Tema 3: Modelo Relacional

MODELADO DE DATOS

M. Colebrook Santamaría

Objetivos

- Estructura del modelo relacional
- Dominio, atributo, tupla y relación
- Restricciones
- Valores nulos
- Algoritmo de transformación del modelo ER al modelo relacional
- Grafo relacional

Introducción (1)

- El **modelo de datos relacional** fue introducido por Edgar F. Codd a finales de los sesenta.
- El modelo utiliza el concepto de **relación matemática**, que tiene apariencia muy similar a una **tabla**, como su bloque de construcción básico, y tiene sus bases teóricas en la **teoría de conjuntos** y la **lógica de predicados** de primer orden.
- El objetivo fundamental del modelo es mantener la **independencia** de esta estructura lógica respecto al modo de **almacenamiento** y a otras características de tipo **físico**.

Introducción (2)

- Los objetivos de forma más específica son:
  - **Independencia física**: el modo cómo se almacenan los datos no debe influir en su manipulación lógica y, por tanto, los usuarios que acceden a esos datos no han de modificar sus programas por cambios en el almacenamiento físico.
  - **Independencia lógica**: añadir, eliminar o modificar cualquier elemento de la BD no debe repercutir en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).
  - **Flexibilidad**: en el sentido de poder ofrecer a cada usuario los datos de la forma más adecuada a la correspondiente aplicación.
  - **Uniformidad**: las estructuras lógicas de los datos presentan un aspecto uniforme (tablas), lo que facilita la concepción y manipulación de la base de datos por parte de los usuarios.
  - **Sencillez**: el modelo de datos relacional debe ser fácil de comprender y de utilizar por parte del usuario final.

Estructura del modelo relacional (1)

NOMBRE			
Atributo 1	**Atributo 2**	**...**	**Atributo *n***
XXX	XXX	...	XXX
XXX	XXX	...	XXX
...	...	...	...
XXX	XXX	...	XXX

→ tupla 1

→ tupla 2

...

→ tupla *m*

Estructura del modelo relacional (2)

- En la tabla anterior podemos distinguir:
  - **Nombre** de la relación.
  - **Atributos**: conjunto de columnas que representan propiedades de la tabla y que también están caracterizadas por su nombre.
  - **Tuplas**: conjunto de filas que contienen los valores que toma cada uno de los atributos para cada elemento de la relación.
  - **Grado**: número de atributos (*n*).
  - **Cardinalidad**: número de tuplas (*m*).
  - **Dominios**: donde los atributos toman sus valores.
- Las relaciones se pueden representar como tablas, pero **no son tablas**:
  - **No** pueden tener tuplas **duplicadas**.
  - El **orden** de las tuplas o los atributos es irrelevante.
  - Los valores de los atributos son **atómicos** (no multivaluados).

Comparación entre Relación, Tabla y Fichero

RELACIÓN	TABLA	FICHERO
Tupla	Fila	Registro
Atributo	Columna	Campo
Grado	# de columnas	# de campos
Cardinalidad	# de filas	# de registros

Dominio y Atributo

- Un **dominio *D*** es un conjunto finito de valores **homogéneos** (todos del mismo tipo) y **atómicos** (indivisibles) V_1, V_2, ..., V_n caracterizado por su nombre y por su tipo de datos.
- Ejemplos de dominios:
  - Número de teléfono: 9 dígitos.
  - Nombres: conjunto de caracteres.
  - Edad: número entero entre 18 y 100 (?).
  - Nacionalidades: Española, Francesa, Británica, etc.
  - Nombres de departamentos: Ventas, Marketing, Contabilidad, etc.
- Un **atributo** es el papel que tiene un determinado dominio *D* en una relación. Se dice que *D* es el dominio del atributo *A* y se denota por **dom(*A*)**.
- Un **dominio compuesto** es una combinación de dominios simples a la que se puede aplicar ciertas restricciones de integridad. Por ejemplo, Fecha se puede considerar un dominio compuesto de Día, Mes y Año al que se le aplican las adecuadas restricciones de integridad.

Definición formal de Relación (1)

- Una **relación *R*** definida sobre los dominios D_1,D_2,…,D_n (no necesariamente todos distintos), es un subconjunto del producto cartesiano de estos dominios, donde cada elemento de la relación (*tupla*) es una serie de *n* valores ordenados. Consta de:
  - **Nombre**: las relaciones se identifican con un nombre.
  - **Cabecera de relación**: conjunto de *n* pares atributo-dominio subyacente $\{(A_i : D_i)\}$, i=1,..., n donde n es el grado. Se corresponde con la primera fila cuando la relación se percibe como una tabla. El conjunto *A* de atributos sobre los que se define la relación se llama **contexto** de la misma.
  - **Cuerpo de la relación**: conjunto de *m* tuplas $\{t_1, t_2, ..., t_m\}$ donde cada tupla es un conjunto de *n* pares atributo-valor $\{(A_i : V_{ij})\}$ siendo V_{ij} el valor *j* del dominio D_i asociado al atributo A_i, y *m* es la **cardinalidad**.

Definición formal de Relación (2)

- El **esquema de relación** estará constituido por el nombre *R* (si existe) y la cabecera, denotándose:

$$R\ (\{\ A_i : D_i\ \},\ i=1,...,n)$$

- El esquema de relación que representa la parte estática se denomina **intensidad (intensión)**. Se corresponde con lo que hemos llamado tipo de entidad en el modelo ER.
- El **estado de relación**, al que denominaremos simplemente relación (también se denomina **extensión** u **ocurrencia** de relación), está constituido por el esquema y el cuerpo de relación:

<esquema, cuerpo>

siendo el **cuerpo el conjunto de tuplas** que, en un instante dado, satisface el correspondiente esquema de relación.

Definición formal de Relación (3)

- Esquema de Relación (*intensidad, definición*):

  AUTOR(**Nombre**: Nombres, **Nacionalidad**: Nacionalidades,
  Institución: Instituciones)

- Relación (*extensión, estado u ocurrencia*):

AUTOR

Nombre	Nacionalidad	Institución
Date, C.J.	Norteamericana	Relational Institute
Saltor, F.	Española	UPC
Ceri, S.	Italiana	Politécnico de Milán

- Una **BD relacional** es una colección de relaciones que varían en el tiempo.

Clases de Relación (1)

- **Persistentes**: son aquellas cuyo esquema de relación permanece en la BD, borrándose solamente mediante una acción explícita del usuario. Se dividen en:
  - **Relaciones base** (se corresponden con el **nivel conceptual** de la arquitectura **ANSI**): existen por sí mismas, y se crean especificando explícitamente su esquema de relación (nombre y conjunto de pares: atributo/dominio). Sus extensiones, al igual que su definición, también se encuentran almacenados.
  - **Vistas** (se corresponden con el **nivel externo** de la arquitectura **ANSI**): son relaciones derivadas que se definen dando un nombre a una expresión de consulta. Son como relaciones "virtuales", ya que no tienen datos almacenados, sino que lo único que se almacena es su definición en términos de otras relaciones con nombre, las cuales pueden ser relaciones base, otras vistas o instantáneas.

Clases de Relación (2)

- **Instantáneas** (*snapshots*, se corresponden con el **nivel interno** de la arquitectura **ANSI**). Son relaciones derivadas (se definen en términos de otras relaciones), pero tiene datos propios almacenados, los cuales son el resultado de ejecutar la consulta o de guardar una relación base. Se suelen denominar **vistas materializadas**. No se actualizan cuando cambian los datos, pero se “refrescan” cada cierto tiempo, de acuerdo con lo indicado por el usuario en el momento de su creación. Son de **solo lectura**, no pudiendo ser actualizadas por el usuario, sino únicamente “refrescadas” por el sistema.

- **Temporales**: desaparece de la BD en un cierto momento sin necesidad de una acción de borrado específica del usuario (por ejemplo, al terminar una sesión o transacción). Suele ser una relación autónoma (no se deriva de otra), pero también podría ser una vista o una instantánea.

