Bases de Datos

Tema 6

Diseño Conceptual, Lógico y Físico

Francisco Ruiz

feb-2001

documentación preparada con ayuda de Esperanza Marcos (Universidad Rey Juan Carlos) y Mario Piattini (Universidad de Castilla-La Mancha)

Tema 6

Diseño Conceptual, Lógico y Físico

Complementar con:

** capítulos 9, 10 y 11 del libro "Diseño de Bases de Datos Relacionales". De Miguel, A.; Piattini, M.; Marcos, E.; Ra-Ma, 1999.*

- ***Aprender a diseñar bases de datos** (BD) **relacionales** mediante la metodología presentada en el capítulo anterior.*
- *Conocer las diversas fases de la metodología:*
  - *modelado o **diseño conceptual** (con el modelo entidad/interrelación),*
  - ***diseño lógico** (con las dos subfases, estándar y específico), y*
  - ***diseño físico**.*

- **Principales:**

  – [de Miguel et al, 1999]
    - Caps 9, 10 y 11
    - De Miguel, A.; Piattini, M.; Marcos, E.; Diseño de Bases de Datos Relacionales. Ra-Ma, 1999.

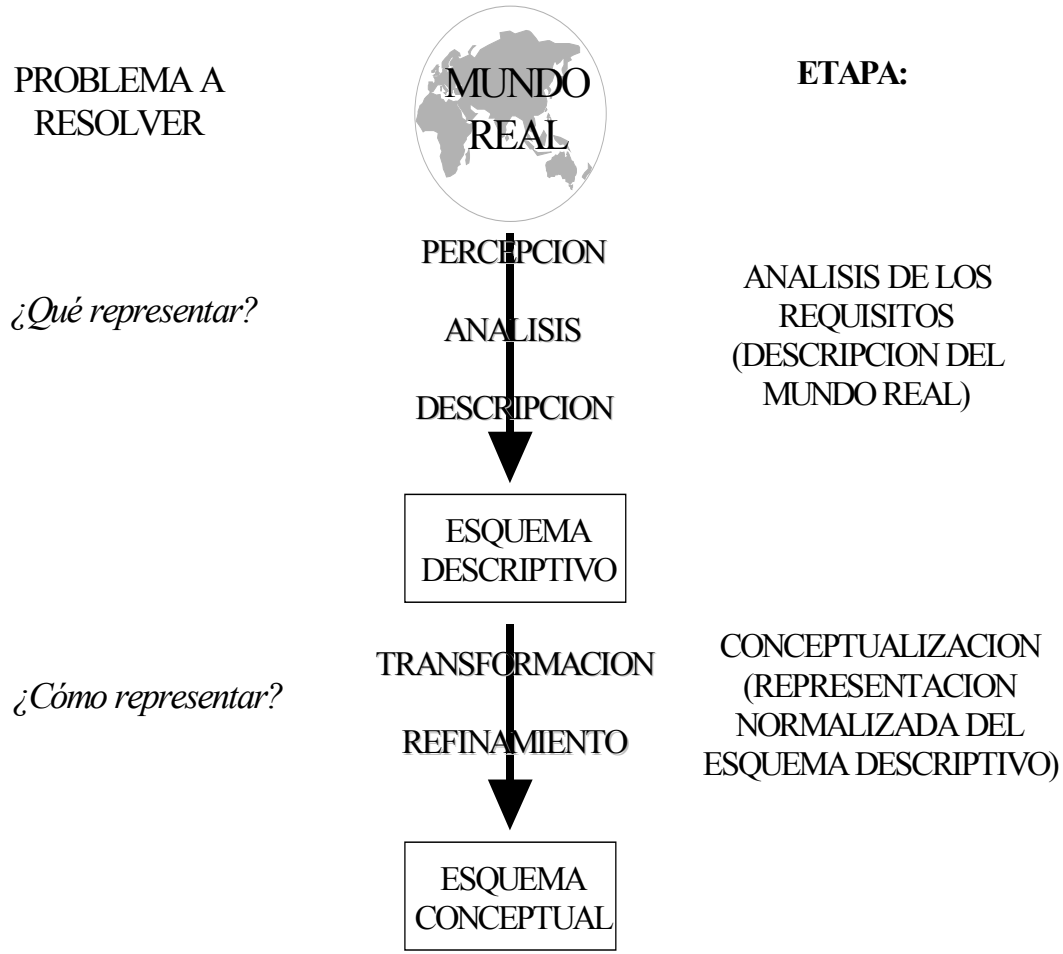
- **Otras:**

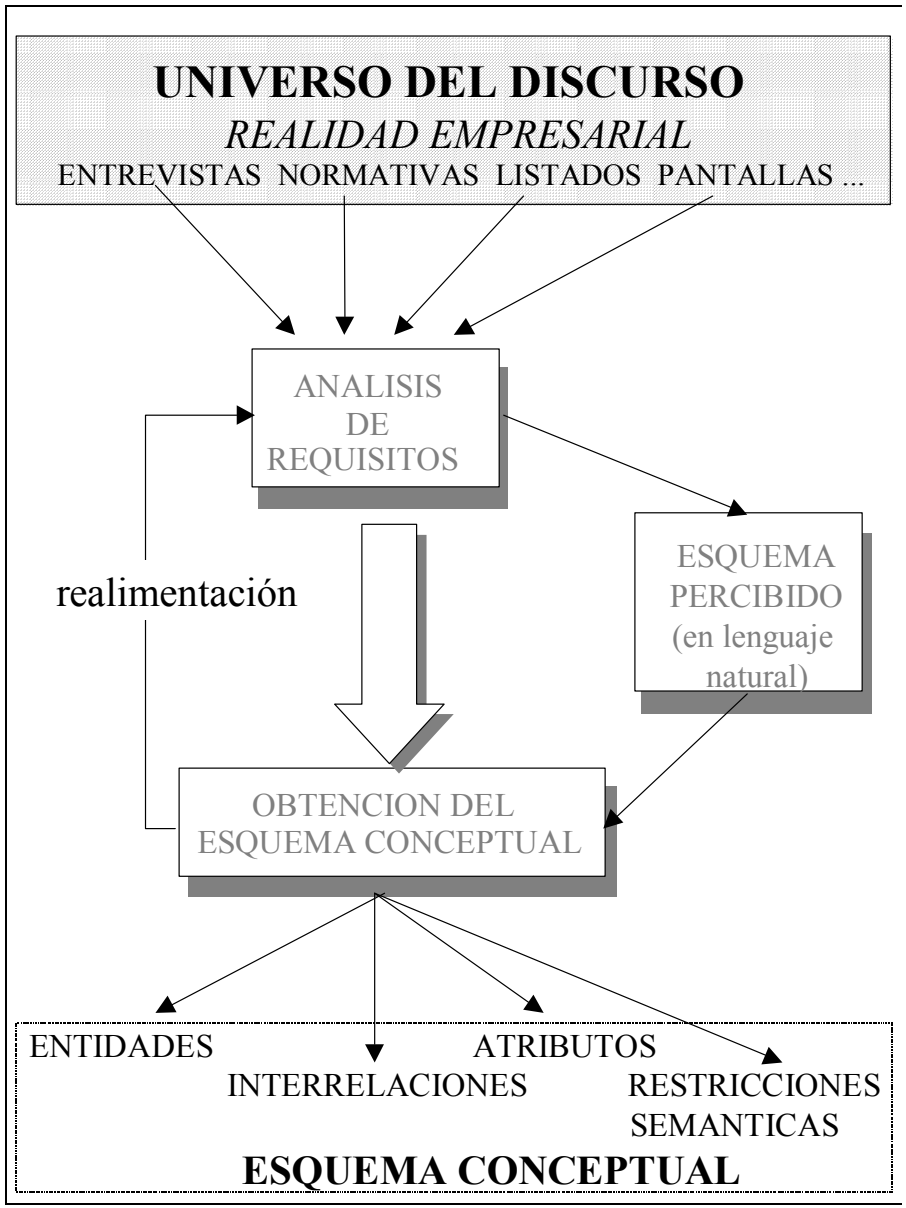
  - Batini, C.; Ceri, S.; Navathe, S.B.; Diseño conceptual de bases de datos. Addison-Wesley Iberoamericana, 1994. Caps. 3 (diseño conceptual) y 12 (diseño lógico en el modelo relacional).

Índice (1)

Índice (2)

- El modelado conceptual, también denominado diseño conceptual, constituye la primera fase de desarrollo de bases de datos, y puede subdividirse en dos etapas claramente diferenciadas:
  - Análisis de Requisitos, y
  - Generación del esquema conceptual (Conceptualización).





- Entradas:
  - Realidad empresarial: entrevistas, normativas, listados, pantallas, etc.
- Salidas:
  - Esquema conceptual: entidades, interrelaciones, atributos, restricciones semánticas, etc.

- Esta primera etapa, en general común para datos y procesos, es la etapa de *percepción, identificación y descripción de los fenómenos* del mundo real a analizar.
- En el análisis de requisitos se ha de responder a la pregunta:
  - **"¿Qué representar?".**
- Mediante el estudio de las reglas de una empresa (que proveen el marco para el análisis del sistema) y de entrevistas a los usuarios de los diferentes niveles de la organización (que proveen los detalles sobre los datos) se llega a elaborar un esquema descriptivo de la realidad.
- El esquema descriptivo se representa utilizando el lenguaje natural. Con ello se ayuda a que el problema de comunicación usuarios/analistas se reduzca (aunque seguirá siendo muy importante).

"Como ven los usuarios a los analistas"

- No entienden el negocio, es decir, la actividad de la empresa
- Intentan decirnos como realizar nuestro trabajo
- No consiguen instrumentar de manera aceptable las especificaciones del sistema
- Dicen NO a todas nuestras sugerencias
- Ponen demasiado énfasis en aspectos técnicos
- Siempre piden más presupuesto
- Siempre se retrasan
- Nos piden tiempo y esfuerzo en detrimento de nuestro trabajo
- No pueden responder de forma rápida y satisfactoria a los cambios necesarios en el sistema

"Como ven los analistas a los usuarios"

- No saben lo que quieren
- Tienen muchas necesidades "políticas"
- Quieren todo YA
- no son capaces de establecer prioridades entre las necesidades
- Quieren poner sus necesidades específicas por delante de las de la compañía u organismo
- Rehusan responsabilidades sobre el sistema
- No son capaces de dar una definición clara del sistema para que funcione
- Son incapaces de respetar la planificación
- No dicen todo lo que saben sobre el sistema

Relaciones entre Analistas y Usuarios

- En esta segunda etapa se transforma el esquema descriptivo, refinándolo y estructurándolo adecuadamente.
- Esta etapa responde a la pregunta:
  - **"¿Cómo representar?"**
- En esta etapa de conceptualización se habrá de buscar una representación normalizada que se apoye en un modelo de datos que cumpla determinadas propiedades (coherencia, plenitud, no redundancia, simplicidad, fidelidad, etc.), para llegar así al denominado esquema conceptual.
- Para la representación del esquema conceptual, usaremos el ME/R extendido,, además de una serie de fichas o plantillas que sirven de soporte documental junto al diagrama E/R.

