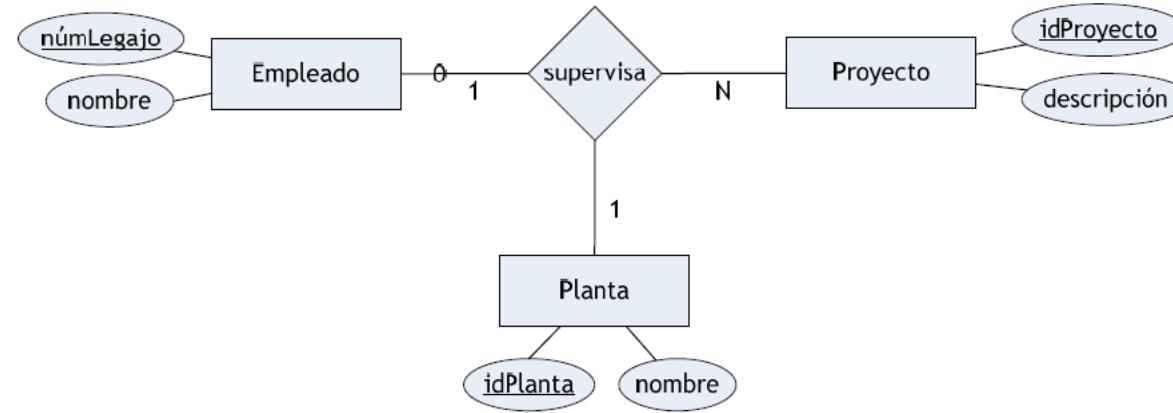
PK = { númeroLegajo, idProyecto }



# Transformación del MER a MLR

## Interrelaciones ternarias 1:1:N

- 2 de los 3 pares de entidades determinan a la restante. **Habrá 2 claves candidatas compuestas.**



### Restricciones adicionales

SUPERVISA.numLegajo debe estar en Empleado.numLegajo

SUPERVISA.idProyecto debe estar en Proyecto.idProyecto

SUPERVISA.idPlanta debe estar en Planta.idPlanta

Empleado.numLegajo puede no estar en SUPERVISA.numLegajo

Proyecto.idProyecto debe estar en SUPERVISA.idProyecto

Planta.idPlanta debe estar en SUPERVISA.idPlanta

### Esquemas resultantes

**EMPLEADO** (numLegajo, nombre)

PK = CK = {numLegajo}

**PROYECTO** (idProyecto, descripción)

PK = CK = {idProyecto}

**PLANTA** (idPlanta, nombre)

PK = CK = {idPlanta}

**SUPERVISA** (numLegajo, idProyecto, idPlanta)

FK = {numLegajo, idProyecto, idPlanta}

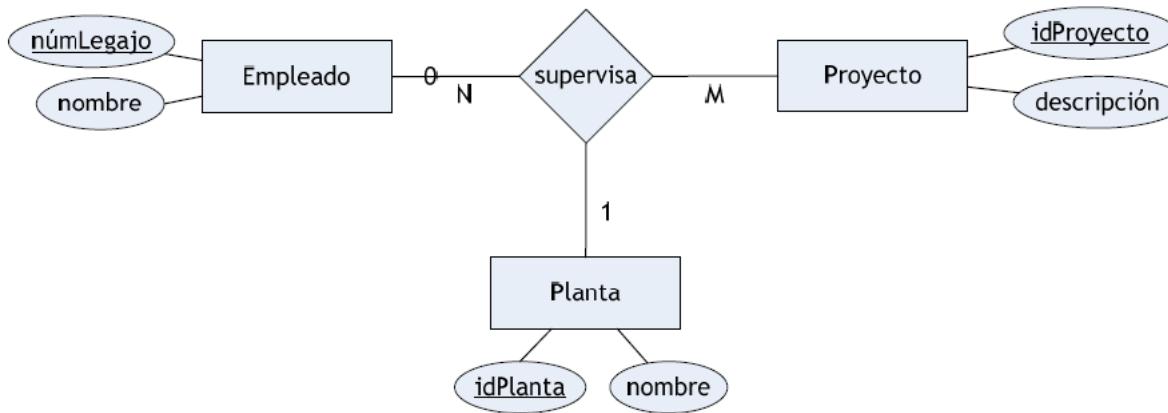
CK = { {numLegajo, idProyecto}, {idProyecto, idPlanta} }

PK = {numLegajo, idProyecto}

# Transformación del MER a MLR

## Interrelaciones ternarias 1:N:M

- Sólo un par de entidades determina a la restante. Habrá sólo 1 claves candidata (compuesta x 2)