Claves

- Una **clave candidata** de una relación es un conjunto de atributos que identifican unívoca y mínimamente cada tupla de la relación.
- Pregunta: ¿existe siempre una clave candidata en toda relación? ¿por qué?
- Una relación puede tener más de una clave candidata:
  - **Clave primaria**: es aquella clave candidata que el usuario elegirá, por consideraciones ajenas al modelo relacional, para identificar las tuplas de la relación. Cuando solo existe una clave candidata, ésta será la clave primaria.
  - **Claves alternativas**: son aquellas claves candidatas que no han sido elegidas como clave primaria.
- Se denomina **clave ajena** de una relación R2 a un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave candidata de una relación R1, y ambas definidas sobre el **mismo dominio**.

Restricciones

- En el modelo relacional, existen **restricciones** (estructuras u ocurrencias no permitidas), siendo preciso distinguir entre:
  - **Restricciones inherentes (implícitas)**: los datos almacenados en la BD deben adaptarse a las estructuras impuestas por el modelo. Por ejemplo, no tener tuplas duplicadas.
  - **Restricciones semánticas (reglas de negocio o basadas en app)**: han de cumplir las restricciones de usuario a fin de constituir una ocurrencia válida del esquema.
- Elmasri y Navathe (2004) también destacan las restricciones que se pueden expresar directamente en el esquema del modelo de datos usando el LDD, y que denominan **restricciones basadas en esquema** o **explícitas**.

Restricciones inherentes

- De la definición matemática de relación, se deduce inmediatamente una serie de características propias de una relación que se han de cumplir obligatoriamente, por lo que se trata de restricciones inherentes y que diferencian una relación de una tabla:
  - No hay dos **tuplas iguales**: de donde se deduce la **obligatoriedad** de la clave primaria.
  - El **orden de las tuplas y de los atributos** no es significativo.
  - Cada atributo solo puede tomar **un único valor** del dominio sobre el que está definido, no admitiéndose por tanto los grupos repetitivos. Se dice que una tabla que cumple esta condición está normalizada en 1ª Forma Normal (1FN).
  - **Regla de integridad de entidad**: ningún atributo que forme parte de la clave primaria de una relación puede tomar un valor nulo.

Restricciones semánticas (RS)

- Dentro del modelo relacional, como en otros modelos de datos, existen **restricciones semánticas** o de usuario, que son facilidades que el modelo ofrece a los usuarios a fin de que éstos puedan reflejar lo más fielmente posible en el esquema la semántica del mundo real.
- Sin embargo, estas restricciones semánticas del modelo relacional, **no son suficientes** para captar toda la semántica del universo del discurso que se está tratando de modelar.
- Por ello, los productos añaden ciertas facilidades que permiten “programarlas”, aunque siempre podrían incluirse en una aplicación con sentencias de manipulación embebidas.

RS: Clave Primaria

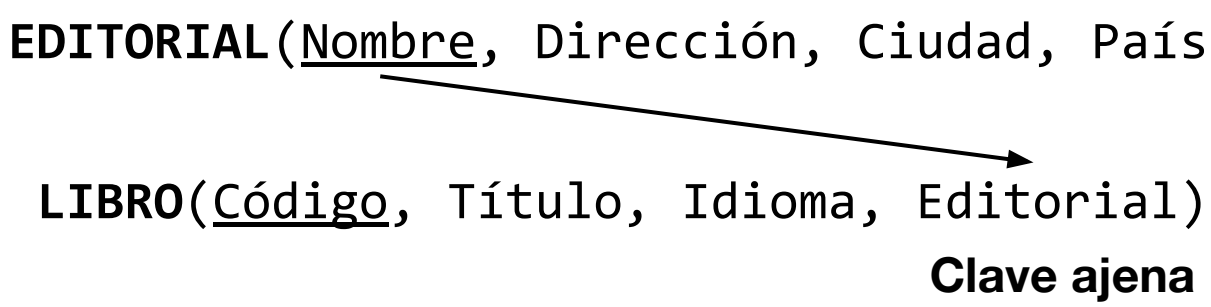
- **Clave primaria** (`PRIMARY KEY`): permite declarar un atributo o un conjunto de atributos como clave primaria de una relación por lo que **sus valores no se podrán repetir ni se admitirán los nulos** (o valores ausentes).
- La obligatoriedad de la clave primaria es un **restricción inherente del modelo relacional**.
- Sin embargo, la declaración de un atributo (o atributos) como clave primaria de una relación es una **restricción semántica** que responde a la necesidad del usuario de imponer que los valores del conjunto de atributos que constituyen la clave primaria no se repitan en la relación ni tampoco tomen valores nulos.
- Ni el SQL ni los productos relacionales **obligan** a la declaración de una clave primaria para cada relación. No imponen la restricción inherente de que no existan tuplas duplicadas.

RS: Unicidad y Obligatoriedad

- **Unicidad** (`UNIQUE`): mediante la cual se indica que los valores de un conjunto de atributos (uno o más) **no pueden repetirse** en una relación. Esta restricción permite la definición de **claves alternativas**.
- **Obligatoriedad** (`NOT NULL`): se indica que uno o más atributos no admiten valores nulos.

RS: Integridad Referencial (1)

- **Integridad referencial** (`FOREIGN KEY`): si una relación R2 (relación que referencia) tiene un descriptor que es una clave candidata de la relación R1 (relación referenciada), todo valor de dicho descriptor debe, o bien **concordar con un valor de la clave candidata** referenciada de R1, o bien ser **nulo**.
- El descriptor es, por tanto, una **clave ajena** de la relación R2.
- Las relaciones R1 y R2 no son necesariamente distintas.
- Además, cabe destacar que la clave ajena puede ser también parte (o la totalidad) de la clave primaria de R2.
- Ejemplo:



RS: Integridad Referencial (2)

- Además de definir las claves ajenas, hay que determinar las consecuencias que pueden tener ciertas operaciones (borrado **“`ON DELETE`”** y modificación **“`ON UPDATE`”**) realizadas sobre tuplas de la relación referenciada.
- **Operación restringida** (`NO ACTION` o `RESTRICT`):
  - **Modificación**: si cualquier fila de la relación que contiene la clave ajena coincide con el valor de la clave primaria, se rechaza la actualización cuando la regla es `RESTRICT`. Si cualquier fila de la relación que contiene la clave ajena no tiene clave primaria correspondiente cuando se completa la sentencia de actualización, se rechaza cuando la regla es `NO ACTION`.
  - **Borrado**: con `RESTRICT` o `NO ACTION` produce un error y no se suprime ninguna tupla.
  - `NO ACTION` es la opción válida por **defecto**.

RS: Integridad Referencial (3)

- **Operación con transmisión en cascada** (`CASCADE`): el **borrado** (o **modificación**) de tuplas de la relación que contiene la clave candidata referenciada lleva consigo el **borrado** (o **modificación**) en cascada de las tuplas de la relación que contiene la clave ajena.
- En nuestro ejemplo, equivaldría a decir que, al modificar el nombre de una editorial en la relación `EDITORIAL`, se modificaría **automáticamente** dicho nombre en todos los libros de la BD publicados por dicha editorial. Análogamente ocurriría en el caso del borrado de la clave referenciada.

RS: Integridad Referencial (4)

- **Operación con puesta a nulos** (`SET NULL`): el **borrado** (o la **modificación**) de tuplas de la relación que contiene la clave candidata referenciada lleva consigo poner a **nulos** los valores de las claves ajenas de la relación que referencia.
- Esto llevaría a que cuando se **borra** una editorial, todos los libros que ha publicado dicha editorial y que se encuentran en la relación `LIBROS` se les pone automáticamente a **nulo** el atributo `Nombre`.
- Esta opción, obviamente, solo es posible cuando el atributo que es clave ajena **admite valores nulos**.

RS: Integridad Referencial (5)

- **Operación con puesta a valor por defecto** (`SET DEFAULT`): el **borrado** de tuplas de la relación que contiene la clave candidata referenciada (o la **modificación** de dicha clave) lleva consigo poner el **valor por defecto** a la clave ajena de la relación que referencia.
- Además de todas estas opciones, existen otras restricciones que podríamos denominar de **rechazo**, en las que el usuario formula una condición mediante un predicado definido sobre un conjunto de atributos, de tuplas o dominios, el cual debe ser verificado por los correspondientes objetos en toda operación de actualización para que el nuevo estado constituya una ocurrencia válida del esquema.
- En caso de que la operación intente violar la condición, se impide que la operación se lleve a cabo (**rechazo** de la operación).