- Para generar el esquema conceptual es preciso interpretar las frases del lenguaje natural en el que está descrito el esquema percibido, convirtiéndolas en elementos del modelo E/R.
- Si bien no existen reglas deterministas que digan qué elemento va a ser una entidad o cuál otro una interrelación, sí es posible enunciar unos principios generales que, junto al buen criterio del diseñador, puedan ayudar a elaborar un primer esquema conceptual, que será sometido después a un proceso de refinamientos sucesivos.
- Chen propone varias heurísticas (no son reglas absolutas):
  - Un sustantivo (nombre común) que actúa como sujeto o complemento directo en una frase es, en general, un tipo de entidad, aunque podría ser un atributo. Por ejemplo, en la frase "Los estudiantes solicitan becas", existen dos posibles entidades: ESTUDIANTE (sustantivo que actúa como sujeto) y BECA (que actúa como complemento directo).
  - Los nombres propios suelen indicar ejemplares de un tipo de entidad; por ejemplo, "Hens, R." indica un ejemplar de ESTUDIANTE.
  - Un verbo transitivo o una frase verbal es un tipo de interrelación, en la frase anterior "solicitar" indica una interrelación entre las dos entidades, ESTUDIANTE y BECA.
  - Una preposición o frase preposicional entre dos nombres suele ser un tipo de interrelación, o también puede establecer la asociación entre una entidad y sus atributos. Por ejemplo, al decir, "el área del departamento", bien podemos estar indicando la interrelación entre las entidades DEPARTAMENTO y AREA, o bien podemos estar asociando el atributo área a la entidad DEPARTAMENTO.

- Carswell y Navathe proponen reglas basadas en el papel o rol que un determinado objeto desempeña en el proceso de información:
  - Una entidad es un objeto de datos que tiene más propiedades que su nombre o se utiliza como operando en una sentencia de selección, borrado o inserción. Por ejemplo, en la universidad existen profesores que poseen una serie de propiedades, como son el nombre, apellidos, DNI, dirección, etc. PROFESOR es una entidad, ya que tiene unas propiedades (nombre, apellidos, etc.). Otro ejemplo, cuando un ESTUDIANTE deja de serlo es preciso darle de baja de la base de datos; ESTUDIANTE es una entidad, por ser un operando en una sentencia de borrado.
  - Un atributo es un objeto de datos al que se le asigna un valor o se utiliza como operando en una operación aritmética, lógica o cadena de caracteres. Por ejemplo, se puede consultar si el nombre de un profesor es Paloma, por lo que nombre es, según este enfoque, un atributo.
  - Una interrelación es un objeto de datos que hace posible la selección de una entidad por medio de una referencia a un atributo de otra entidad; así, por ejemplo, podemos seleccionar los profesores que pertenecen a una determinada área, por lo que, "pertenecer", es una interrelación, ya que nos permite seleccionar una entidad (PROFESOR) por medio de una referencia a un atributo de otra entidad (Nombre de área).

- Los verbos "SER" y "ESTAR" han sido especialmente estudiados:
  - "**ES UN**" permite crear jerarquías de entidades, que corresponde al concepto de **generalización**. Ejemplo: "...tanto un doctor como un no doctor de nuestra base de datos son profesores...", que puede modelarse como una especialización de PROFESOR en DOCTOR y NO_DOCTOR.
  - El verbo "**TIENE**" posee múltiples interpretaciones en castellano, que pueden ser más o menos específicas según la acepción del verbo en la correspondiente frase:
    - Interrelación general entre entidades: en cuyo caso el verbo se utiliza como otro cualquiera, sin una acepción específica; por ejemplo, "...los alumnos tienen un tutor..." establece la interrelación entre las entidades ALUMNO y TUTOR.
    - Asociación de las entidades con sus atributos (corresponde a la abstracción de **agregación**): si decimos que "...los profesores tienen un nombre y apellidos, un DNI y una dirección..", estamos asociando a la entidad PROFESOR una serie de atributos: nombre, apellidos, DNI, dirección, teléfono.
    - Agregación de entidades para formar una entidad compuesta (también corresponde a la abstracción de **agregación**): por ejemplo, "el curso tiene una parte teórica y una parte práctica", donde CURSO es el elemento compuesto y PARTE_TEÓRICA y PARTE_PRÁCTICA son los elementos componentes.
    - Dependencia en identificación (o en existencia): así, al decir que "...un curso de doctorado tiene varias ediciones...", en el sentido de que una edición es un ejemplar de un curso de doctorado. En este caso el identificador de la entidad que es ejemplar (EDICION) se suele formar con el identificador de la entidad principal (en el ejemplo, CURSO) junto a un atributo discriminante de la ocurrencia.

- Un caso especial es la dificultad muchas veces para determinar si un "objeto de datos" se debe modelar como atributo o como entidad interrelacionada con la entidad de la que se supone que podría ser atributo. Brathwaite propone la siguiente regla:
  - Es preferible considerar el objeto de datos como entidad, en lugar de como atributo, en los siguientes casos:
    - Si el objeto de datos tiene por si mismo asociados otros atributos, por ejemplo, si la materia que imparte un profesor (que considerábamos un atributo de PROFESOR) tiene a su vez otros atributos, como número de temas, horas de práctica, horas de teoría, etc.), conviene crear la entidad MATERIA.
    - Si el objeto de datos esta relacionado con otras entidades: por ejemplo, si el área la hubiéramos considerado como un atributo de PROFESOR, no podríamos reflejar las posibles interrelaciones existentes entre las áreas y los departamentos; por ejemplo, para especificar que el departamento de Informática se compone de las áreas de Lenguajes y Sistemas Informáticos y de Ciencias de la Computación e Inteligencia Artificial.

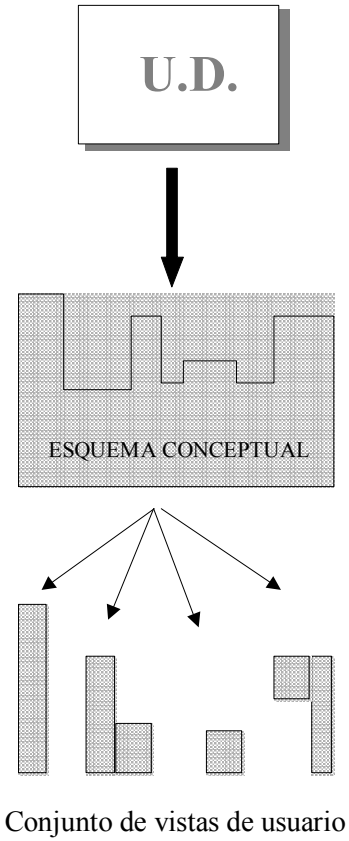
- Como resultado de la fase de modelado conceptual se obtendrá un esquema conceptual que debe cumplir los siguientes objetivos:
  - Captar y almacenar el universo del discurso mediante una descripción rigurosa, representando la información que describe a la organización y que es necesaria para su funcionamiento.
  - Aislar la representación de la información de los requisitos de la máquina y exigencias de cada usuario particular.
  - Independizar la definición de la información de los SGBD en concreto.
- En resumen, "*un esquema conceptual comprende una descripción central única de los distintos contenidos de información que pueden coexistir en una base de datos*".
- Para ello se debe contar con un modelo de datos adecuado: E/R extendido.

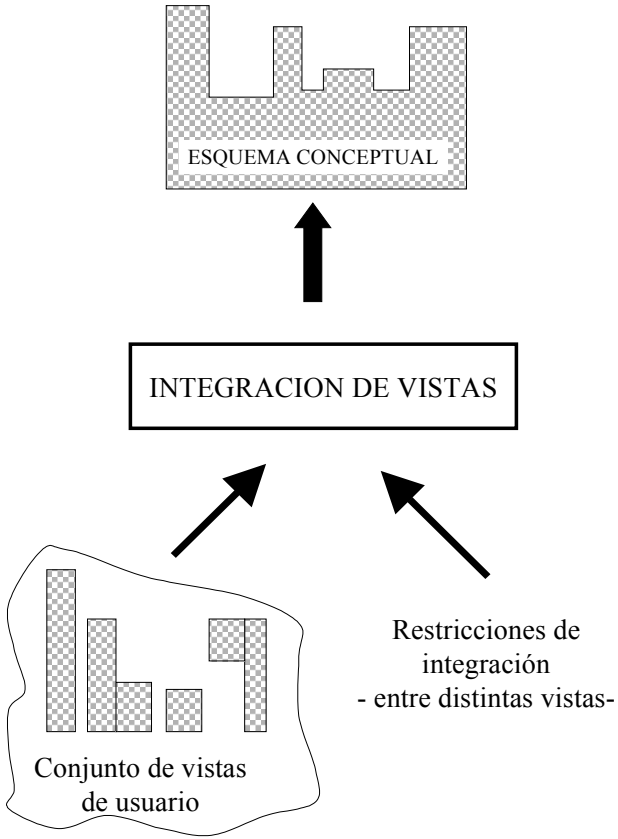
- El ME/R extendido permite obtener esquemas conceptuales que se caracterizan por su:
  - **Claridad**, esto es, que el significado no sea ambiguo.
  - **Coherencia**, es decir, que no existan contradicciones o confusiones.
  - **Plenitud**, en cuanto a que el esquema conceptual ha de representar lo esencial del fenómeno sin buscar la exhaustividad.
  - **Fidelidad**, en el sentido de que la representación del universo del discurso ha de hacerse sin desviaciones ni deformaciones.
  - **Sencillez**, se ha de buscar la máxima sencillez sin atentar contra las anteriores características.

- La sencillez del esquema conceptual ha de estar basada en que:
  - El número de componentes básicos debe ser tan reducido como sea posible.
  - Ha de separar claramente conceptos distintos.
  - Debe preservar la simetría, es decir, no destruir las simetrías naturales.
  - La redundancia tiene que ser cuidadosamente controlada.

- Un mismo universo del discurso puede ser representado mediante múltiples esquemas E/R.
- Es importante establecer en qué medida cada solución propuesta es capaz de recoger la semántica inmersa en el universo del discurso, a fin de poder elegir aquel esquema que incorpore el mayor número de restricciones semánticas del mundo real.
- Para ello, diversos autores han establecido tres criterios de equivalencia entre diagramas E/R, que, de menos a más estrictos, son:
  - Compatibilidad de dominios de datos, que asegura que los diagramas representan, en conjunto, el mismo UD.
  - Equivalencia de dependencias de datos, que asegura que los diagramas satisfacen las mismas restricciones entre los datos representados.
  - Equivalencia de ejemplares, que requiere que los esquemas E/R puedan almacenar los mismos ejemplares del universo del discurso.

