DISEÑO LÓGICO DE Bases de Datos

- Objetivo Principal: Transformar el Esquema Conceptual de Datos en el Esquema Lógico de Datos
- Otros objetivos del Diseño Lógico:
  - Eliminar redundancias
  - Conseguir máxima simplicidad
  - Evitar cargas suplementarias de programación

  para conseguir ...
  - una Estructura Lógica adecuada
  - un equilibrio entre los requisitos de usuario y la eficiencia
- Diseño Lógico con la máxima PORTABILIDAD

  ✌ Introducción "tardía" del SGBD específico
  - Implementación del Esquema Lógico sobre diferentes SGBD comerciales
  - Migración entre versiones de un mismo SGBD

ETAPAS del DISEÑO LÓGICO

① **Diseño Lógico Estándar (DLS)**

- Transformación independiente del SGBD específico
- Se elige el MODELO DE DATOS, no el SGBD concreto

  Esquema Conceptual ⇨ Esquema Lógico eStándar (ELS)
- Uso de un Modelo Lógico de datos eStándar (MLS)
  - Relacional ←
  - Red
  - Jerárquico
  - Orientado a Objetos
- ELS descrito mediante lenguaje estándar del modelo de datos
  - Diagrama de Estructura de Datos
  - SQL-92 en el Modelo Relacional

ETAPAS del DISEÑO LÓGICO

② **Diseño Lógico Específico (DLE)**

- Se elige el SGBD específico
- Adaptación del Esquema de BD a un SGBD concreto (comercial)

Esquema Lógico Estándar ⇨ Esquema Lógico Específico (ELE)

- Uso del Modelo Lógico de Datos propio del SGBD elegido
  - Informix, Oracle, DB2, Interbase,...
- ELE descrito mediante lenguaje DDL del SGBD específico

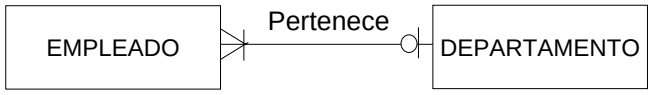
Diagrama de Estructura de Datos, DED

- Técnica de representación gráfica de los esquemas **lógicos** de datos en los modelos convencionales (en particular, el modelo relacional)
- Notación "a medio camino" entre los modelos E/R y Relacional
- Soportados por herramientas CASE (ej. *System Architect*)
- Uso del DED en la metodología METRICA v.2.1
  - Fase 1:
    - ARS 3.2: Diseño del Esquema Lógico Actual de Datos
    - EFS 2.1: Construcción del Esquema Lógico de Datos
    - EFS 2.2: Normalización del Esquema Lógico de Datos

Diagrama de Estructura de Datos, DED

- Características del DED:
  - Únicamente Interrelaciones Binarias (dos entidades)
  - Sólo permitidas las Interrelaciones 1:N
    - Transformación de Interrelaciones 1:1
      - Fusionar en una única entidad, o mantener la interrelación
    - Transformación de Interrelaciones M:N
      - Creación de una entidad auxiliar + Dos interrelaciones 1:N

Ejemplo de Interrelación N:1



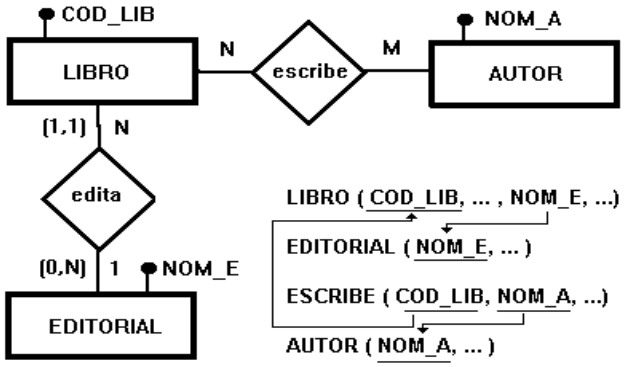
Normalización del Diagrama de Estructura de Datos

- 1FN: atributos con valor atómico
  - Evitar atributos multivalorados
  - Evitar atributos compuestos
- 2FN: en toda entidad E, los atributos no identificadores dependen de manera total del identificador principal de E
  - Ningún atributo (no identificador) de E depende sólo de una parte de cualquier identificador (principal, alternativo) de E
- 3FN: No existen dependencias funcionales transitivas entre los atributos de E.
  - Todo atributo no identificador sólo depende directamente de los identificadores.