RS: Integridad Referencial (6)

- En el modelo relacional (más bien, en SQL) se pueden distinguir dos **restricciones de rechazo** distintas, según la condición afecte a un único elemento de la base de datos (por ejemplo, a una relación) o más de uno:
  - **Verificación** (`CHECK`): comprueba en toda operación de actualización si el predicado es cierto o falso, y en caso negativo, rechaza la operación. Se define sobre un único elemento (incluyéndose en la definición de dicho elemento) y puede o no tener nombre.
  - **Aserción** (`ASSERTION`): actúa de forma idéntica a la anterior, pero se diferencia de ella en que puede afectar a varios elementos (dos relaciones distintas). Se define con un nombre, ya que es un elemento más del esquema que tiene existencia por sí mismo.
  - **Disparador** (`TRIGGER`): nos permite indicar una condición y la acción a realizar si es verdadera (reglas evento-condición-acción). Estos **disparadores** (*triggers*) permiten definir restricciones en las que el usuario pueda especificar libremente la respuesta (acción) ante una determinada condición.

RS: Integridad Referencial (7)

- Las restricciones de rechazo que se definen sobre un solo dominio, relación o atributo, se denominan **intraelemento**, y las que se definen sobre varios elementos se dice que son restricciones **interelementos**.
- Las anteriores reglas de integridad son **declarativas**, pero los disparadores son **procedimentales**, siendo preciso que el usuario escriba el procedimiento que ha de aplicarse en caso de que se cumpla la condición.
- También es importante en una restricción el momento en el que ésta se verifica dentro de una transacción:
  - **Inmediato**: la restricción se verificará al finalizar cada sentencia.
  - **Diferido**: se verificará al finalizar la transacción.

Esquema de Relación vs. Esquema Relacional

- Esquema de Relación:

  $$\mathbf{R} < \mathbf{A}:\mathbf{D}, \mathbf{S}>$$

  R: nombre de la relación.

  A: lista de atributos.

  D: dominios sobre los que están definidos los atributos.

  S: restricciones de integridad (*intraelementos*).

- Esquema de la BD relacional:

  $$\mathbf{E} < \{\mathbf{R_i}\}, \{\mathbf{I_i}\} >$$

  E: nombre del esquema relacional.

  $\{\mathbf{R_i}\}$: conjunto de esquemas de relación.

  $\{\mathbf{I_i}\}$: conjunto de restricciones de integridad (*interelementos*).

- Podemos definir una BD Relacional como un **esquema relacional** junto con una **ocurrencia válida** de dicho esquema.

Valores nulos (1)

- Se define un valor **nulo** (o ausente) a una *señal* utilizada para representar información desconocida, inaplicable, inexistente, no válida, no proporcionada, indefinida.
- Codd (1990), propone abandonar el término valor nulo y sustituirlo por el de **marca**, ya que:
  - Los SGBD Relacionales no deberían tratar las marcas como si fueran cualquier otro valor.
  - Al introducir la lógica tetravaluada, se desdobla el concepto de valor nulo.
  - Algunos lenguajes anfitriones tratan objetos que denominan “nulos” pero con diferente significado al de las marcas.
  - Es más fácil decir “marcado” y “sin marcar”, que “anulado”, “nulificado”, etc.

Valores nulos (2)

- La **necesidad de los valores nulos** o marcas en las BD es evidente por diversas razones:
  - Crear tuplas (filas) con ciertos atributos desconocidos en ese momento, por ejemplo, el año de edición de un libro.
  - Añadir un nuevo atributo a una relación existente, el cual, al añadirse, no tendría ningún valor para las tuplas existentes de antemano en la relación.
  - Incluir atributos inaplicables a ciertas tuplas, por ejemplo, la profesión de un menor.
- Sin embargo, el tratamiento de valores nulos exige definir:
  - Operaciones de **comparación**: problema de saber si dos valores nulos son iguales o no.
  - Operaciones **aritméticas**: se considera nulo el resultado de sumar, restar, multiplicar o dividir algún valor nulo.
  - Operaciones **algebraicas**.
  - Funciones de **agregación**.

Valores nulos (3)

- Por tanto, en los operadores de comparación, no podemos afirmar que es **cierto (C)** que dos valores nulos son iguales puesto que estaríamos afirmando que no son “tan” desconocidos.
- Pero tampoco podemos decir que es **falso (F)** que sean iguales.
- La única solución es decir que **quizás (Q)** son iguales.
- Surge una lógica distinta a la habitual (L2V), denominada **lógica trivaluada (L3V)**.

AND	**C**	**Q**	**F**
C	C	Q	F
Q	Q	Q	F
F	F	F	F

OR	**C**	**Q**	**F**
C	C	C	C
Q	C	Q	Q
F	C	Q	F

NOT	
C	F
Q	Q
F	C

C: Cierto, **F**: Falso, **Q**: Quizás

Valores nulos (4)

- Date (1995) señala que también es preciso introducir dos nuevos operadores:
  - **ES_NULO** (*IS_NULL*): que toma el valor cierto si el operando es nulo y falso en caso contrario.
  - **SI_NULO** (*IF_NULL*): que se aplica a dos operandos y devuelve el valor del primero, salvo que sea nulo, en cuyo caso devuelve el valor del segundo.
- Además, dos tuplas se consideran **duplicadas** si, atributo a atributo ambos son iguales y no nulos o ambos nulos.

Problemas de lógica trivaluada y valores por defecto

- Según Date (1995), la lógica trivaluada puede causar *problemas psicológicos* a los usuarios porque *atenta contra la intuición*. Por ejemplo, si un usuario realizara las siguientes consultas:

  "Libros editados en 1995", "Libros editados antes de 1995", "Libros editados después de 1995"

  tendría que haber recuperado todos los libros de la BD.
- Pero esto no es así, ya no habría recuperado los libros cuyo **año de edición se desconoce**, es decir, el año de edición es **nulo**.
- Algunos expertos recomiendan **evitar los valores nulos** mediante la utilización de **valores por defecto**. Pero este enfoque también ha sido duramente criticado por Codd (1990).
- La **mejor recomendación** es diseñar BD evitando, en la medida de lo posible, los valores nulos (sobre todo en el paso de ER a relacional), y especificar siempre que se pueda la obligatoriedad de los atributos (**`NOT NULL`**).

Lógica tetravaluada (1)

- La lógica tetravaluada surge de la necesidad de diferenciar dos tipos importantes de valores nulos:
  - **Inaplicables (‘i’)**: es decir, no tienen sentido. Por ejemplo, la profesión de un menor.
  - **Desconocidos aplicables (‘a’)**: momentáneamente son desconocidos pero deberían existir y puede que lleguen a conocerse. Por ejemplo, el año de edición de un libro.
- En este caso, las operaciones aritméticas quedan modificadas como sigue, siendo **&** un **operador aritmético** (suma, resta, división y producto), y **“x”** un **valor no nulo** de la BD:
  - x & a = a & x = a
  - x & i = i & x = i
  - a & i = i & a = i
  - a & a = a
  - i & i = i

Lógica tetravaluada (2)

- Tabla de verdad de la lógica tetravaluada según Codd (1990)

AND	C	A	F	I
C	C	A	F	I
A	A	A	F	I
F	F	F	F	F
I	I	I	F	I

OR	C	A	F	I
C	C	C	C	C
A	C	A	A	A
F	C	A	F	F
I	C	A	F	I

NOT	
C	F
A	A
F	C
I	I

- Tabla de verdad de la lógica tetravaluada según Gessert (1990)

AND	C	A	F	I
C	C	A	F	I
A	A	A	F	I
F	F	F	F	F
I	I	I	I	I

OR	C	A	F	I
C	C	C	C	C
A	C	A	A	A
F	C	A	F	F
I	C	A	F	I

NOT	
C	F
A	A
F	C
I	I

INV	
C	I
A	F
F	A
I	C

Lógica tetravaluada (3)

- Veamos con un ejemplo las consecuencias de emplear estas tablas de verdad en las siguiente consulta sobre una tabla de DOCUMENTOS, que pueden ser **artículos o libros**:

```
(IDIOMA = “INGLÉS”) OR (EDITORIAL = “RAMA”)
```

a. Si se trata de un artículo en italiano, el primer término se evaluaría como **falso**, mientras que el segundo sería **inaplicable** (al carecer de sentido que un artículo tenga editorial), siendo el resultado **falso** para Codd y Gessert.
b. Si se trata de un libro en italiano cuya editorial desconocemos, daría como resultado “aplicable” en ambos casos, al ser el primer término **falso** y el segundo **aplicable**.