- Tradicionalmente, en el diseño de bases de datos se han venido utilizando distintas metodologías para elaborar el esquema conceptual, que pueden agruparse en dos familias:
  - **Descendentes** (*Top-Down*): cuya filosofía responde a que "el esquema conceptual refleje directamente la visión de la empresa que se intenta modelar en la BD". Se parte del estudio del universo del discurso para elaborar el esquema conceptual y, posteriormente, sobre él se definen las vistas de usuario como subconjuntos de este esquema conceptual.
    - Recomendadas para BD pequeñas.
  - **Ascendentes** (*Bottom-Up*): este tipo de metodologías entiende el esquema conceptual como "el resultado de la integración de las vistas de los grupos de usuarios" –subsistemas-, por lo que se empieza construyendo las vistas de cada uno de estos subsistemas (que corresponde a las aplicaciones más importantes) y, teniendo en cuenta las restricciones entre dichas vistas, se elabora el esquema conceptual mediante un proceso de integración de vistas.
    - Recomendadas para BD medianas y grandes.





Diseño Descendente *vs* *Diseño Ascendente*

- Al integrar vistas se debe distinguir entre:
  - las vistas idénticas, que son aquellas en las que se encuentran los mismos tipos de objetos, aunque puede que con distintos nombres; y
  - las vistas no idénticas, denominadas así por poseer (en todo o en parte) distintos tipos de objetos. Dentro de estas últimas, hay que distinguir aquellas que, aunque no sean idénticas, sí resultan equivalentes (por ejemplo, porque lo que en una vista es un atributo en la otra está representado por una entidad) de las que no son equivalentes.
- La integración de vistas consiste en partir de dos vistas y obtener una nueva vista que las englobe, con ésta y una tercera se obtiene una nueva vista, y así sucesivamente, hasta llegar al esquema global que refleje la estructura de información de la BD completa.
- En el proceso de integración de vistas existen dos etapas:
  - Resolución de conflictos.
  - Análisis de redundancias en interrelaciones.

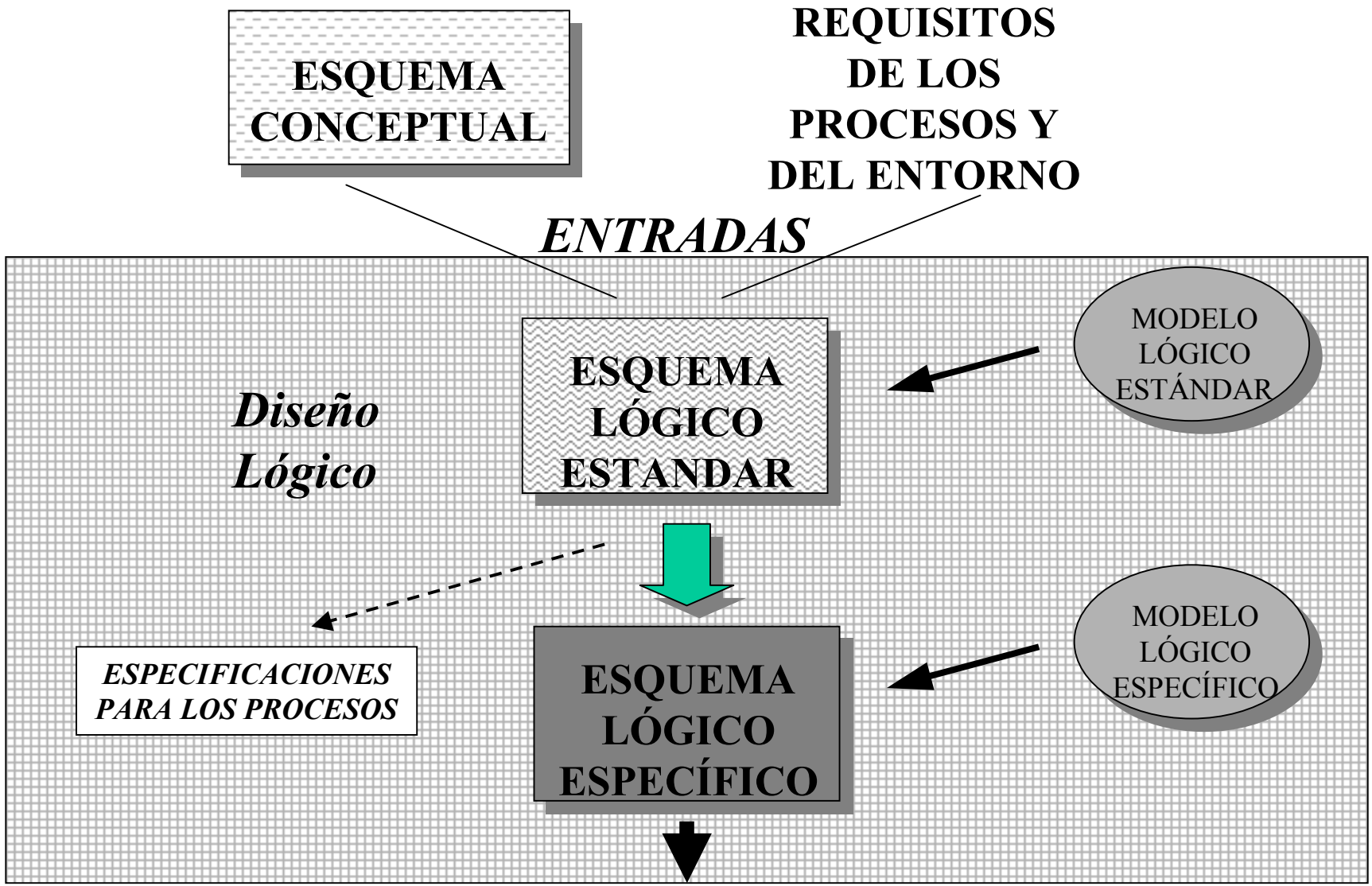
- Al querer integrar varias vistas se pueden producir diversos problemas:
  - A) Conflictos de nombres: pueden ser tanto por *homonimia* (a dos objetos distintos se les ha asignado el mismo nombre) como por *sinonimia* (un mismo objeto que posee más de un nombre).
  - B) Conflictos entre entidades: pueden ser de varios tipos ...
    - 1) Una entidad es un subconjunto de otra. En este caso la solución consiste en introducir un subtipo.
    - 2) Una entidad es disjunta con respecto a otra, pero ambas poseen atributos comunes; es decir, son un subtipo de una tercera entidad. La solución es crear esa tercera entidad, esto es, el supertipo.
  - C) Conflicto entre tipos de objetos en los que un atributo en una vista es una entidad en otra o viceversa: la solución es transformar el atributo en entidad o la entidad en atributo, según convenga. Así, por ejemplo, si existen atributos de este atributo, o si éste está interrelacionado con otras entidades, convendrá considerarlo como entidad.
  - D) Conflictos de cardinalidades en interrelaciones: pueden reflejar que las dos interrelaciones son la misma, que hay dos interrelaciones distintas o que una de las entidades involucradas en la interrelación tiene uno o varios subtipos.

- Una vez integradas las vistas, habrá que analizar si se producen redundancias de interrelaciones, lo que puede reflejarse gráficamente como ciclos en el diagrama E/R. Estos ciclos se deben detectar y estudiar tal como se vió en el capítulo 2:

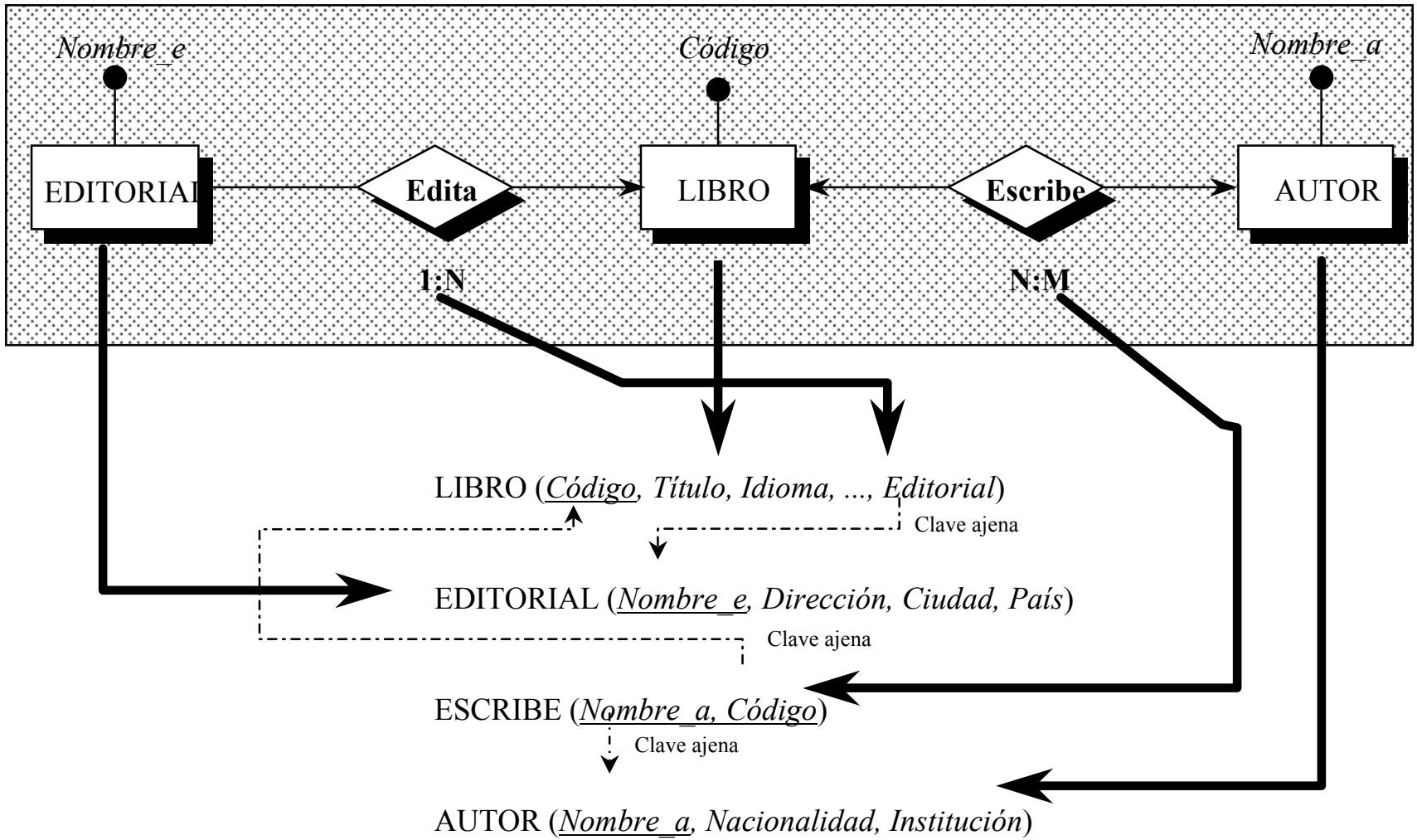
Repaso del capítulo 2:

- La existencia de un ciclo no implica la existencia de interrelaciones redundantes.
- Para que una interrelación pueda ser eliminada por redundante se tiene que cumplir :
  a) que exista un ciclo,
  b) que las interrelaciones que componen el ciclo sean equivalentes semánticamente,
  c) que se puedan asociar los ejemplares de las dos entidades que estaban interrelacionadas, aún habiéndose eliminado la interrelación, y
  d) que la interrelación no tenga atributos o que éstos puedan ser transferidos a otro elemento del esquema a fin de no perder su semántica.