DISEÑO LÓGICO ESTÁNDAR

- Reglas para el Modelo Básico
  - Dominios
  - Atributos
  - Entidades
  - Interrelaciones
- Reglas para el Modelo Extendido
  - Jerarquías de Especialización/Generalización

DISEÑO LÓGICO ESTÁNDAR



REGLAS BÁSICAS

Tipo de Entidad----------⇨ Relación
Tipo de Interrelación N:M------⇨ Relación
Tipo de Interrelación 1:N y 1:1 ⇨ "Propagación de clave" o nueva Relación

☹ ¡¡ Pérdida de semántica **!!**

Dominios y Tipos de Entidad

- **DOMINIO**

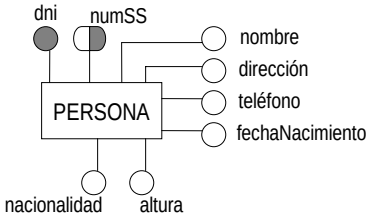
M E/R

MR

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

- **Tipo de ENTIDAD**
  - Relación o tabla (se recomienda usar un nombre similar, o el mismo)

MER



MR

```
PERSONA( dni          PRIMARY KEY,
         numSS        NOT NULL UNIQUE,
         nombre,
         direccion,
         telefono,
         fechaNacimiento,
         nacionalidad,
         altura)
```


Atributos

(a) Atributo Identificador y No Identificador

- Identificador Principal ⇨ Clave Primaria (PRIMARY KEY)
- Identificador Alternativo ⇨ Clave Alternativa (UNIQUE)

- Atributo NO identificador ⇨ Columna
  Podrá contener NULO si no se indica lo contrario

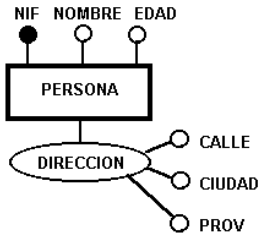
(b) Atributo Simple y Monovaluado ⇨ Columna

Atributos

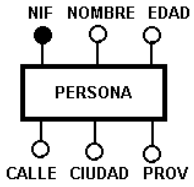
(c) *Atributo Compuesto* del tipo de entidad E

A) "Eliminar" atributo compuesto y considerar todos sus componentes como atributos simples de la relación R

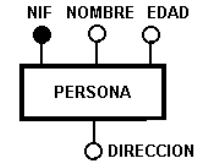
B) "Eliminar" los componentes y considerar el atributo compuesto como un único atributo de R



A)



B)

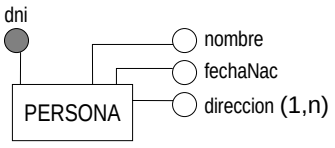


¿Cuándo será más adecuado utilizar una opción u otra?

Atributos

(d) *Atributo Multivalorado* del tipo de entidad E

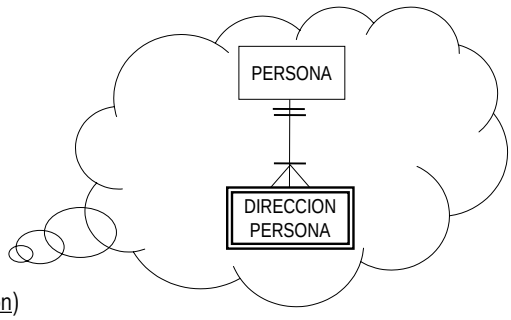
- Nueva Relación S, en la que el atributo multivalorado se representa como un atributo simple A
- S contendrá un atributo F, clave ajena a la clave primaria de R
- Clave Primaria de S = (F, A)



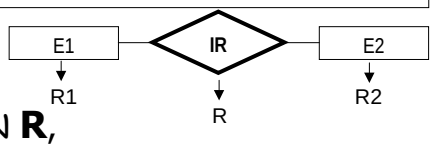
PERSONA(<u>dni</u>, nombre, fechaNac)

FK

DIRECCION_PERSONA(<u>dni, direccion</u>)



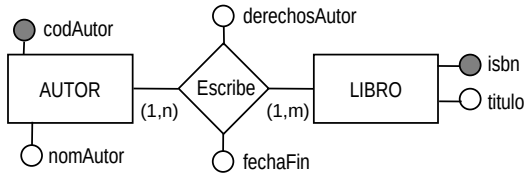
Interrelación Binaria M:N



»» AÑADIR UNA NUEVA RELACIÓN **R**,

que incluye atributos...