Lógica tetravaluada (4)

- Sin embargo, para la consulta:

  `(IDIOMA = “INGLÉS) AND (EDITORIAL = “RAMA”)`

  a. Si se trata de un artículo en italiano, el primer término se evaluaría como **falso**, mientras que el segundo sería **inaplicable**, siendo el resultado **falso** para Codd e **inaplicable** para Gessert.
  b. Si se trata de un libro en italiano, daría como resultado **falso** para ambos.

Transformación del Modelo ER al Relacional

- Las tres reglas básicas para convertir un esquema ER al modelo relacional son:
  a. Todo tipo de **entidad** se convierte en una **relación (tabla)**.
  b. Todo tipo de **relación N:M** se transforma en una **relación**.
  c. Para todo tipo de **relación 1:N** se realiza la **propagación de clave** (regla general), o se crea una nueva **relación**.
- Las reglas más específicas implican la transformación de:
  a. Dominios, entidades, atributos normales y derivados
  b. Relaciones M:N, 1:N, y 1:1
  c. Atributos de relaciones
  d. Restricciones
  e. Dependencias en existencia y en identificación
  f. Restricciones de relaciones (inclusión, exclusión, etc)
  g. Tipos y subtipos
  h. Dimensión temporal

Grafo relacional (1)

- Una forma de representar el esquema relacional de forma sencilla y completa es el denominado **grafo relacional**, que es un grafo dirigido donde:
  - **Nodos**: relaciones (tablas) de la BD.
  - **Arcos**: restricciones de clave ajena.
- Convenciones:
  - Nombre de las tablas en **MAYÚSCULAS** y en **NEGRITA**.
  - Atributos separados por comas entre paréntesis con la primera letra en Mayúsculas.
  - Claves primarias <u>subrayadas</u>.
  - Claves ajenas en *cursiva*, y referencian a la relación en la que son clave primaria mediante una flecha.
  - Atributos con valores nulos se marcan con un asterisco (*).

Grafo relacional (2)

- Las opciones de integridad referencial para el **BORRADO** (`ON DELETE ...`) son:
  - **B:C** = Borrado en Cascada (`CASCADE`).
  - **B:N** = Borrado con puesta a Nulos (`SET NULL`).
  - **B:D** = Borrado con puesta a valor por Defecto (`SET DEFAULT`).
  - **B:R** = Borrado Restringido (`RESTRICT`).
- Las opciones de integridad referencial para la **MODIFICACIÓN** (`ON UPDATE ...`) son:
  - **M:C** = Modificación en Cascada (`CASCADE`).
  - **M:N** = Modificación con puesta a Nulos (`SET NULL`).
  - **M:D** = Modificación con puesta a valor por Defecto (`SET DEFAULT`).
  - **M:R** = Modificación Restringida (`RESTRICT`).

1. Transformación de dominios

- En el modelo relacional estándar, un dominio es un objeto más, y tendrá su definición concreta en el **LDD** (Lenguaje de Definición de Datos), como puede ser SQL:2003.
- Por ejemplo:

```
CREATE DOMAIN Estados_Civiles AS CHAR(1)
CHECK (VALUE IN (‘S’, ‘C’, ‘V’, ‘D’))
```

2. Transformación de entidades

- Cada tipo de entidad se convierte en una **relación** (**tabla**).
- La tabla se llamará igual que el tipo de entidad de donde proviene.
- En SQL disponemos de la sentencia `CREATE TABLE`.

3. Transformación de atributos de entidades (1)

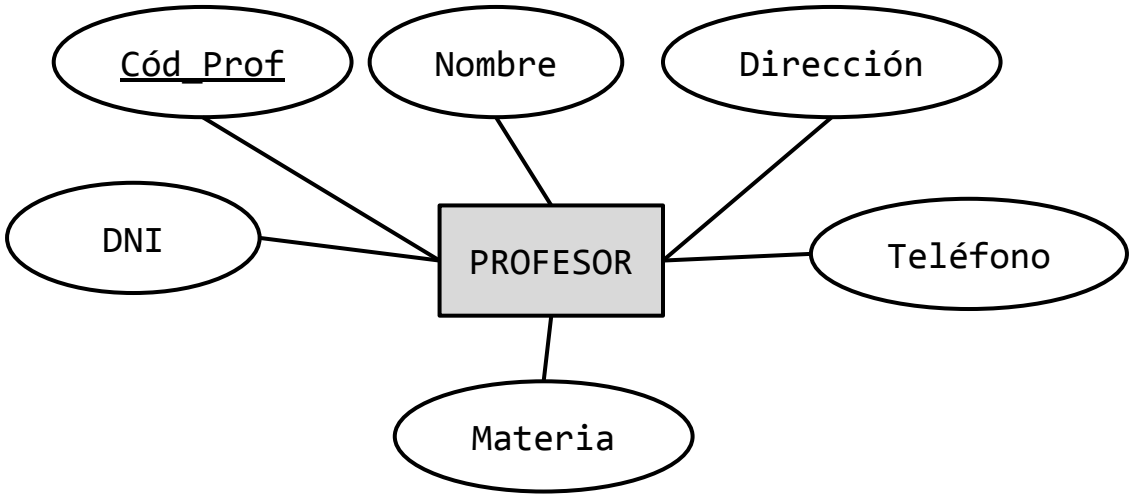
- Cada atributo de una entidad se transforma en una **columna** de la relación a la que ha dado lugar la entidad.
  Dependiendo del tipo de atributo, tenemos que:
  - **Atributos Identificadores Principales (AIP)**: pasan a ser la **clave primaria**. En SQL tenemos la cláusula `PRIMARY KEY`.
  - **Atributos Identificadores Alternativos (AIA)**: se especifican con la cláusula `UNIQUE`. Si queremos que no tomen valores nulos, habrá que indicarlo.
  - **Atributos no identificadores**: estos atributos pasan a ser **columnas** de la relación, las cuales tienen permitido tomar valores nulos a no ser que se indique lo contrario.

3. Transformación de atributos de entidades (2)

- En este último caso, debemos distinguir entre:
  - **Atributos multivaluados**: dan lugar a una nueva relación cuya clave primaria es la concatenación de la clave primaria de la entidad en la que se sitúa el atributo multivaluado más el nombre del atributo multivaluado.
  - **Atributos obligatorios/opcionales**: los obligatorios tienen la restricción `NOT NULL`, y los opcionales pueden tomar valores nulos.
  - **Atributos compuestos**: se transforman en los atributos que los componen.
  - **Atributos derivados**: tienen un **disparador** (*trigger*) asociado que calcula el valor cada vez que se insertan o borran las ocurrencias de los atributos que intervienen en el cálculo de éste.

3. Transformación de atributos de entidades (3)

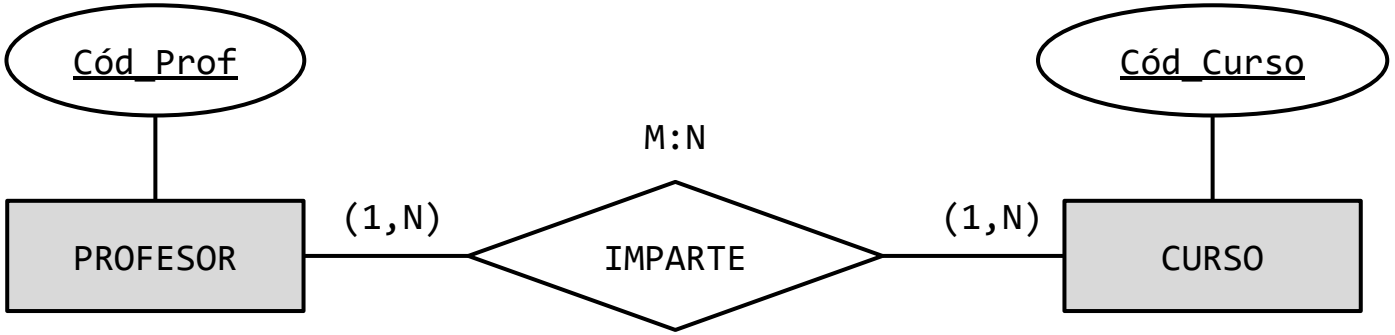
- Ejemplo:



PROFESOR(Cód_Prof, Nombre, DNI, Dirección, Teléfono, Materia)

4. Transformación de relaciones M:N (1)

- Las **relaciones M:N** se transforman en una relación que tendrá como clave primaria la concatenación de los AIP de los tipos de entidad que asocia.