- Diseño Lógico Estándar:
  - A partir del esquema conceptual, y teniendo en cuenta los requisitos de proceso y de entorno, se elabora un esquema lógico estándar (ELS), que se apoya en un modelo lógico estándar (MLS), que será el mismo modelo de datos soportado por el SGBD que se vaya a utilizar, pero sin las restricciones ligadas a ningún producto comercial.
  - En nuestro caso el MLS es el modelo relacional.
  - El ELS se describirá utilizando un lenguaje estándar. En nuestra metodología se usa el SQL-2, ISO (1992).
- Diseño Lógico Específico:
  - Con el ELS, y teniendo en cuenta el modelo lógico específico (MLE) propio del SGBD (INGRES, SYBASE, DB2, ORACLE, INFORMIX, INTERBASE, etc.), se elabora el esquema lógico específico (ELE), que será descrito en el lenguaje de definición de datos (LDD) del producto comercial que estemos utilizando.


ESQUEMA CONCEPTUAL
REQUISITOS DE LOS PROCESOS Y DEL ENTORNO
ENTRADAS
Diseño Lógico
ESQUEMA LÓGICO ESTANDAR
MODELO LÓGICO ESTÁNDAR
ESPECIFICACIONES PARA LOS PROCESOS
ESQUEMA LÓGICO ESPECÍFICO
MODELO LÓGICO ESPECÍFICO

- Las **tres reglas básicas** para convertir un esquema en el modelo E/R al relacional son las siguientes:
  - 1) *Todo tipo de entidad se convierte en una relación.*
  - 2) *Todo tipo de interrelación N:M se transforma en una relación.*
  - 3) *Para todo tipo de interrelación 1:N se realiza lo que se denomina propagación de clave (regla general), o se crea una nueva relación.*
- Debido a que el modelo relacional no distingue entre entidades e interrelaciones, ambos conceptos deben representarse mediante relaciones. Esto implica una **pérdida de semántica** con respecto al esquema E/R:
  - Las interrelaciones N:M no se distinguen de las entidades (ambas se transforman en tablas), y
  - las 1:N se suelen representar mediante una *propagación de clave*, desapareciendo incluso el nombre de la interrelación.



Ejemplo de aplicación de las tres reglas básicas de transformación

- A partir del ELS obtenido en la etapa anterior, y teniendo en cuenta el modelo lógico específico (MLE) en el que se va a instrumentar la base de datos, se elabora el esquema lógico específico (ELE), que será descrito en el lenguaje de definición de datos del producto comercial (SGBD) que se utilizará.
- El paso del ELS al ELE lleva consigo un conocimiento del SGBD para:
  - Ver en que grado soporta el MLS;
  - Adaptar el ELS a las características específicas del SGBD que se va a utilizar (a su ELE); y
  - Definir el ELE en la sintaxis propia del SGBD.
- Al estudiar la correspondencia entre los conceptos del MLS y los del SGBD concreto pueden darse los siguientes casos:
  - A) El SGBD soporta todos los conceptos del MLS sin restricciones. La transformación del ELS al ELE es prácticamente directa, sólo se ha de transcribir a la sintaxis propia del SGBD utilizado (normalmente SQL).
  - B) El SGBD no soporta ciertos conceptos, o bien los soporta pero con restricciones. Tendremos que utilizar entonces nuevos objetos, realizar una programación complementaria, incluyéndola en la metabase o bien, en último caso, transferir a los programas restricciones no soportadas convenientemente por el MLE.

- Algunos aspectos en los que es necesaria una transformación adicional, debido a que los conceptos del MLS no son soportados por el MLE de los SGBD relacionales actuales, son:
  - **1. Dominios**: Si el SGBDR no permite sentencias para la definición de dominios, existen dos opciones:
    - A) Al definir la columna de una tabla especificar el tipo de dato, la longitud y (si el esquema lo permite) las restricciones pertinentes (CHECK).
    - B) Construir un procedimiento que compruebe que los valores que se pretenden insertar o modificar se encuentran en una tabla de una sola columna (llamada tabla de dominio) que no sea susceptible de inserción, borrado o modificación (tabla estática que sólo el ABD podrá modificar).
  - **2. Claves Primarias**: Si el SGBDR no permite la cláusula PRIMARY KEY, el procedimiento sustitutorio es:
    - 1- No admitir nulos en los atributos de la clave primaria (asignar NOT NULL).
    - 2- Definir una restricción de unicidad (UNIQUE) para el conjunto de atributos de la clave primaria.
    - 3- Asegurar la existencia del índice asociado a la restricción de unicidad anterior: crearlo al crear la tabla y no borrarlo nunca.
    - 4- Añadir un comentario en el esquema /catálogo indicando cuál debería ser la clave primaria.

- Continuación:
  - **3. Claves Ajenas**: Si el SGBDR no permite la cláusula PRIMARY KEY, o no incorpora toda la semántica que conlleva, los pasos a seguir son los siguientes:
    - 1- Introducir las restricciones de clave ajena (integridades referenciales) como requisitos en la especificación de los programas.
    - 2- Asignar la cláusula NOT NULL a los atributos de la clave ajena que no admiten nulos.
    - 3- Mantener la definición de clave ajena como un comentario en el catálogo.
    - 4- Utilizar los mecanismos de seguridad del SGBDR para prohibir las operaciones de los usuarios que pueden violar la restricción de integridad.
    - 5- Realizar un programa que permita el chequeo periódico automático de la integridad referencial.
    - 6- Para mejorar la eficiencia en las opciones anteriores, se puede crear un índice formado por todos los atributos de la clave ajena.
  - **4. Otros conceptos del modelo relacional:** Los demás conceptos suelen necesitar el uso de disparadores (triggers) o procedimientos almacenados que comprueben las restricciones de integridad definidas en el ELS.

Reglas Detalladas de Transformación:

- R1: Transformación de dominios.
- R2: Transformación de entidades.
- R3: Transformación de atributos de entidades.
  - R3.1 Atributos identificadores principales.
  - R3.2 Atributos identificadores alternativos.
  - R3.3 Atributos no identificadores.
  - R3.4 Atributos multivaluados.
- R4: Transformación de interrelaciones.
  - R4.1 Interrelaciones N:M.
  - R4.2 Interrelaciones 1:N.
  - R4.3 Interrelaciones 1:1.
  - R4.4 Interrelaciones de grado > 2.
- R5: Transformación de atributos de interrelaciones.
- R6: Transformación de restricciones.
- R7: Transformación de dependencias en identificación y en existencia.
- R8: Transformación de restricciones de interrelaciones.
- R9: Transformación de generalizaciones/especializaciones.
- R10: Transformación de la dimensión temporal.
- R11: Transformación de atributos derivados.

- **R1 - Transformación de Dominios**:
  - Cada dominio E/R se representa mediante una sentencia CREATE DOMAIN en SQL-2:

```
CREATE DOMAIN e_civil AS CHAR(1)
        CHECK (VALUE IN ('S', 'C', 'V', 'D'))
```

ME/R

MR (SQL-2)

DOMINIO E_CIVIL CHAR(1) CHECK (VALUE IN ('S','C','V','D'))

- **R2 - Transformación de Entidades**:
  - Cada tipo de entidad se convierte en una tabla. La tabla se llamará igual que el tipo de entidad de donde proviene.
  - Para su definición en SQL-2 se utiliza la sentencia CREATE TABLE.

- **R3 - Transformación de Atributos de Entidades**:
  - Cada atributo de una entidad se transforma en una columna de la tabla a la que ha dado lugar la entidad. Teniendo en cuenta que existen atributos identificador principal, otros que son identificadores alternativos y el resto de atributos que no son identificadores –atributos no principales-, esta regla se divide en tres subreglas:
  - **R3.1 - Atributos Identificadores**:
    - Los atributos que son identificadores principales (AIP) pasan a formar la clave primaria de la tabla.
    - En SQL se representan con la cláusula PRIMARY KEY dentro de la orden CREATE TABLE.
  - **R3.2 – Atributos Identificadores Alternativos**:
    - Se representan en SQL por medio de la claúsula UNIQUE dentro de la orden CREATE TABLE.
  - **R3.3 - Atributos No Identificadores**:
    - Se representan como columnas de la tabla correspondiente.

ME/R

MR

PROFESOR

Cod_prof	*Nombre*	*DNI*	*Dirección*	*Teléfono*	*Materia*
00001	Juan	12223433	Rios Rosas, 23	670123123	Ing. Software
00002	Coral	54656754	Alarcos, 8	567983456	Bases de datos
00003	Belén	53567523	Getafe, 4	6º9267854	Orientación objetos
00004	Goyo	97856757	Pez, 102	679345763	Sistemas operativos
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
03568	Roberto	34534522	Fundación, 10	639456239	Redes

CLAVE PRIMARIA

Ejemplo de transformación de un tipo de entidad y sus atributos

ME/R

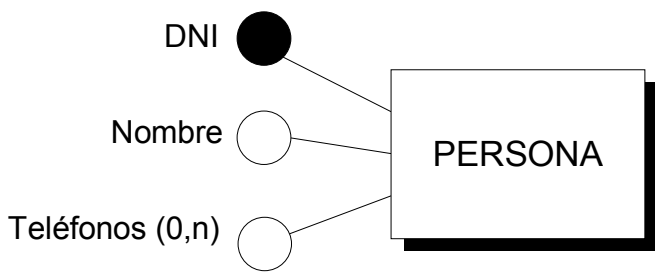
MR – SQL

```
CREATE TABLE Profesor (
                    Cod_Profesor Códigos,
                    Nombre Nombres,
                    DNI DNIS, NOT NULL
                    Dirección Lugares,
                    Teléfono Nos_Teléfono,
                    Materia Materias,
                    PRIMARY KEY (Cod_Profesor),
                    UNIQUE (DNI));
```

Ejemplo de transformación de un tipo de entidad y sus atributos (2)

- **R3.4 - Transformación de Atributos Multivaluados.**
  - Puesto que el modelo relacional no permite dominios multivaluados, deberá crearse una nueva tabla cuyos únicos atributos (y clave primaria) será la concatenación de la clave primaria de la entidad original y el atributo multivaluado. Además, se debe crear una clave ajena referenciando a la tabla primera.