### Restricciones adicionales

SUPERVISA.númLegajo debe estar en Empleado.númLegajo

SUPERVISA.idProyecto debe estar en Proyecto.idProyecto

SUPERVISA.idPlanta debe estar en Planta.idPlanta

Empleado.númLegajo puede no estar en SUPERVISA.númLegajo

Proyecto.idProyecto debe estar en SUPERVISA.idProyecto

Planta.idPlanta debe estar en SUPERVISA.idPlanta

### Esquemas resultantes

**EMPLEADO** (númLegajo, nombre)

PK = CK = {númLegajo}

**PROYECTO** (idProyecto, descripción)

PK = CK = {idProyecto}

**PLANTA** (idPlanta, nombre)

PK = CK = {idPlanta}

**SUPERVISA** (númLegajo, idProyecto, idPlanta)

FK = {númLegajo, idProyecto, idPlanta}

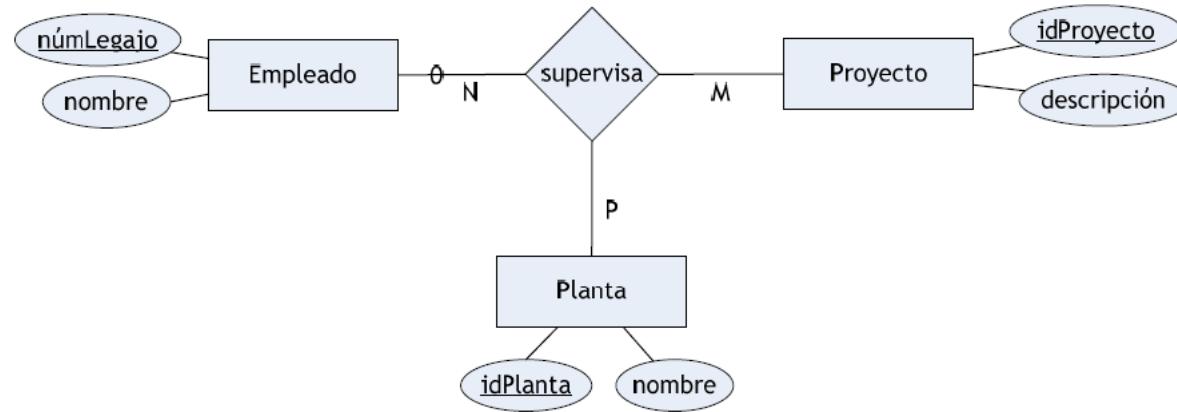
CK = { {númLegajo, idProyecto} }

PK = {númLegajo, idProyecto}

# Transformación del MER a MLR

## Interrelaciones ternarias M:N:P

- Ningún par de entidades determina a la restante. Habrá 1 clave candidata (compuesta x 3).



### Restricciones adicionales

SUPERVISA.numLegajo debe estar en Empleado.numLegajo

SUPERVISA.idProyecto debe estar en Proyecto.idProyecto

SUPERVISA.idPlanta debe estar en Planta.idPlanta

Empleado.numLegajo puede no estar en SUPERVISA.numLegajo

Proyecto.idProyecto debe estar en SUPERVISA.idProyecto

Planta.idPlanta debe estar en SUPERVISA.idPlanta

### Esquemas resultantes

**EMPLEADO** (numLegajo, nombre)

PK = CK = {numLegajo}

**PROYECTO** (idProyecto, descripción)

PK = CK = {idProyecto}

**PLANTA** (idPlanta, nombre)

PK = CK = {idPlanta}

**SUPERVISA** (númLegajo, idProyecto, idPlanta)

FK = {númLegajo, idProyecto, idPlanta}

CK = { {númLegajo, idProyecto, idPlanta} }

PK = { {númLegajo, idProyecto, idPlanta} }

# Transformación del MER a MLR

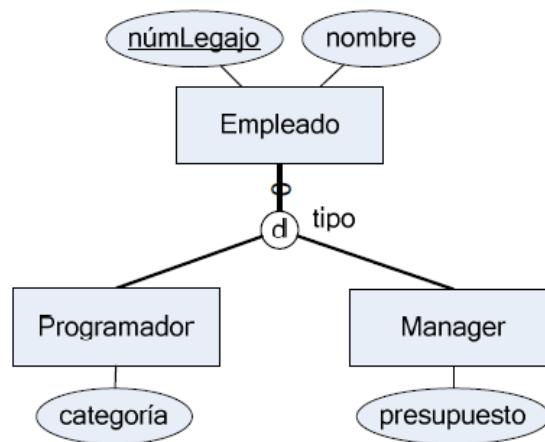
## Jerarquías

- La regla es generar **una relación por la superentidad y una por cada subentidad**.
- Los atributos identificatorios de las superentidades se convierten en las PK de las relaciones derivadas de la superentidad y de sus subentidades.

# Transformación del MER a MLR

## Jerarquía disjunta

- El discriminante se agrega como un atributo descriptivo en la relación surgida de la superentidad.