PROFESOR(Cód_Prof, ...)

CURSO(Cód_Curso, ...)

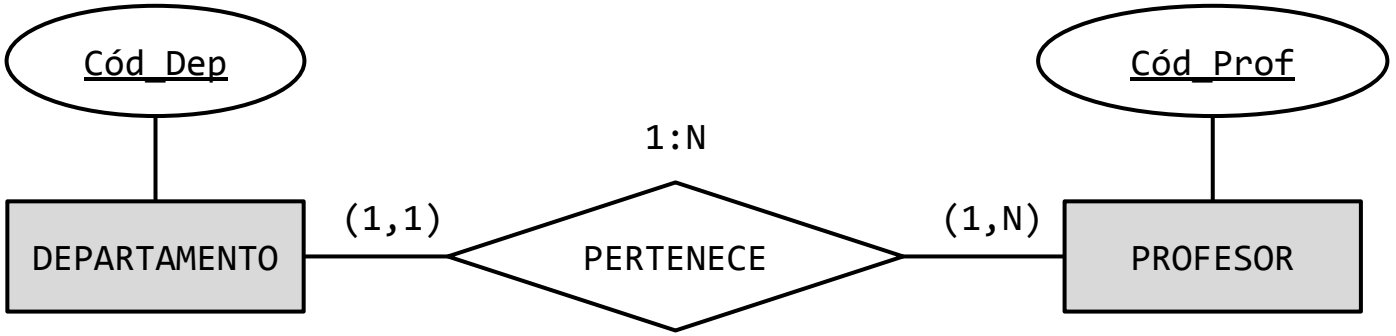
IMPARTE(*Cód_Curso*, *Cód_Prof*, ...)

4. Transformación de relaciones M:N (2)

- No hay forma de diferenciar dentro de un esquema relacional qué relaciones provienen de una entidad y cuáles de ellas proceden de la transformación de relaciones, por lo que hay **cierta pérdida de semántica**.
- Este problema solo se puede resolver almacenando **comentarios** sobre la procedencia de cada una de las tablas.
- Además, cada uno de los atributos que forman la clave primaria de esta relación son **claves ajenas** que referencian a las tablas en las que se han convertido las entidades relacionadas.
- Esto se especifica mediante la cláusula `FOREIGN KEY`.
- Otra característica que debemos recoger en esta transformación es la **cardinalidad mínima** de cada una de las entidades que participan en la relación, lo que se hace mediante la especificación de restricciones (`CHECK`), aserciones (`ASSERT`) o disparadores (`TRIGGER`).

5. Transformación de relaciones 1:N (1)

- **Opción 1 - Propagar los AIP**: del tipo de entidad que tiene cardinalidad máxima 1 a la que tiene N, desapareciendo el nombre de la relación (se pierde semántica).



DEPARTAMENTO(Cód_Dep, ...)

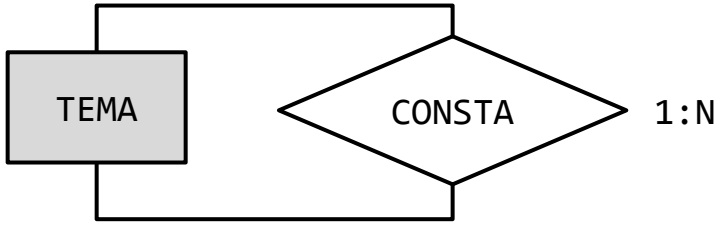
PROFESOR(Cód_Prof, *Cód_Dep*, ...)

5. Transformación de relaciones 1:N (2)

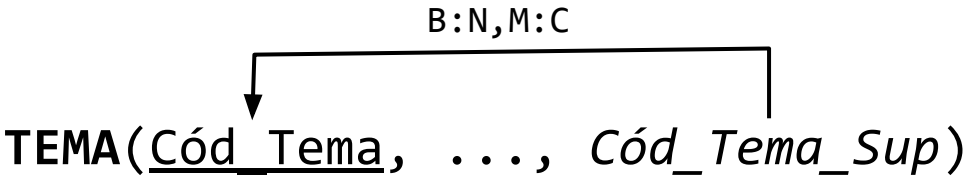
- **Opción 2 - Transformarlo en una relación**: como si se tratara de una relación M:N. Sin embargo, la clave primaria de la relación creada es sólo el AIP de la entidad a la que corresponde la **cardinalidad N**.
- Casos en los que puede ser apropiado transformar la relación son:
  1. Cuando el número de ejemplares relacionados de la entidad que propaga su clave es muy pequeño y, por tanto, existirían muchos valores nulos en la clave propagada.
  2. Cuando se prevé que dicha relación en un futuro se convertirá en una de tipo M:N.
  3. Cuando la relación tiene atributos propios y no deseamos propagarlos (a fin de conservar la semántica).

5. Transformación de relaciones 1:N (3)

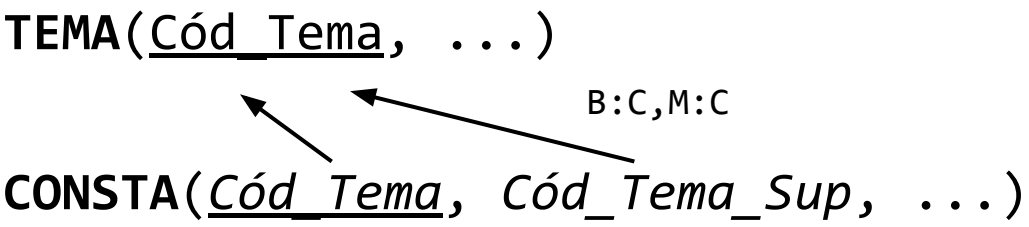
- Ejemplo:



Opción 1:



Opción 2:

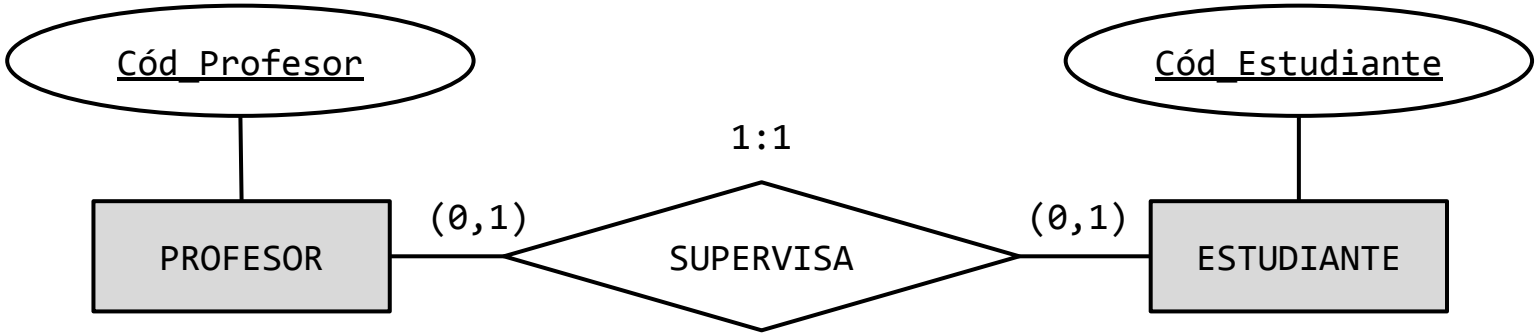


6. Transformación de relaciones 1:1 (1)

- Una relación 1:1 es un caso particular de una M:N o también de una 1:N, en la que no hay regla fija, por lo que podemos aplicar lo visto en el **punto 4** (crear una relación) o en el **punto 5 - Opción 1** (propagar la clave).
- En este último caso, hay que observar que la propagación de la clave puede efectuarse en **ambos sentidos**.
- Los criterios para aplicar una u otra regla se basan en las **cardinalidades mínimas**, en **no perder semántica**, en **evitar valores nulos** o en motivos de **eficiencia**.