- para cada clave primaria de R1 y de R2
  - Claves ajenas a la clave primaria de R1 y R2, respectivamente
  - Su combinación (concatenación) forma la **clave primaria de R**
- para cada atributo simple (o componente simple de atributo compuesto) del tipo de interrelación IR



AUTOR(<u>codAutor</u>, nomAutor, ...)
FK
ESCRIBE(<u>codAutor, isbn</u>, fechaFin, derecAutor)
FK
LIBRO(<u>isbn</u>, titulo, ...)

[dMP93]

Interrelación Binaria M:N

- Especificación de acciones disparadas por Integridad Referencial

```
CREATE TABLE ESCRIBE
(codAutor     Autores,
 codLibro     Codigos,
 fechaFin     DATE NOT NULL,
 derecAutor   NUMBER(2) DEFAULT 20,
 PRIMARY KEY (codAutor, codLibro),
 FOREIGN KEY(codAutor) REFERENCES AUTOR(codAutor)
                       ON DELETE RESTRICT
                       ON UPDATE CASCADE,
 FOREIGN KEY(codLibro) REFERENCES LIBRO(isbn)
                       ON DELETE CASCADE
                       ON UPDATE CASCADE
);
```

Interrelación Binaria M:N

- Especificación de Restricciones o Asertos
  - Para recoger CARDINALIDADES MÍNIMA y MÁXIMA

```
Un libro debe tener entre 1 y 4 autores
CREATE ASSERTION num_autores_por_libro
CHECK (
  (4>=(SELECT MAX(ocurrencias)
        FROM (SELECT COUNT(*) AS ocurrencias
              FROM ESCRIBE
              GROUP BY codLibro)))
  AND
  (1<=(SELECT MIN(ocurrencias)
        FROM (SELECT COUNT(*) AS ocurrencias
              FROM ESCRIBE
              GROUP BY codLibro))));
```

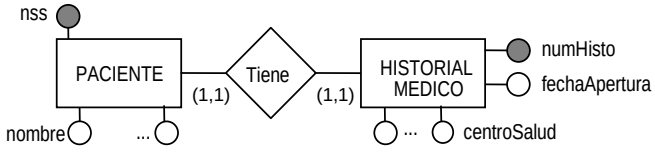

Interrelación Binaria 1:1

(a) Participación TOTAL de ambos tipos de entidad

Los tipos de entidad NO participan en otros tipos de interrelación

»» UNA ÚNICA RELACIÓN **R**

- Propagación de **claves** en una u otra dirección (es indiferente)
  - Clave Primaria de R = clave primaria de R1 **o** de R2 (*si son distintas*)
  - La otra será clave alternativa (UNIQUE) y además NOT NULL
- **Atributos** simples de **IR** o componentes simples de atributos compuestos, también se incluyen como atributos de la relación R

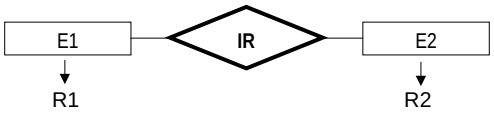


PACIENTE(nss, nombre, numHisto, fechaApertura, centrosalud, ...)
PK **AK, NN**

Interrelación Binaria 1:1

(b) Participación TOTAL de una entidad y PARCIAL de la otra

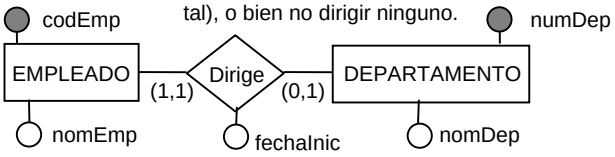
(b.1) **Caso general**



»» PROPAGACIÓN DE CLAVE

– La clave de la entidad con participación parcial "se propaga" hacia la entidad con participación total → clave ajena

Un empleado de una empresa puede ser el gerente de un (único) departamento (desde cierta fecha, en la que fue nombrado como tal), o bien no dirigir ninguno.



[dMP93]

EMPLEADO(codEmp, nomEmp, ...)

FK

DEPARTAMENTO(numDep, nomDep, codDire, fechaInic...)

Interrelación Binaria 1:1

(b.2) **Hay pocas instancias del tipo de interrelación**

»» AÑADIR UNA NUEVA RELACIÓN **R**

– Atributos de R:
  - uno(s) para cada clave primaria de R1 y de R2
    – son claves ajenas (a la clave primaria de R1 y de R2, respectivamente)
    – son claves candidatas en R
      » uno de ellos será la Clave Primaria de R (la de participación total, si existe)
      » el otro será Clave Alternativa de R (UNIQUE) y además NOT NULL
  - atributos simples (o componentes simples de atributos compuestos) de IR

– Evita NULOS en los atributos propagados

EMPLEADO(codEmp, nomEmp, ...)