Modelo E/R



Modelo Relacional

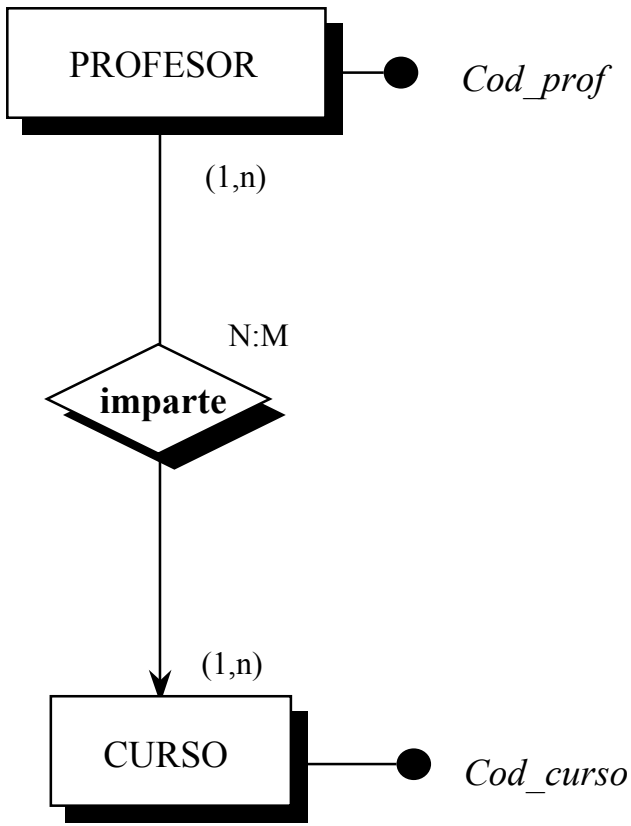
Persona (<u>dni</u>, nombre, ...)

clave ajena

Telefonos(<u>dni, número</u>)

- **R4.1 - Transformación de Interrelaciones N:M.**
  - Un tipo de interrelación N:M se transforma en una tabla que tendrá como clave primaria la concatenación de los AIP de los tipos de entidad que asocia.
  - Además, cada uno de los atributos que forman la clava primaria de esta tabla también son claves ajenas que referencian a las tablas en que se han convertido las entidades interrelacionadas (claves primarias): EN SQL se representa con la cláusula FOREIGN KEY dentro de la sentencia de creación de la tabla.
  - Para cada clave ajena así obtenida deberá estudiarse cuales son los modos de borrado y modificación adecuados (opciones ON DELETE y ON UPDATE en SQL). Las opciones permitidas en SQL-92 son:
    - operación restringida (en caso de no especificar la acción o poner NO ACTION), puesta a nulo (SET NULL), puesta a valor por defecto (SET DEFAULT) y operación en cascada (CASCADE).
  - Las cardinalidades mínimas de las entidades participantes en la interrelación se pueden modelar utilizando restricciones CREATE ASSERTION.

Modelo E/R



Modelo Relacional

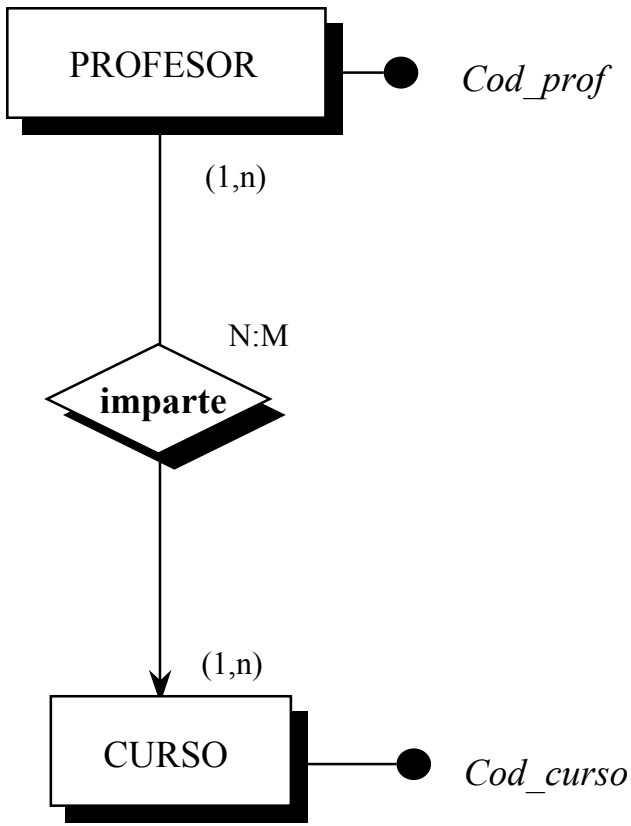
PROFESOR (*Cod_prof*, ...)

CURSO (*Cód_curso*, ...)

IMPARTE (*Cód_curso, Cod_prof*, ...)

Transformación de una interrelación N:M

Modelo E/R



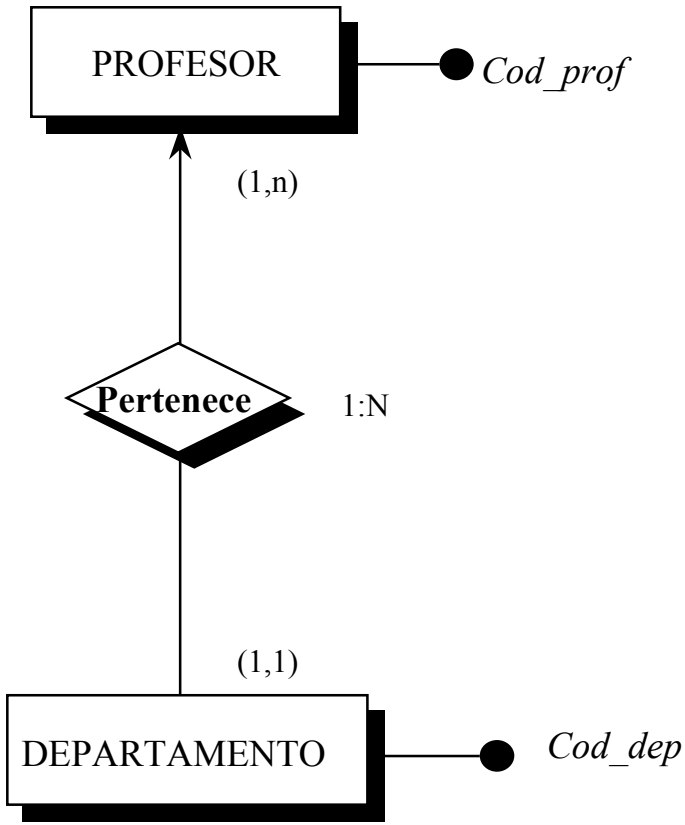
SQL

```
CREATE TABLE Imparte
(Cod_Profesor Codigos_P,
Cod_Curso Codigos_C,
…. ,
  PRIMARY KEY (Cod_Profesor, Cod_Curso),
  FOREIGN KEY (Cod_Profesor) REFERENCES Profesor
    ON DELETE CASCADE
    ON UPDATE CASCADE,
  FOREIGN KEY (Cod_Curso) REFERENCES Curso
    ON DELETE CASCADE
    ON UPDATE CASCADE)
```

Transformación de una interrelación N:M (2)

- **R4.2 - Transformación de Interrelaciones 1:N.**
  - Existen dos soluciones:
    - a) Propagar los AIP del tipo de entidad que tiene de cardinalidad máxima 1 a la que tiene N (propagación de clave). Esta es la regla habitual.
    - b) Transformar la interrelación en una tabla como si se tratara de una interrelación N:M; pero ahora la clave primaria de la tabla creada es sólo la clave primaria de la tabla a la que le corresponde la cardinalidad n.
  - La opción b) se utiliza cuando
    - el número de ejemplares interrelacionados de la entidad que propaga su clave es muy pequeño y, por tanto, existirían muchos valores nulos en la clave propagada.
    - se prevé que la interrelación en un futuro se convertirá en una de tipo N:M.
    - la interrelación tiene atributos propios y no es deseable propagarlos (a fin de conservar la semántica).

Modelo E/R



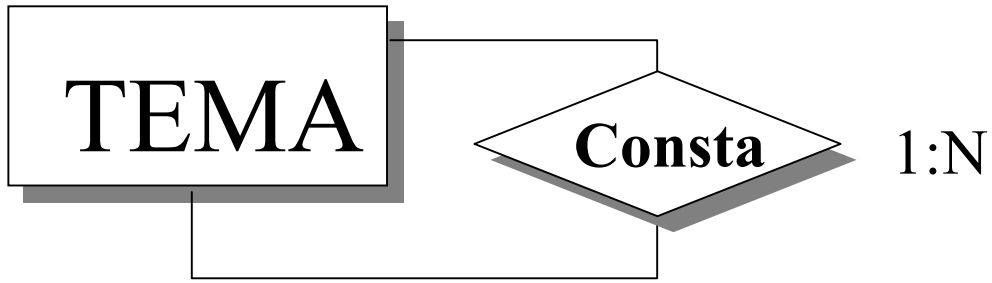
Modelo Relacional

PROFESOR (*Cód_prof*, ..., *Cod_dep*)

DEPARTAMENTO (*Cod_dep*, ...)

Transformación de una interrelación 1:N con propagación de clave

ME/R



Solución a: *propagación de clave*

MR

TEMA (*Cód_tema,, ..., Cod_tema_sup*)
(borrado: puesta a nulos; modificación: cascada)

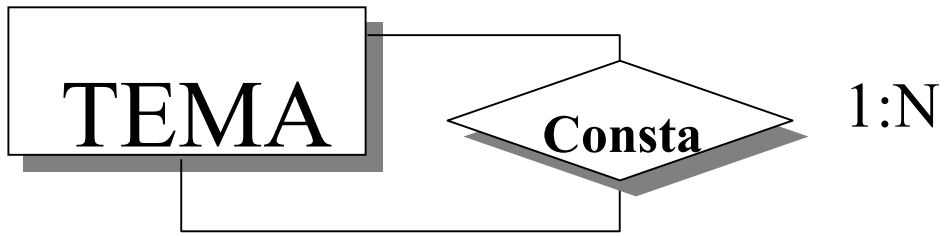
Solución b: *nueva tabla*

Nulos no permitidos

TEMA (*Cód_tema, ...*)
CONSTA (*Cód_tema, Cod_tema_sup...*)
(borrado y modificación: cascada)