### Esquemas resultantes

**EMPLEADO** (númLegajo, nombre, tipo)

PK = CK = {númLegajo}

**PROGRAMADOR** (idEmpleado, categoría)

PK = CK = {idEmpleado}

**MANAGER** (idEmpleado, presupuesto)

PK = CK = {idEmpleado}

### Restricciones adicionales

PROGRAMADOR.númLegajo debe estar en EMPLEADO.NúmLegajo

MANAGER.númLegajo debe estar en EMPLEADO.NúmLegajo

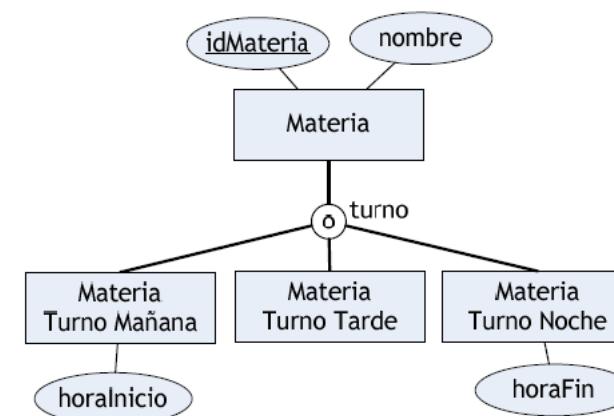
EMPLEADO.NúmLegajo puede estar en PROGRAMADOR.númLegajo o (exclusivo) puede estar en MANAGER.númLegajo o (exclusivo) puede no estar en ninguno de los anteriores

*(esto último es por la participación parcial de la superentidad en la jerarquía)*

# Transformación del MER a MLR

## Jerarquía con solapamiento

- Aquí el discriminante no se agrega como un atributo descriptivo.



Esquemas resultantes

**MATERIA** (idMateria, nombre)

PK = CK = { idMateria }

**MATERIA\_T\_MAÑANA** (idMateria, horaInicio)

PK = CK = { idMateria }

**MATERIA\_T\_TARDE** (idMateria)

PK = CK = { idMateria }

**MATERIA\_T\_NOCHE** (idMateria, horaFin)

PK = CK = { idMateria }

## Restricciones adicionales

MATERIA\_T\_MAÑANA.idMateria debe estar en MATERIA.idMateria

MATERIA\_T\_TARDE.idMateria debe estar en MATERIA.idMateria

MATERIA\_T\_NOCHE.idMateria debe estar en MATERIA.idMateria

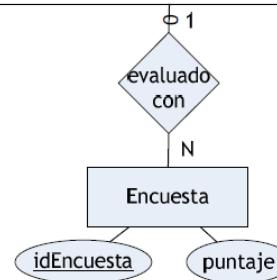
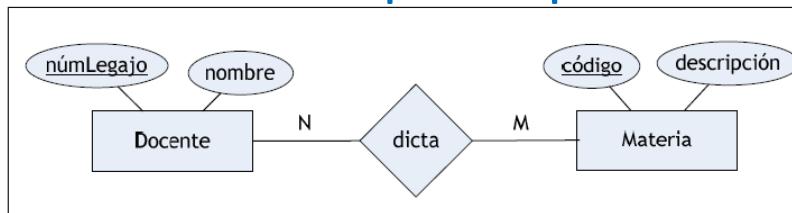
MATERIA.idMateria puede estar en MATERIA\_T\_MAÑANA.idMateria , MATERIA\_T\_TARDE.idMateria y

MATERIA\_T\_NOCHE.idMateria simultáneamente

# Transformación del MER a MLR

## Agregación

- Reglas similares a las de las interrelaciones binarias, considerando a la **agregación como una de las entidades participantes en la interrelación “exterior”**



### Esquemas resultantes

**DOCENTE** (número de legajo, nombre)

PK = CK = {número de legajo}

**MATERIA** (código, nombre)

PK = CK = {código}

**DICTA** (número de legajo, código)

PK = CK = {número de legajo, código}

**ENCUESTA** (idEncuesta, puntaje, número de legajo, código)

PK = CK = {idEncuesta}

FK = {{número de legajo, código}}

### Restricciones adicionales

DICTA.número de legajo debe estar en DOCENTE.número de legajo

DICTA.código debe estar en MATERIA.código

DOCENTE.número de legajo debe estar en DICTA.número de legajo

MATERIA.código debe estar en DICTA.código

(ENCUESTA.número de legajo, ENCUESTA.código) debe estar en (DICTA.número de legajo, DICTA.código)

(DICTA.número de legajo, DICTA.código) puede no estar en (ENCUESTA.número de legajo, ENCUESTA.código)

*Pregunta... ¿La FK de encuesta podría apuntar directamente a MATERIA y DOCENTE en lugar de apuntar a DICTA?*

# Bibliografía

- *Elmasri R. & Navathe S. (2016) Fundamentals of Database Systems (7ma. Ed.) Pearson, Cap. 5*
- *Silberschatz A., Korth H. & Sudarshan S.(2020) Database System Concepts (7ma. Ed.), Mc.Graw Hill, Cap. 2*
- *Teorey, Yang & Fry (1986) A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model. ACM*

# Dudas



# Muchas gracias!