6. Transformación de relaciones 1:1 (2)

- Si ambas entidades tienen **cardinalidades (0,1)**, puede ser conveniente transformar la relación 1:1 en una **relación**.



Clave alternativa:
(UNIQUE, NOT NULL)

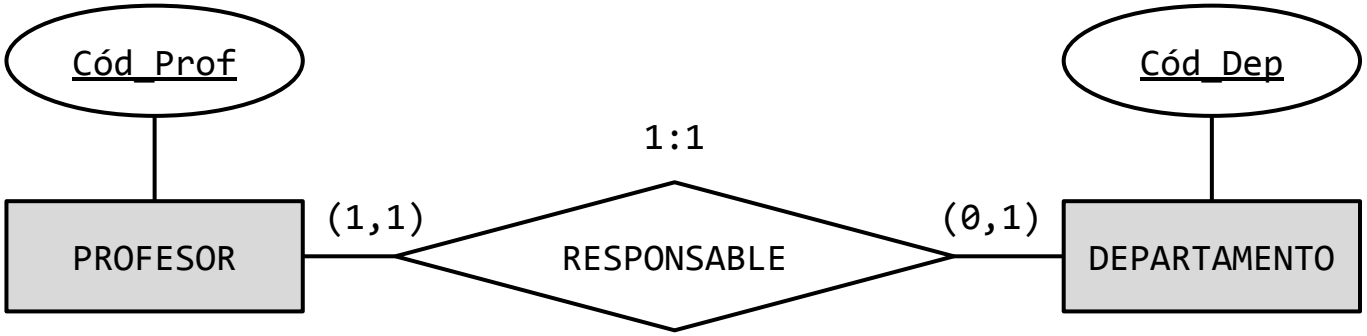
SUPERVISA(*Cód_Profesor*, *Cód_Estudiante*, ...)

PROFESOR(Cód_Profesor, ...)

ESTUDIANTE(Cód_Estudiante, ...)

6. Transformación de relaciones 1:1 (3)

- Si una de las entidades que participa posee **cardinalidad (0,1)**, mientras que la otra tiene **(1,1)**, conviene **propagar la clave** de la entidad con cardinalidades (1,1) a la tabla resultante.



PROFESOR(Cód_Prof, ...)

DEPARTAMENTO(Cód_Dep, ..., *Cód_Prof*)

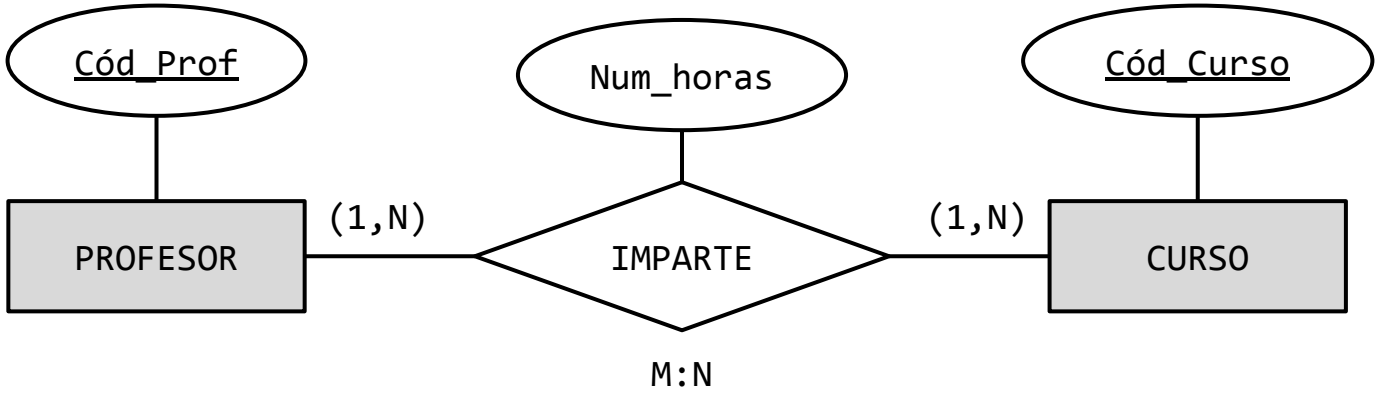
Clave ajena
(NOT NULL)

6. Transformación de relaciones 1:1 (4)

- En el caso de que ambas entidades presenten **cardinalidades (1,1)**, se puede **propagar la clave** de cualquiera de ellas a la tabla resultante de la otra, teniendo en cuenta los accesos más frecuentes y prioritarios a los datos de las tablas.
- Se puede plantear la **propagación de las dos claves**, lo que introduce redundancias que deben ser controladas por medio de restricciones.

7. Transformación de atributos de relaciones

- Si la (inter)relación se transforma en relación (tabla), **todos sus atributos pasan a ser columnas** de la relación, incluso en el caso de propagación de clave.



PROFESOR(Cód_Prof, ...)

IMPARTE(*Cód_Prof*, *Cód_Curso*, Núm_horas)

CURSO(Cód_Curso, ...)

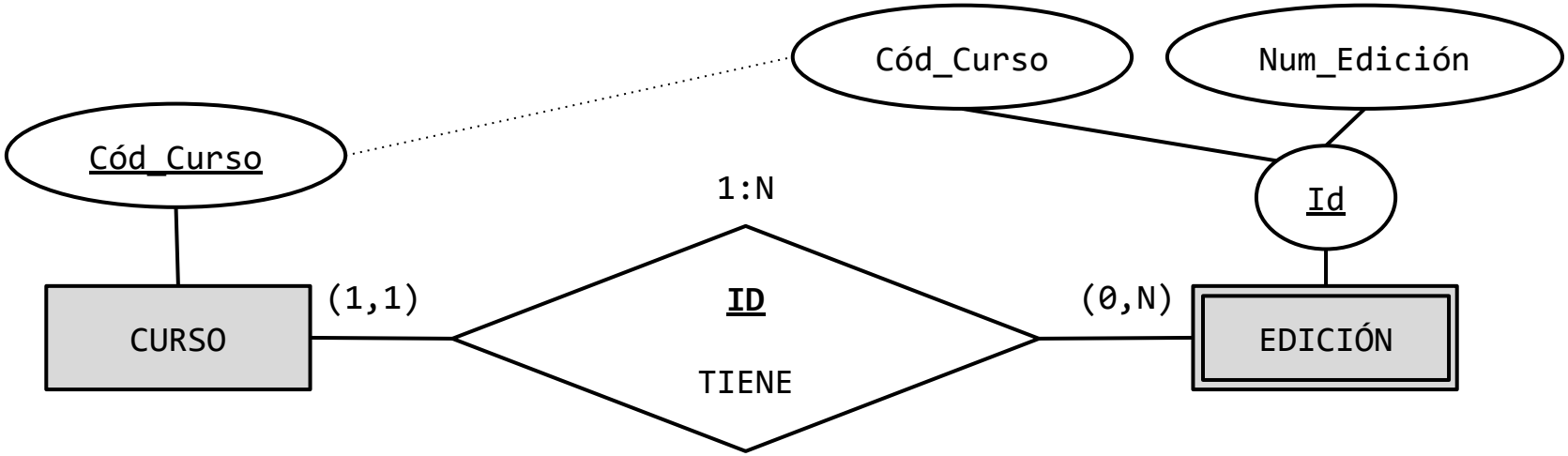
8. Transformación de restricciones

- Las restricciones de usuario se pueden recoger en el modelo relacional usando **cláusulas especiales** (del tipo `BETWEEN` o `IN` para los dominios), o bien a través de la cláusula **`CHECK`** dentro de la descripción de la tabla.
- También se pueden usar **aserciones** (`ASSERT`) en el caso de que la comprobación afecte a atributos de más de una tabla.
- Además, existe la posibilidad de usar **disparadores** (`TRIGGER`), aunque no se ofrecen en todos los productos de SGBD.

9. Transformación de dependencias en identificación y en existencia (1)

- El mecanismo usado para transformar una relación de estos dos tipos es el de **propagación de clave**, creando una **clave ajena**, con **nulos no permitidos**, en la relación de la entidad dependiente, y con la característica de obligar a una **modificación y borrado en cascada**.
- En el caso caso de **dependencia en identificación** la **clave primaria** de la relación en la que se ha transformado la entidad débil debe estar formada por la **concatenación de las claves** participantes en la relación del ER.