Transformación de una interrelación 1:N en una nueva tabla

ME/R

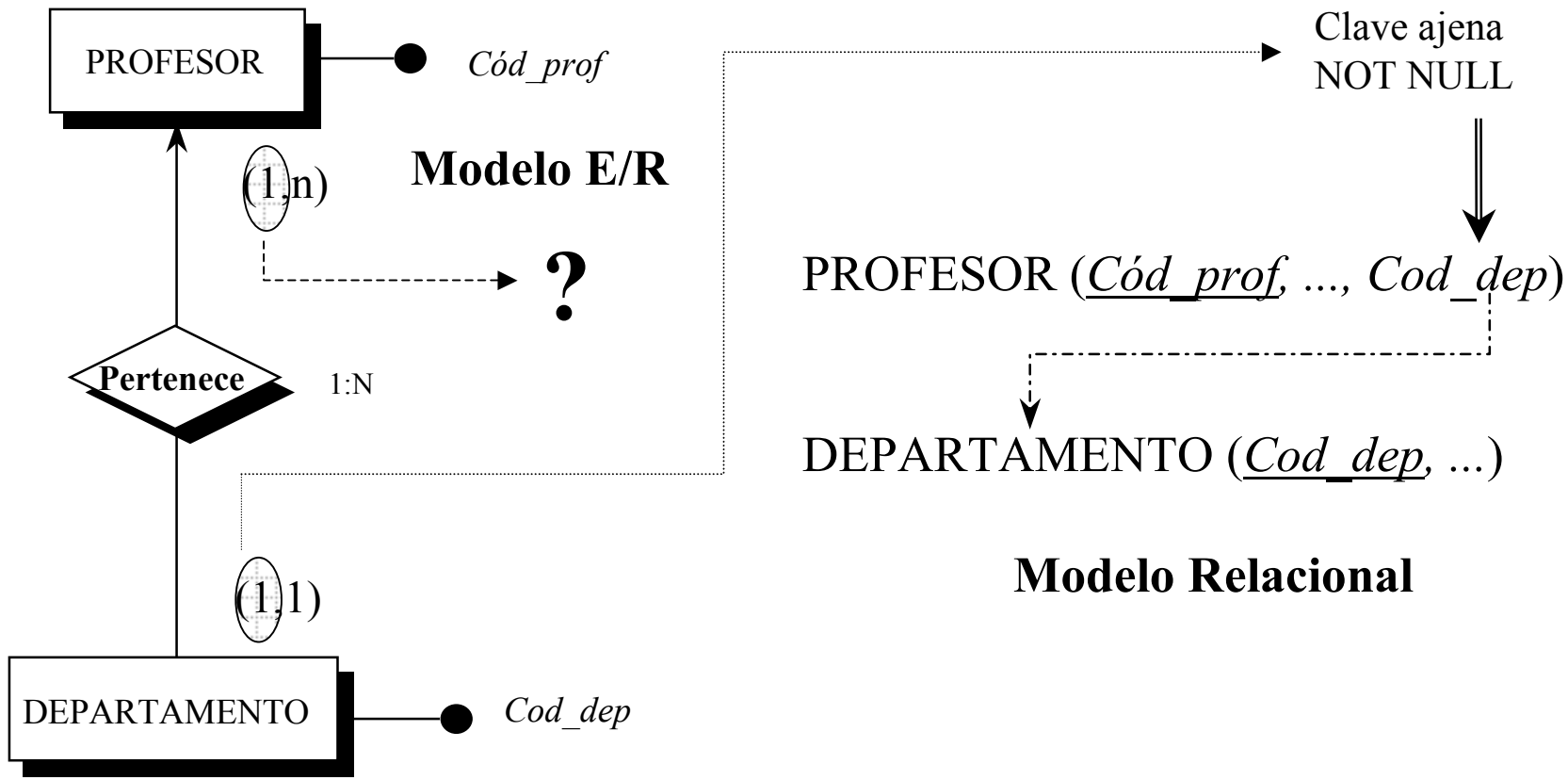


MR

```
CREATE TABLE Consta (
        Cod_Tema Codigos,
        Cod_Tema Sup Codigos,
        PRIMARY KEY (Cod_tema)
        FOREIGN KEY (Cod_Tema) REFERENCES Tema
                ON DELETE CASCADE
                ON UPDATE CASCADE,
        FOREING KEY (Cod_Tema_SUP) REFERENCES Tema
                ON DELETE CASCADE
                ON UPDATE CASCADE
        );
```

Transformación de una interrelación 1:N en una nueva tabla (2)

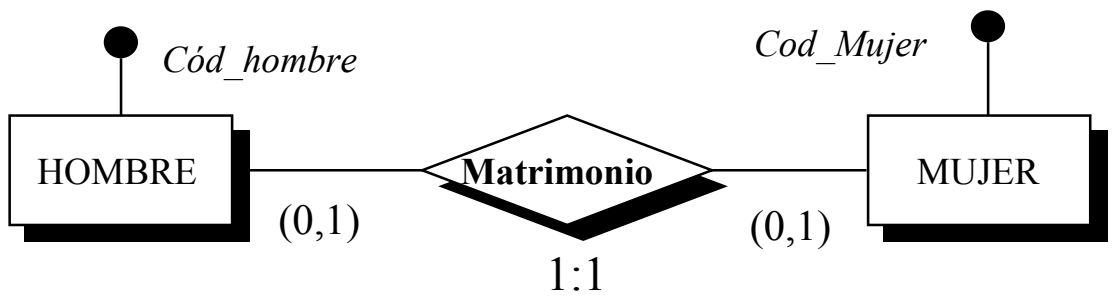
- **R4.2 - Transformación de Interrelaciones 1:N.** ***(cont)***
  - Al hacer propagación de clave, la cardinalidad mínima de la interrelación con máxima 1 se puede modelar usando NOT NULL para el valor 1. Para la interrelación de cardinalidad máxima N es necesaria una restricción (CHECK, ASSERTION o TRIGGER).



- **R4.3 - Transformación de Interrelaciones 1:1.**
  - Una interrelación de tipo 1:1 es un caso particular de una 1:N, por lo que se pueden aplicar las dos opciones ya comentadas: la regla 4.1 (crear una nueva tabla) o aplicar la regla 4.2. (propagación de clave, teniendo en cuenta que ahora la propagación de la clave puede efectuarse en ambos sentidos).
  - Los criterios para aplicar una u otra regla y para propagar la clave se basan en:
    - las cardinalidades mínimas,
    - recoger la mayor cantidad de semántica posible,
    - evitar los valores nulos, o
    - Aumentar la eficiencia.

- **R4.3 - Transformación de Interrelaciones 1:1** (cont)
  - Si las entidades que se asocian poseen cardinalidades (0,1), suele ser conveniente transformar la interrelación 1:1 en una tabla.

Modelo E/R



Modelo Relacional

MATRIMONIO (*Cod_hombre, Cod_mujer*)

HOMBRE (*Cod_hombre*)

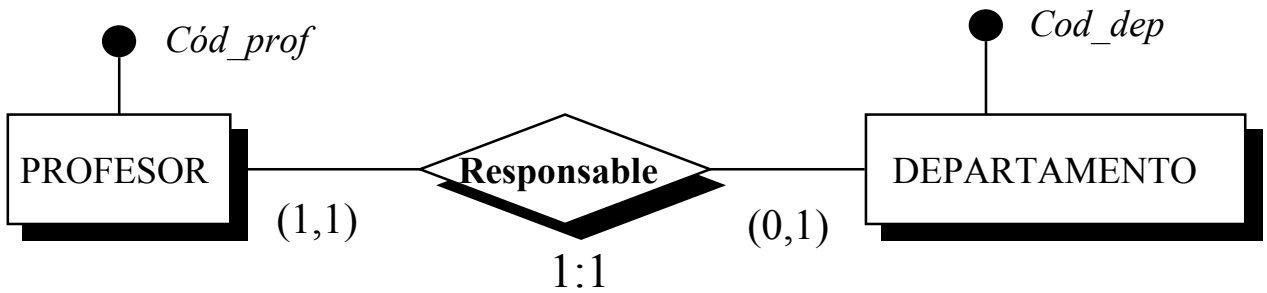
MUJER (*Cod_mujer*)

Clave alternativa

(UNIQUE, NOT NULL)

- **R4.3 - Transformación de Interrelaciones 1:1** (cont)
  - Si las entidades que participan en la interrelación poseen cardinalidades (0,1) y (1,1), conviene propagar la clave de la entidad con cardinalidades (1,1) a la tabla resultante de la entidad con cardinalidades (0,1).

Modelo E/R



Modelo Relacional

PROFESOR (*Cod_prof*)

DEPARTAMENTO (*Cod_dep, ..., Cod_prof*)

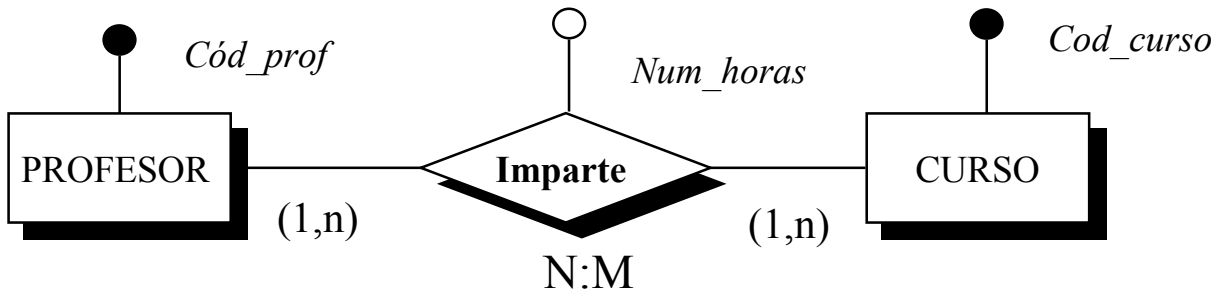
Clave ajena
NOT NULL

- **R4.3 - Transformación de Interrelaciones 1:1** (cont)
  - 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 en este caso los accesos más frecuentes y prioritarios a los datos de las tablas.

- **R4.4 - Transformación de Interrelaciones de grado > 2**
  - Las interrelaciones ternarias, cuaternarias, etc, se representan igual que las interrelaciones N:M, es decir, creando una nueva tabla cuya clave primaria será la concatenación de las claves primarias de los tipos de entidades participantes.
  - Para controlar las cardinalidades mínimas y máximas de cada entidad participante deberá recurrirse a restricciones (CHECK, ASSERTION, ...).

- **R5 - Transformación de Atributos de Interrelaciones**
  - Si la interrelación se transforma en una tabla, todos sus atributos pasan a ser columnas de la tabla.
  - En caso de que la interrelación se transforme mediante propagación de clave, sus atributos migran junto a la clave a la tabla correspondiente.

Modelo E/R



Modelo Relacional

PROFESOR (*Cod_prof*, ...)

IMPARTE (*Cod_prof, Cod_curso*, *Num_horas*)

CURSO (*Cod_curso*, ...)

- **R6- Transformación de Restricciones:** En cuanto a las restricciones de usuario, existen ciertas cláusulas en el LLS que pueden recogerlas:
  - Se pueden restringir a un rango determinado los valores de un dominio a través de la cláusula **BETWEEN**.
  - Se pueden determinar por enumeración los valores que puede tomar una columna en una tabla con la claúsula **IN**.
  - Otra posibilidad es utilizar la cláusula **CHECK** dentro de la descripción de una tabla para expresar una condición que deben cumplir un conjunto de atributos de la tabla (o la sentencia **CREATE ASSERTION** si la comprobación afecta a atributos de más de una tabla). Ejemplo:

```
CREATE TABLE Curso (
    Cod_Curso Cursos,
    Nombre Nombres,
    Num_Horas Horas
    Fecha_I Fechas,
    Fecha_F Fechas,
    PRIMARY KEY (Cod_Curso),
    CHECK ( Fecha_I < Fecha_F));
```

  - También es posible utilizar disparadores que, si bien no existen en el SQL92, si se ofrecen en SQL3 y la mayoría de nuevas versiones de SGBDR.