FK

DIRIGE(codEmp, numDep, fechaInic)

AK, NN **FK**

DEPARTAMENTO(numDep, nomDep,...)

Interrelación Binaria 1:1

(b.3) **Hay muchas instancias del tipo de interrelación**

»» UNA ÚNICA RELACIÓN **R**

- Atributos: todos (los del tipo de entidad y los del tipo de interrelación)
- Clave Primaria: la de la entidad con participación PARCIAL (EMPLEADO)
- Debe permitirse NULOS en los atributos propagados (empleados NO directores)
  - desde la entidad con participación TOTAL y
  - desde la interrelación

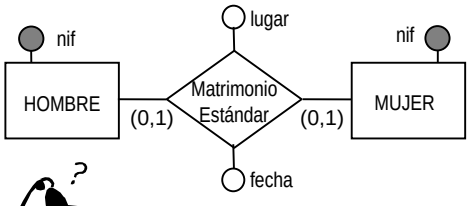
```
EMPLEADO(
   codEmp  ...      PRIMARY KEY,
   nomEmp ... ,
   ...,
   numDepDir ...  NULL UNIQUE,
   nomDepDir ...  NULL,
   ...,
   fechaInicDir ...  NULL,
   ... )
```


Interrelación Binaria 1:1

(c) Participación PARCIAL de ambos tipos entidad

»» AÑADIR UNA NUEVA RELACIÓN **R**

- R se construye exactamente igual que en el caso (b.2)
- Evita los valores nulos que aparecerían si se propagara la clave de R1 a R2 o viceversa (caso (b.1))



HOMBRE(nif, ...)

FK

MATRIMONIO(nifEsposa, nifEsposo, fecha, lugar)

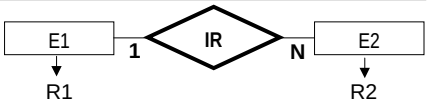
AK, NN

FK

MUJER(nif, ...)

Y... ¿qué **acciones disparadas por la Integridad Referencial** deberemos imponer para (todos los casos de) las interrelaciones 1:1?...

Interrelación Binaria 1:N

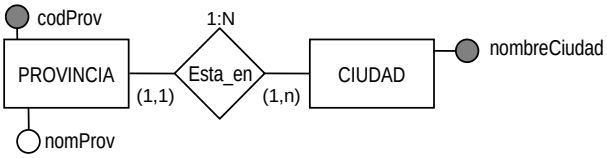


(a) *Caso general*

»» PROPAGACIÓN DE CLAVE

- En R2 se incluyen nuevos atributos para contener valores de...
  - **clave primaria** de R1
    - Clave ajena en R2 hacia R1 (ojo con acciones disparadas por Integridad Referencial)
  - **atributos** simples (o componentes simples de atributos compuestos) de IR

(a.1) Participación TOTAL u obligatoria de E2 en IR: card(E2)=(1,1)



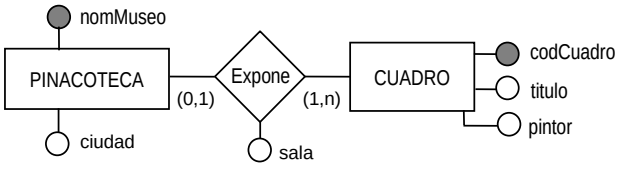
CIUDAD(<u>nomCiudad</u>, codProv, ...)

FK: NULOS NO PERMITIDOS

PROVINCIA(<u>codProv</u>, nomProv, ...)

Interrelación Binaria 1:N

(a.2) Participación PARCIAL u opcional de E2 en IR: card(E2)=(0,1)



NULOS PERMITIDOS

CUADRO(<u>codCuadro</u>, titulo, pintor, nomMuseo, sala...)

FK

PINACOTECA(<u>nomMuseo</u>, ciudad, ...)