9. Transformación de dependencias en identificación y en existencia (2)



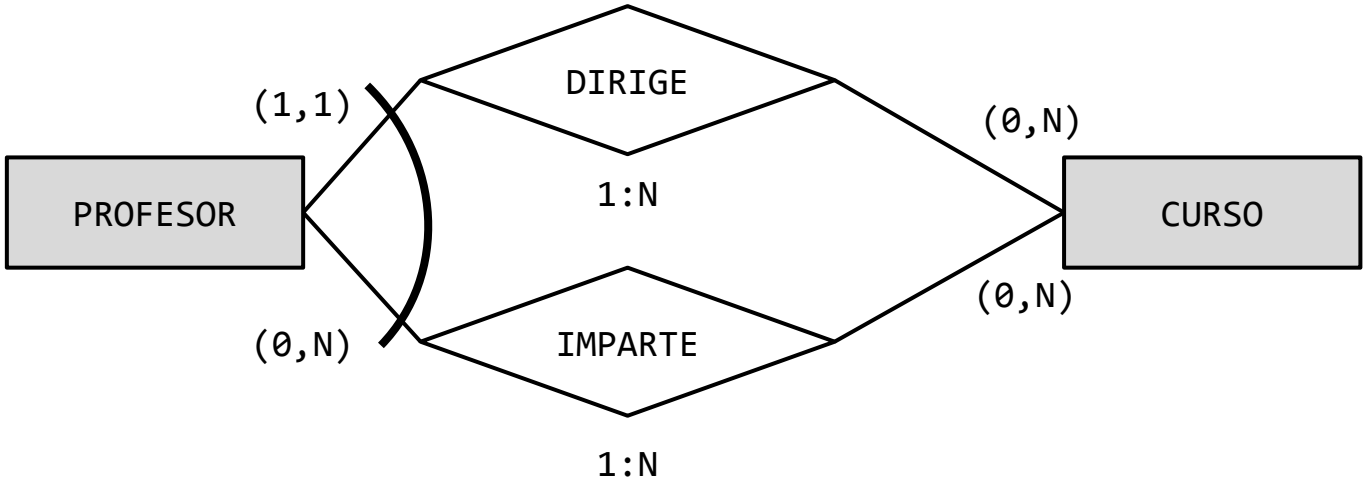
CURSO(Cód_Curso, ...)

EDICIÓN(Num_Edición, *Cód_Curso*, ...)

Clave ajena
B:C,M:C

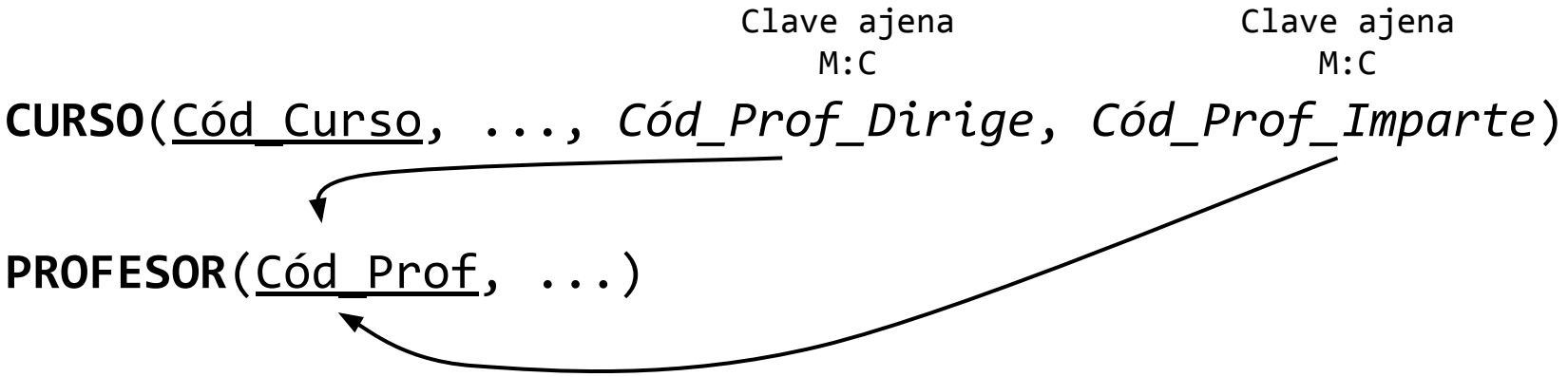
10. Transformación de restricciones de relaciones (1)

- Para poder soportar restricciones de relaciones (exclusión, inclusión, exclusividad, inclusividad) debemos definir las **restricciones pertinentes** en cada caso usando, por ejemplo, la cláusula `CHECK`.



- En este caso, podemos resolver las relaciones `DIRIGE` e `IMPARTE` propagando la clave de `PROFESOR` a `CURSO` en forma de `Cód_Prof_Dirige` y `Cód_Prof_Imparte`.

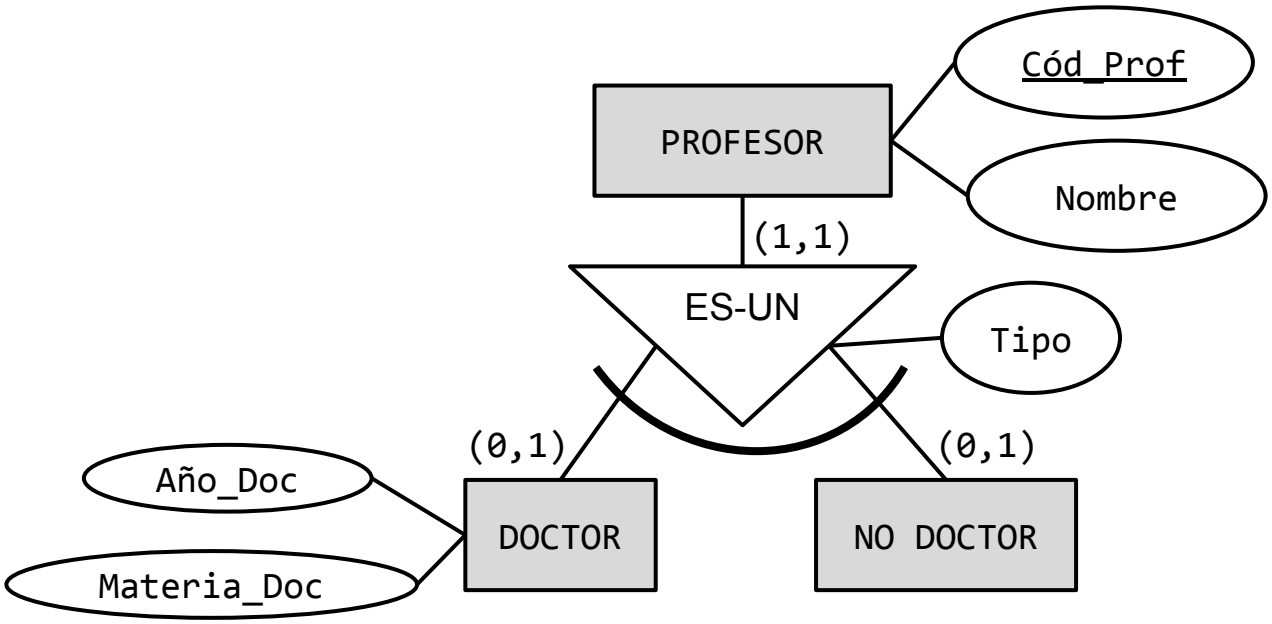
10. Transformación de restricciones de relaciones (2)




```
CHECK ((Cod_Prof_Dirige IS NULL AND
        Cod_Prof_Imparte IS NOT NULL)
     OR
      (Cod_Prof_Dirige IS NOT NULL AND
       Cod_Prof_Imparte IS NULL))
```

11. Transformación de tipos y subtipos (1)

- **Opción 1**: englobar todos los atributos de la entidad y sus subtipos en una sola relación. Se adoptará esta solución cuando los subtipos se diferencien en muy pocos atributos, y las relaciones que los asocian al resto de entidades sean las mismas para todos los subtipos.