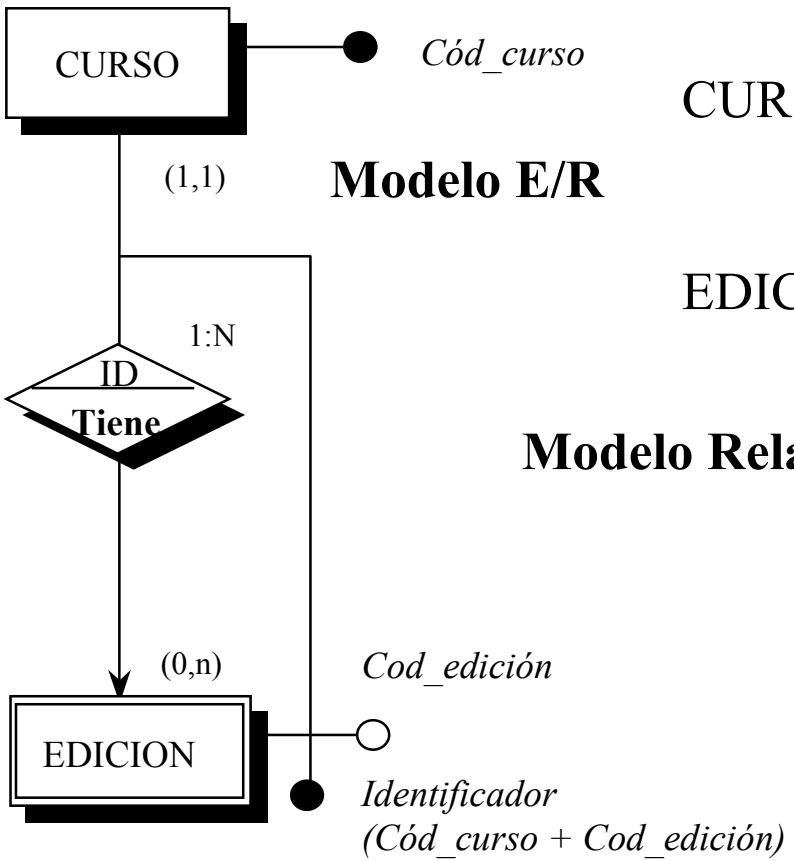
R7 - Transformación de Dependencias en Identificación y en Existencia

- No son recogidas directamente en el MLS. La manera de transformar una interrelación de este tipo es utilizar el mecanismo de propagación de clave, creando una clave ajena, con nulos no permitidos, en la tabla de la entidad dependiente, con la característica de obligar a una modificación y un borrado en cascada.
- Además, en el caso de dependencia en identificación la clave primaria de la tabla en la que se ha transformado la entidad débil debe estar formada por la concatenación de las claves de las dos entidades participantes en la interrelación.
- Ejemplo:

```
CREATE TABLE Curso (
    (Cod_Curso Codigos_cursos, …. , PRIMARY KEY (Cod_Curso));

CREATE TABLE Edicion (
    Cod_Curso Codigos_Cursos,
    Cod_Edicion Codigos_Ediciones, …. ,
    PRIMARY KEY (Cod_Curso, Cod_Edicion)
    FOREIGN KEY (Cod_Curso) REFERENCES Curso
            ON DELETE CASCADE
            ON UPDATE CASCADE);
```

R7 - Transformación de Dependencias en Identificación y en Existencia (cont)



Modelo E/R

CURSO (*Cód_curso*, ...)

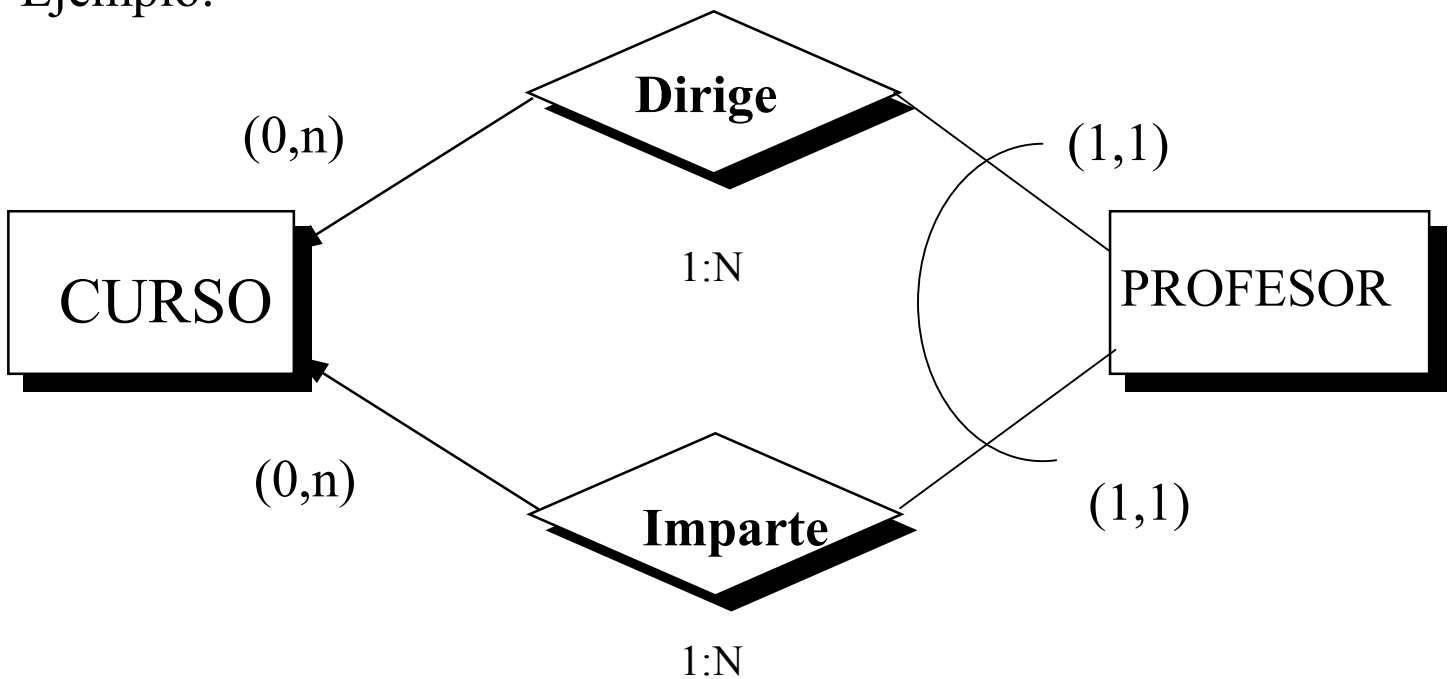
EDICION (*Cód_edicion, Cod_curso*, ...)

Modelo Relacional

Clave ajena
NOT NULL
ON DELETE CASCADE
ON UPDATE CASCADE

R8 - Transformación de Restricciones de Interrelaciones

- Para representar restricciones de interrelaciones en el MLS (exclusión, inclusión, etc.) es necesario utilizar las reglas 4.X comentadas junto con la definición de las restricciones (CHECK, ASSERTION) pertinentes en cada caso.
- Ejemplo:



R8 - Transformación de Restricciones de Interrelaciones (cont)

– Ejemplo:

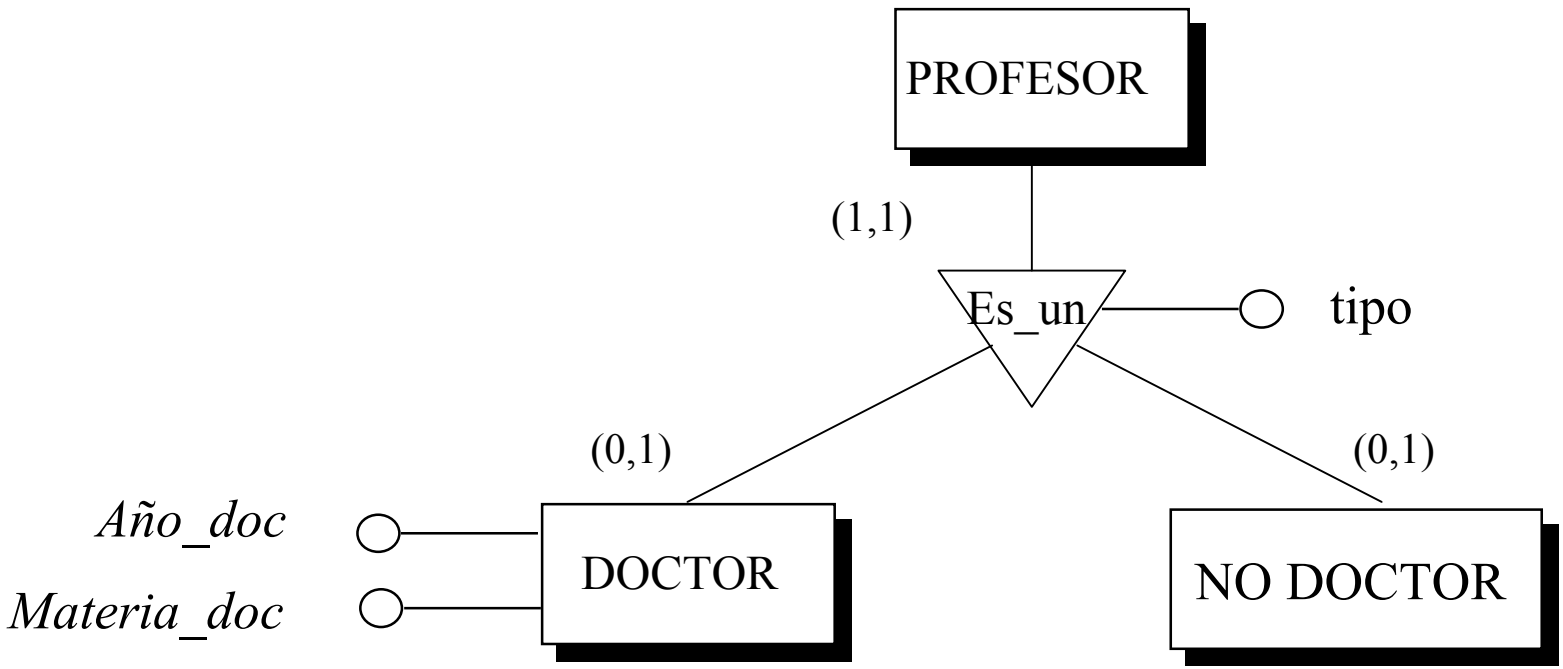
```
CREATE TABLE Curso (
        Cod_Curso Codigos_Cursos,
        Nombre Nombres,
        …. ,
        Cod_prof_dirige Cods_profesores,
        Cod_prof_imparte Cods_profesores,
        PRIMARY KEY (Cod_Curso)
        FOREIGN KEY (Cod_prof_dirige) REFERENCES Profesor
                ON UPDATE CASCADE
        FOREIGN KEY (Cod_prof_imparte) REFERENCES Profesor
                ON UPDATE CASCADE
        CHECK (( Cod_prof_dirige NOT IN (SELECT Cod_prof_imparte FROM
Curso))
                AND
                ( Cod_prof_imparte NOT IN (SELECT Cod_prof_dirige FROM
Curso)));
```

R9 - Transformación de Generalizaciones (tipos y subtipos)

Existen 3 soluciones de transformación al modelo relacional:

- A) Englobar todos los atributos de la entidad y sus subtipos en una sola tabla. En general, se debe adoptar esta solución cuando los subtipos se diferencian en muy pocos atributos y las interrelaciones que los asocian con el resto de las entidades del esquema sean las mismas para todos (o casi todos) los subtipos.
- B) Crear una tabla para el supertipo y tantas tablas como subtipos haya, con sus atributos correspondientes. Esta 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 tabla.
- C) Considerar tablas distintas para cada subtipo, que contengan, además de los atributos propios, los atributos comunes. Se elegirá esta opción cuando se dan las mismas condiciones que en el caso anterior –muchos atributos distintos- y los accesos realizados sobre los datos de los distintos subtipos siempre afectan a atributos comunes.