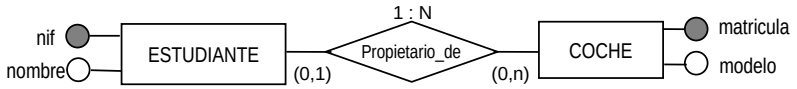
Interrelación Binaria 1:N

(b) *Caso en el que se cumple uno de estos supuestos*:

- Hay pocas ocurrencias del tipo de interrelación
  - Se tendrían demasiados NULOS en la clave propagada, o
- IR tiene varios atributos propios, o
- Es probable que IR se transforme en un futuro en una interrelación N:M

»» AÑADIR UNA NUEVA RELACIÓN **R**

- R se construye exactamente igual que para interrelaciones 1:1 (caso b.2)
  - **Clave primaria**: propagada desde la entidad cuyas instancias s ólo participan una vez (o ninguna) en la interrelación



ESTUDIANTE(nif, nombre, ...)

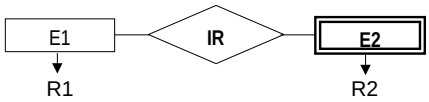
FK NN

PROPIEDAD(nifEstudiante, matricula)

FK

COCHE(matricula, modelo, ...)

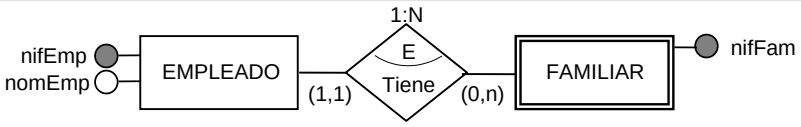
Dependencia en existencia e identificación



- Caso particular de IR 1:1 o 1:N con propagación de clave y participación total de E2 - (caso a.1)
  - clave ajena F de R2 hacia R1 (propagación de clave desde R1 a R2)
    no permite NULO
  - **clave primaria** de R2:
    - DEPENDENCIA EN EXISTENCIA
      » atributos clave primaria de R2 (identificador principal de E2)
    - DEPENDENCIA EN IDENTIFICACIÓN
      » combinación de atributos: F y clave (parcial) de R2

- Actualizaciones y Borrados en R1 se transmiten en CASCADA hacia R2

Dependencia en existencia e identificación

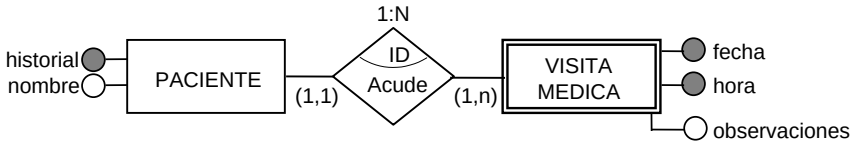


EMPLEADO (nifEmp, nomEmp, ...)

FAMILIAR (nifFam, nifEmp, ...)

FK

nulos no permitidos
ON DELETE CASCADE
ON UPDATE CASCADE



PACIENTE (historial, nombre, ...)

VISITA_MEDICA (historial,fecha,hora, ...)

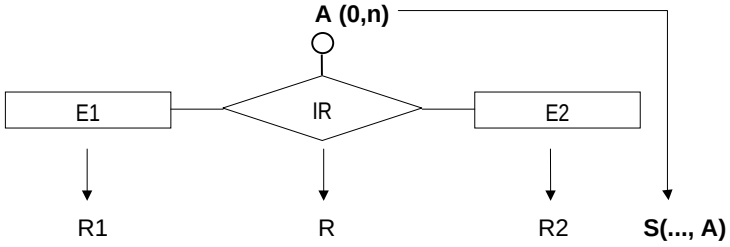
FK

nulos no permitidos
ON DELETE CASCADE
ON UPDATE CASCADE

Atributo Multivalorado de Interrelación

»» NUEVA RELACIÓN **S**,

- El atributo multivalorado se representa como un atributo simple A
- El resto de atributos simples de IR (si los hay) pasan a S



Atributo Multivalorado de Interrelación

Según IR sea...

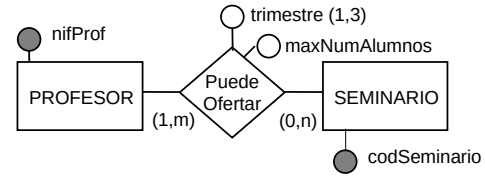
- **1:1**
  - **S** incluye un atributo **F**, clave ajena a la clave primaria de **R1 o** de **R2**
  - Clave Primaria de S = **(F, A)**

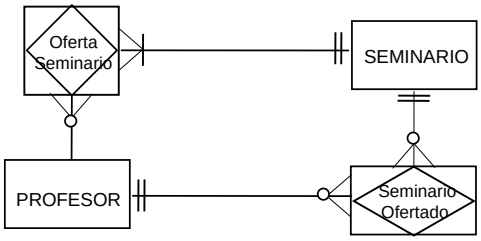
- **1:N (E2 *es el tipo entidad con cardinalidad N*)**
  - **S** incluye un atributo **F**, clave ajena a la clave primaria de **R2**
  - Clave Primaria de S = **(F, A)**

- **N:M**
  - **S** incluye dos atributos **F1 y F2**, claves ajenas a las claves primarias de **R1** y **R2**, respectivamente
  - Clave Primaria de S = **(F1, F2, A)**

Atributo Multivalorado de Interrelación

Caso N:M





R1 PROFESOR(nifProf, ...)