PROFESOR(Cód_Prof, Nombre, ..., Tipo, Año_Doc, Materia_Doc)

11. Transformación de tipos y subtipos (2)

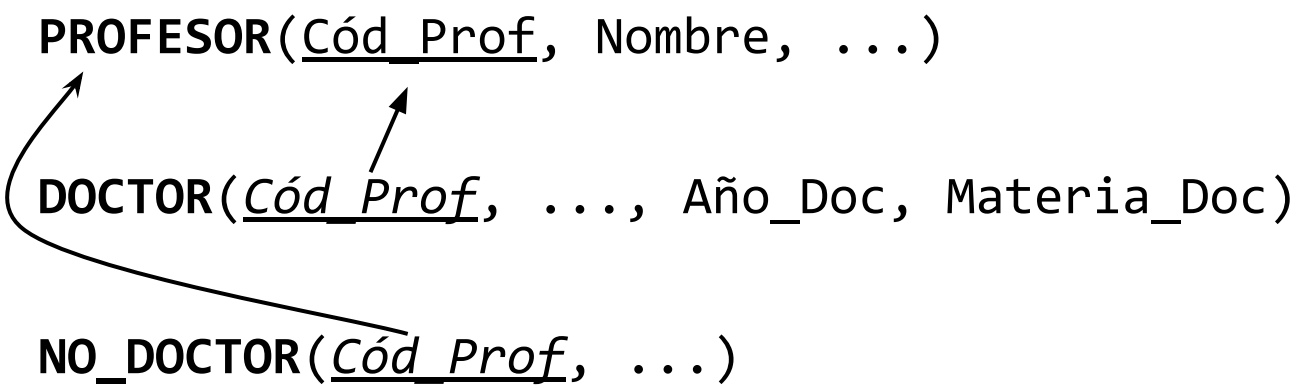
- También habrá que especificar las restricciones semánticas correspondientes:

```
CHECK ((Tipo = “NO_DOCTOR” AND Año_Doc IS NULL AND
        Materia_Doc IS NULL)
    OR (Tipo = “DOCTOR” AND Año_Doc IS NOT NULL AND
        Materia_Doc IS NOT NULL))
```

- Hay que observar que:
  - El atributo discriminante (`Tipo`) podrá **admitir valores nulos** en el caso de jerarquía **parcial**, y deberá declararse como `NOT NULL` si la jerarquía es **total**.
  - Si los subtipos se solapan, el atributo discriminante constituirá un grupo repetitivo, por lo que habrá que separarlo en una relación aparte.
- En cuanto a eficiencia, el **acceso a una fila** que refleje toda la información de una determinada entidad es muy rápido.

11. Transformación de tipos y subtipos (3)

- **Opción 2**: crear una relación para el supertipo y tantas relaciones como subtipos hayan, con sus atributos correspondientes. Ésta es la solución adecuada cuando existen muchos atributos distintos entre los subtipos y se quieren mantener de todas maneras los atributos comunes a todos ellos en una relación. Además, habrá que incluir las restricciones y/o aserciones oportunas.



- Es la opción **menos eficiente**, aunque es la **mejor** desde un punto de vista exclusivamente **semántico**.

11. Transformación de tipos y subtipos (4)

- **Opción 3**: considerar relaciones distintas para cada subtipo, que contengan, además de los atributos propios, los atributos comunes. Se elegiría esta opción cuando se dieran las mismas condiciones que en el caso anterior (muchos atributos distintos).

  DOCTOR(Cód_Prof, Nombre, ..., Año_Doc, Materia_Doc)

  NO_DOCTOR(Cód_Prof, Nombre, ...)

- Se aumenta la eficiencia en determinadas consultas (las que afectan a todos los atributos, tanto comunes como propios de un subtipo), pero la podemos disminuir en otras.
- Es la solución que **más semántica pierde**. Si además existe **solapamiento**, se introduce **redundancia** que debe controlarse.

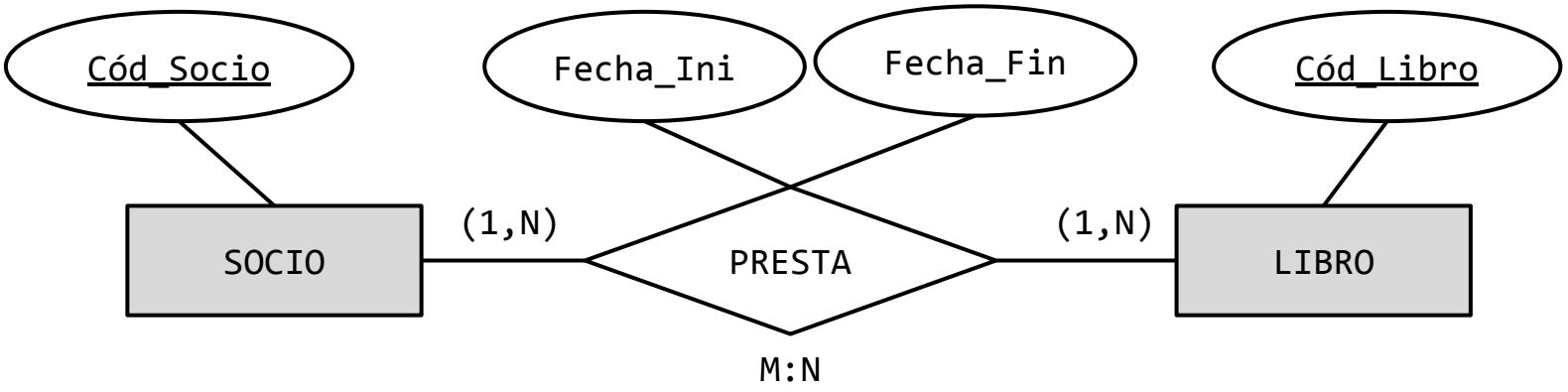
12. Transformación de la dimensión temporal (1)

- En el caso de que en el esquema ER aparezca el tiempo como un tipo de entidad, la transformación en el esquema relacional estándar no constituye mayor problema, ya que se tratará como un tipo de entidad cualquiera y, por tanto, se creará una relación (tabla) adicional:

  `TIEMPO(Fecha_I, Fecha_F, Hora_I, Hora_F, ...)`

- Sin embargo, cuando la dimensión temporal la hemos recogido en el esquema E/R a través de atributos de relación tipo FECHA, la transformación consiste en pasarlos a columnas de la relación que corresponda.
- Sobre este punto debemos tener cuidado a la hora de **elegir la clave primaria de la relación resultante**.

12. Transformación de la dimensión temporal (2)



PRESTA(*Cód_Socio*, *Cód_Libro*, Fecha_Ini, Fecha_Fin)

SOCIO(Cód_Socio, ...)

LIBRO(Cód_Libro, ...)

13. Transformación de relaciones de grado superior a dos

- Si el modelo ER contiene relaciones de grado superior a dos (ternarias, cuaternarias, etc), se requiere hacer un estudio de las cardinalidades mínimas y máximas para llevar a cabo la transformación.
- En general, la transformación de **relaciones ternarias M:N:P** establece que da lugar a una relación cuya clave primaria es la concatenación de los AIP de las entidades que relaciona.
- En el caso de que las **cardinalidades** sean todas **(1,N)**, no es necesario ningún mecanismo adicional para no perder semántica.
- Sin embargo, en el caso de que alguna **cardinalidad mínima sea 0** o en el caso de que algún lado de la relación tenga **cardinalidad (1,1)**, será necesario estudiar cada caso para no perder semántica al realizar la transformación al modelo relacional.

Cuatro pautas de diseño de un esquema relacional

- **Semántica de los atributos**: no mezclar tipos de entidades o de relaciones en un mismo esquema.

  Pauta 1: *Diseñar un esquema de relación de explicación fácil.*

- **Reducción de valores redundantes**: previenen las anomalías de inserción, eliminación y modificación.

  Pauta 2: *Diseñar los esquemas de las relaciones para evitar las anomalías de actualización.*

- **Reducción de los valores nulos**: producen malgasto de espacio y generan dudas de interpretación.

  Pauta 3: *Tratar de evitar situaciones en las que aparezcan atributos con valores nulos.*

- **Prohibición de tuplas espurias**: surgen por una descomposición inadecuada de una relación.

Pauta 4: *Diseñar los esquemas para poder reunirlos mediante condiciones de igualdad (equireunión) sobre atributos que sean claves principales o ajenas.*