R9 - Transformación de Generalizaciones (tipos y subtipos) (cont)



R9 - Transformación de Generalizaciones (tipos y subtipos) (cont)

- Opción A)

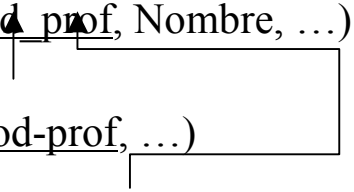
  PROFESOR (Cod_prof, nombre, …, tipo,Año_doc, ...)

- Opción B)

  PROFESOR (Cod_prof, Nombre, …)

  DOCTOR (Cod-prof, …)

  NO_DOCTOR (Cod_prof,…)



- Opción C)
  - DOCTOR (Cod_prof, Nombre, …, Año_doc, ...)
  - NO_DOCTOR (Cod_prof, Nombre, …)

R9 - Transformación de Generalizaciones (tipos y subtipos) (cont)

- También habrá que especificar las restricciones semánticas correspondientes, por ejemplo:

```
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) );
```

- El atributo discriminante de la generalización podrá admitir valores nulos en el caso de que haya recubrimiento parcial y deberá declararse como NOT NULL si el recubrimiento es total.
- El atributo discriminante constituirá un grupo repetitivo (multivaluado), si los subtipos solapan (generalización con solapamiento), debiendo, en este caso, separar este atributo en una tabla aparte que tendrá como clave la concatenación de la clave del supertipo con el atributo discriminante.

R9 - Transformación de Generalizaciones (tipos y subtipos) (cont)

- Aunque es posible elegir cualquiera de las tres estrategias para la transformación de un tipo y sus subtipos al modelo relacional,
  - desde un punto de vista exclusivamente semántico la opción b es la mejor; y
  - desde el punto de vista de la eficiencia deberá tenerse en cuenta que:
    - Opción a: El acceso a una fila que refleje toda la información de una determinada entidad es mucho más rápido (no hace falta combinar varias tablas).
    - Opción b: La menos eficiente aunque es la mejor desde un punto de vista exclusivamente semántico.
    - Opción c: Con esta solución se aumenta la eficiencia ante determinadas consultas (las que afecten a todos los atributos, tanto comunes como propios, de un subtipo) pero se disminuye ante otras. Esta solución es en la que se pierde más semántica; además si existe solapamiento se introduce redundancia que debe ser controlada si se quieren evitar inconsistencias.
- Se deberá elegir una estrategia u otra dependiendo de que sea la semántica o la eficiencia la que prime para el usuario en un momento determinado.

R10 - Transformación de la Dimensión Temporal

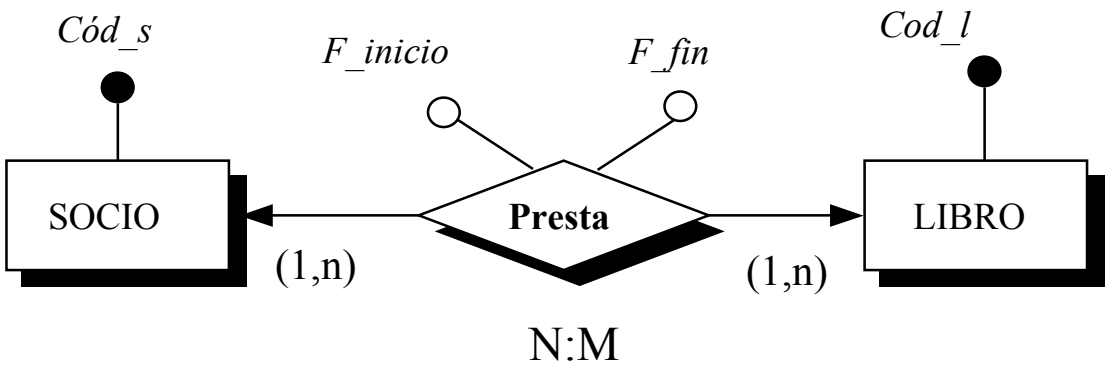
- En el caso de que en el esquema E/R aparezca el tiempo como un tipo de entidad, se tratará como otro tipo de entidad cualquiera y, por tanto, se creará una tabla más:

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

- Cuando la dimensión temporal se ha representado a través de atributos de interrelación de tipo FECHA, la transformación en el MLS consiste en pasarlos a columnas de la tabla que corresponda, pero teniendo especial cuidado a la hora de elegir la clave primaria de la tabla resultante, dependiendo de los supuestos semánticos del entorno.

R10 - Transformación de la Dimensión Temporal (cont)

Modelo E/R



Modelo Relacional

PRESTA (*Cod_s, Cod_l, F_inicio, F_fin*)

SOCIO (*Cod_s, ...*)

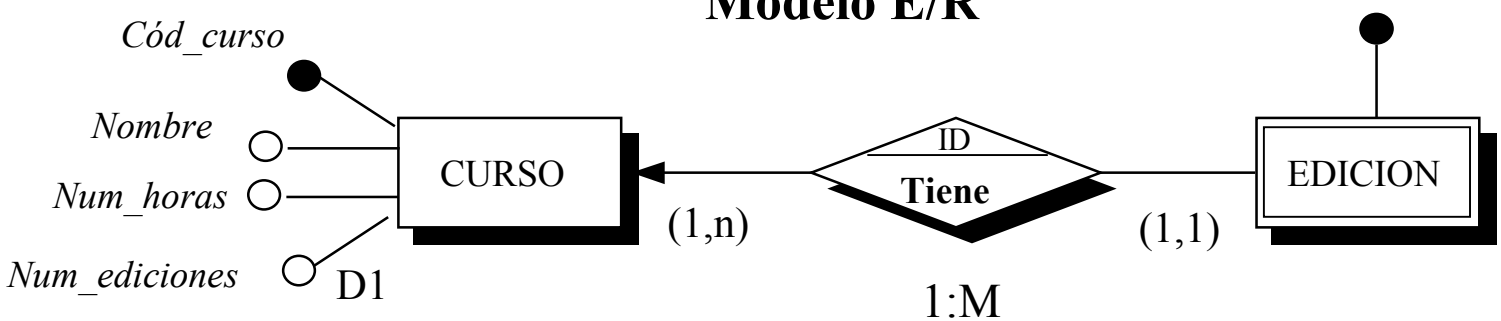
LIBRO (*Cod_l, ...*)

!OJO!
con la clave primaria

R11 - Transformación de Atributos Derivados

- No existe para los atributos derivados una representación directa y concreta en el MLE. Por tanto, se deben tratar como atributos normales, que pasarán a ser columnas de la tabla correspondiente. Además, se deberá construir un disparador que calcule el valor del atributo derivado cada vez que se inserten o borren las ocurrencias de los atributos que intervienen en el cálculo y añadir la restricciones correspondientes.

Modelo E/R



Modelo Relacional

CURSO (*Cod_curso, Nombre, Num_horas, Num_ediciones*)

- La última etapa de la metodología de diseño de bases de datos es el **diseño físico**, cuyo *objetivo general es satisfacer los requisitos del sistema optimizando la relación costes/beneficios*. Esto se concreta en los siguientes objetivos concretos:
  - Disminuir los tiempos de respuesta,
  - Minimizar el espacio de almacenamiento,
  - Evitar las reorganizaciones periódicas,
  - Proporcionar la máxima seguridad, y
  - Optimizar el consumo de recursos.

- Las **entradas** de la fase de diseño físico son:
  - Lista de objetivos de diseño físico con sus correspondientes prioridades y cuantificación (a ser posible);
  - Esquema lógico específico;
  - Recursos de máquina disponibles;
  - Recursos de software disponibles (Sistema Operativo, middleware, ...);
  - Información sobre las aplicaciones que utilizarán la base de datos; y
  - Políticas de seguridad de datos.
- A partir de estas entradas, se producirán las siguientes **salidas**:
  - Estructura interna (esquema interno);
  - Especificaciones para el ajuste (*tunning*) de la base de datos; y
  - Normas de seguridad.

- Los fabricantes de SGBDR abordan el problema del diseño físico desde tres perspectivas diferentes:
  - A) El SGBD impone una estructura interna y deja muy poca flexibilidad al diseñador. Esto suele suponer una mayor independencia físico/lógica a costa de menor eficiencia.
  - B) El ABD diseña la estructura interna. Esto supone más trabajo y un perjuicio para la independencia de datos, aunque puede mejorar la eficiencia.
  - C) El SGBD proporciona una estructura interna inicial a partir de algunos parámetros dados por el diseñador. El ABD puede ir afinándolos (tunning) para mejorar el rendimiento.
- En general, la mejor opción es la C porque:
  - La base de datos puede empezar a funcionar inmediatamente;
  - La eficiencia va aumentando al ir realizando ajustes posteriores;
  - La independencia físico/lógica se mantiene.
  - Es la estrategia de la mayoría de los SGBDR actuales y también la mejor se adapta a la metodología propuesta.

- Algunos de las técnicas más importantes que se pueden considerar en el diseño físico son:
  - Determinación de los índices secundarios y sus características (compresión, orden, etc.);
  - Tipo de registros físicos;
  - Uso de punteros;
  - Direccionamiento calculado (*Hashing*);
  - Agrupamientos (*Clustering*) de tablas;
  - Bloqueos (*Locking*) y compresión de datos;
  - Definición de tamaños de memorias intermedias (*Buffers*);
  - Asignación de conjuntos de datos a particiones y/o a dispositivos físicos; y
  - Redundancia de datos.
- NO existe un modelo formal general para el diseño físico, sino que depende mucho de cada producto comercial concreto.

Repasar estructuras de ficheros e índices de busqueda (árboles B, Hashing, ...)