FK

R **OFERTA_SEMINARIO**(nifProf, codSeminario, maxNumAlumnos)

FK

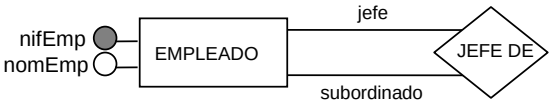
R2 SEMINARIO(codSeminario,...)

FK **FK**

S **SEMINARIO_OFERTADO**(nifProfesor, codSemin, trimestre)

F1 **F2** **A**

Interrelaciones Reflexivas



Caso N:M

EMPLEADO (nifEmp, nomEmp, ...)

JEFE_DE(nifJefe, nifSubordinado, ...)

Casos 1:N

EMPLEADO (nifEmp, nomEmp, ...)

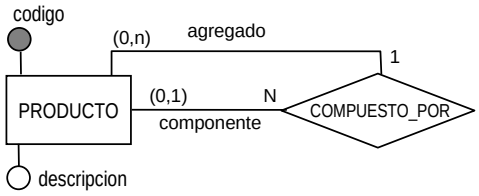
JEFE_DE (nifJefe, nifSubordinado, ...)

EMPLEADO (nifEmp, nomEmp, ..., nifJefe, ...)

(Solución problemática si puede haber muchos empleados sin jefe → demasiados nulos)

- **Relación** donde la clave primaria del tipo de entidad aparece referenciada DOS VECES
- Nombrar esos atributos según los roles del tipo entidad en la interrelación

Interrelaciones Reflexivas (cont.)



[EN97]

PRODUCTO(codigo, descripcion, ...)

FK FK

COMPONENTE(codAgregado, codComponente)

--- un producto es componente de un único producto, o de ninguno

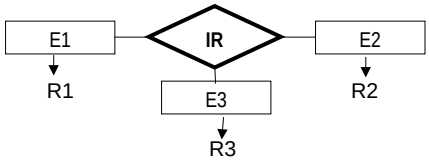
(Al Producto Agregado o Compuesto)

FK: nulos permitidos

PRODUCTO(codigo, descripcion, codProducto,...)

Producto o Componente

Interrelaciones n-arias



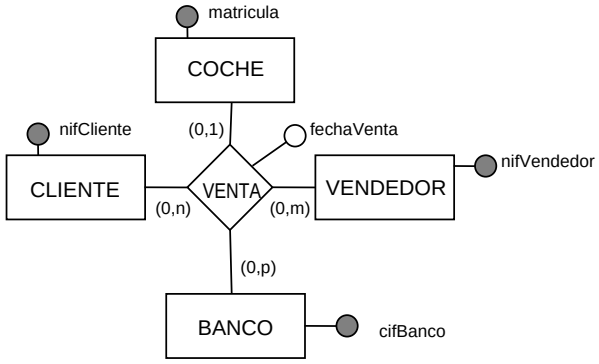
»» AÑADIR UNA NUEVA RELACION **R** que incluye atributos...

- para cada clave primaria de R1, R2, R3...
  - Claves ajenas a la relación R_i correspondiente
- para cada atributo simple (o componente simple de atributo compuesto) de IR

– Clave primaria de R:

- Normalmente, es la combinación de todas las claves externas hacia R_i
- pero es posible que la PK de R sea un subconjunto de esa superclave

Interrelaciones n-arias



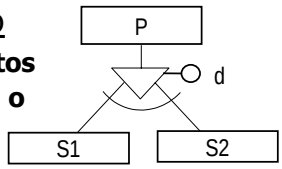
VENTA (matricula, nifVendedor, nifCliente, cifBanco, fechaVenta, ...)

*¿Cuál es la **superclave** de esta relación?
¿y cuál es su **clave primaria?
***¿Cómo asegurar que no haya **ventas** sin cliente o sin coche o sin vendedor?
****¿Puede reflejarse la existencia de ventas directas (sin banco)?

Jerarquías de Especialización/Generalización

(a) Transformación Dirigida por el Supertipo

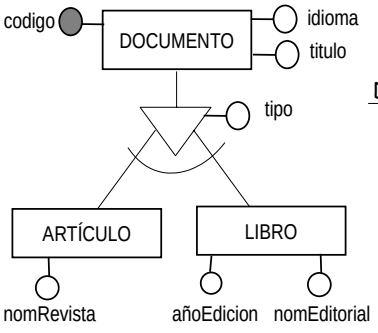
- Los **subtipos se diferencian en pocos atributos**
- **Interrelaciones** establecidas **con el supertipo o son las mismas para todos los subtipos**



– Se crea una **única relación** R que contiene...
  - TODOS los atributos del supertipo **P** y de los subtipos **S1** y **S2**
  - un atributo nuevo -- atributo **discriminante d** de la jerarquía
  - (posibles) nuevas **restricciones semánticas**

– La clave primaria de R es el atributo correspondiente al AIP del supertipo

Jerarquía de Especialización/Generalización



Atributo DISCRIMINANTE

```
CREATE TABLE DOCUMENTO(
   codigo ... PRIMARY KEY,
   titulo... ,
   idioma ... ,
   tipo ... ,
   nomEditorial ... NULL,
   añoEdicion ... NULL,
   ...
   CHECK (( tipo = "ARTICULO" AND
            nomRevista IS NOT NULL AND
            añoEdicion IS NULL AND
            nomEditorial IS NULL)
       OR  ( tipo = "LIBRO" AND
            nomRevista IS NULL AND
            añoEdicion IS NOT NULL AND
            nomEditorial IS NOT NULL))
);
```

Restricciones SEMÁNTICAS

Jerarquía de Especialización/Generalización

- Si la **jerarquía** es **TOTAL**, el discriminante no permite NULOS
- Si la **jerarquía** es **SOLAPADA**,
  - Tratar el discriminante como un ATRIBUTO MULTIVALUADO, o
  - Añadir un atributo por cada subtipo (booleano que indica si $\in$ o $\notin$ al subtipo)

Ventajas e Inconvenientes

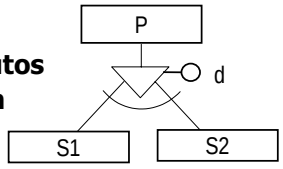
☺ Acceso eficiente a TODA la información sobre una entidad concreta (acceso a una sola relación)

☹ *Aparición de nulos (atributos que proceden de subtipos para entidades que no pertenecen a tales subtipos)

* Toda operación sobre subtipos debe "buscar" las instancias de los subtipos en el conjunto completo (supertipo) de instancias

Jerarquía de Especialización/Generalización

(b) Transformación "Total"



- Los **subtipos se diferencian en muchos atributos**
- **Se desea mantener los atributos comunes en una relación separada**

- una relación R para el supertipo P
  - incluye atributos de P
  - la clave primaria de R es el atributo correspondiente al AIP del supertipo
- una relación R_i para cada subtipo S_i
  - contiene atributos del subtipo S_i y un atributo clave ajena hacia la clave primaria de R
  - La clave primaria de cada R_i es el atributo clave ajena a la clave primaria de R

☺ *Funciona para jerarquías de todo tipo. La mejor desde el pto. de vista semántico

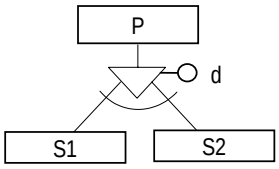
*Conviene si las operaciones son estrictamente "locales" a los subtipos o bien al supertipo (e.d. casi nunca se accede a la vez a atributos de subtipo y supertipo)

☹ Menos eficiente en el acceso (**¿Por qué?**)

Jerarquía de Especialización/Generalización

(c) Transformación Dirigida por los Subtipos

- Existen **muchos atributos NO comunes** (en los subtipos)
- Existen **pocos atributos comunes** (en el supertipo)
- Los **accesos a** datos de **subtipos siempre afectan también a datos comunes**



– Se crea una relación R_i para cada subtipo S_i
  - contiene atributos del subtipo S_i y
  - atributos comunes (del supertipo)

– La clave primaria de cada R_i es el atributo AIP del supertipo

☺ *Conviene si el concepto que representa el supertipo no se requiere en el dis. lógico
*Funciona para jerarquías totales y exclusivas
*Acceso muy eficiente a toda la información (reunión ya "integrada" en el esquema)

☹ *Con jerarquías solapadas aparecen "repeticiones"
*Con jerarquías parciales surgen problemas de "falta de representación"
*Para buscar una ocurrencia del supertipo, hay que buscar en todas las relaciones procedentes de los subtipos.

DISEÑO LÓGICO ESPECÍFICO (DLE)

- Conocimiento del SGBD
  ¿Soporta el MLS? ¿Hasta qué punto?
  ¿Cómo escribir el ELE con la sintaxis propia del SGBD?
- Estudiar la correspondencia entre conceptos del MLS y del SGBD

  Pueden darse dos casos:

  1. SGBD con soporte total del MLS sin restricciones
     - Transformación (casi) directa al SQL propio del SGBD
  2. SGBD no soporta algunos conceptos, o sí lo hace pero con restricciones
     - Uso de conceptos distintos alternativos
     - Programación complementaria
- **La mayor parte del ELS sirve como ELE**, así que sólo veremos los aspectos que necesitan transformaciones adicionales

DLE: **Dominios**

- Algunos productos comerciales sólo ofrecen sintaxis de definición de dominios, pero no implementan la semántica asociada
  - Según Codd (1990) el uso de dominios tiene estas ventajas
    - Declaración única de cada tipo de datos permitido en el esquema,
    - Soporte de integridad y coherencia entre dominios (operaciones compatibles como la UNION, INTERSECCION, etc.),
    - Posibilidad de creación de operadores y características propias de los dominios,
    - Facilitar la definición de comprobaciones del SGBD (menor / mayor que),
    - Posible indexación sobre el dominio, no sobre las columnas de las tablas,
    - Simplificar operaciones complejas sobre varias columnas, haciendolas directamente sobre el dominio
- La mayoría de SGBD NO ofrece ningún soporte para dominios

  Alternativa:
  - Definir tipo de datos, longitud, restricciones para cada atributo (columna)
  - Simulación:
    - Tablas de Dominio y
    - Procedimientos de comprobación de valores correctos (control de integridad)

DLE: **Claves Primarias**

- Si el SGBD no dispone de sintaxis para definición de PK o sólo ofrece la sintaxis para hacerlo, pero no implementa su semántica (como Oracle6)...
  - Especificar cada atributo componente de la PK como NOT NULL
  - Especificar que la combinación de todos los componentes de la PK ha de tener valores únicos (y asegurar esto tras inserciones y actualizaciones)
  - Mantener la definición de cada clave primaria como comentario en el catálogo del SGBD o, si éste lo soporta, incluir la definición sintáctica

*Nota: en el estándar SQL2 no es obligatorio especificar la PK de una relación, y en los productos comerciales tampoco (por compatibilidad con versiones anteriores)

DLE: **Claves Ajenas**

- Pocos productos soportan este concepto (Oracle7 sí)
- Algunos lo incluyen sólo a nivel sintáctico, pero no implementan la semántica asociada (Oracle6)
- Otros permiten crear un procedimiento (almacenado en el catálogo) que implementa cada clave ajena

- El mecanismo de Integridad Referencial penaliza los tiempos de respuesta del sistema
  - importante en consultas interactivas, sobre todo.
  - Borrados/Actualizaciones en cascada.
  - Implementación de Integridad Referencial "en diferido".

DLE: **Claves Ajenas**

- La mayoría de productos NO soportan este concepto, entonces...
  - Introducir las restricciones de clave ajena FK como **requisitos de especificación** de programas
  - Especificar como NOT NULL los atributos de FK con nulos no permitidos
  - Mantener la **definición** de cada clave ajena **como comentario** en el catálogo del SGBD **o**, si éste lo soporta, incluir su **definición sintáctica**
  - Utilizar mecanismos de seguridad (GRANT, REVOKE) para **prohibir operaciones de actualización** interactivas **que pueden violar RI referencial**
  - Crear un **procedimiento** que periódicamente **compruebe** y notifique **posibles violaciones** de la Integridad Referencial

DLE: Otros conceptos del Modelo Relacional

- Será necesario crear procedimientos y/o disparadores (*triggers*) que verifiquen las **restricciones de integridad** definidas en la fase de Diseño Lógico Estándar
- Si el SGBD lo permite, se almacenarán en el catálogo del SGBD
- Si no, serán parte de los programas de aplicación
  - Restricciones de integridad como **especificaciones** de procesos