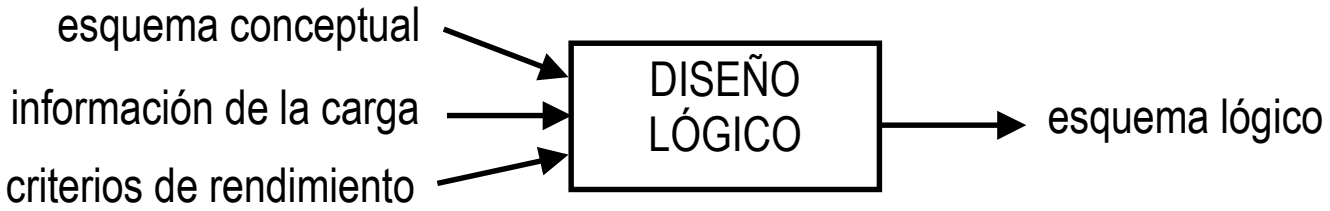
TEMA 7. DISEÑO LÓGICO DE BASES DE DATOS RELACIONALES

1. Introducción
2. Metodología de diseño lógico en el modelo relacional
3. Normalización
4. Desnormalización, partición de relaciones y optimización

1. Introducción

Diseño lógico: conversión del esquema conceptual de datos en un esquema lógico.

Objetivo: obtener una representación que use de la manera más eficiente posible los recursos para la estructuración de datos y el modelado de restricciones disponibles en el modelo lógico.



Información de la carga

- Volumen de la base de datos.
- Conocimiento de consultas y transacciones a realizar, y su frecuencia.

Criterios de rendimiento

- Tiempo de respuesta medio o máximo.
- Espacio de almacenamiento ocupado por la base de datos.
- Utilización de CPU o tiempo de E/S.

2. Metodología de diseño lógico en el modelo relacional

Construir y validar los esquemas lógicos locales para cada vista de usuario

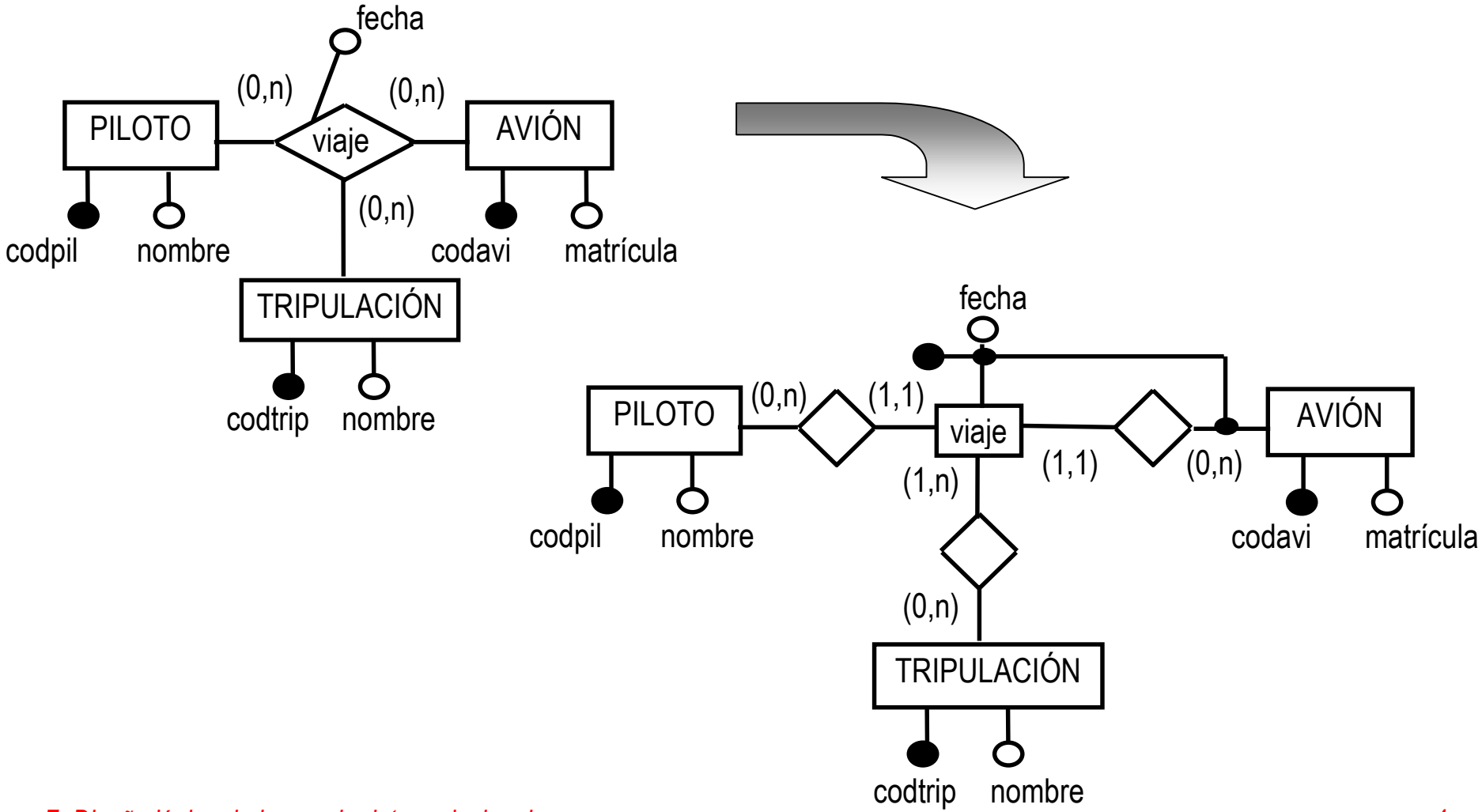
1. Convertir los esquemas conceptuales locales en esquemas lógicos locales.
2. Derivar un conjunto de relaciones (tablas) para cada esquema lógico local.
3. Validar cada esquema mediante la normalización.
4. Validar cada esquema frente a las transacciones del usuario.
5. Dibujar el diagrama entidad – relación.
6. Definir las restricciones de integridad.
7. Revisar cada esquema lógico local con el usuario correspondiente.

Construir y validar el esquema lógico global

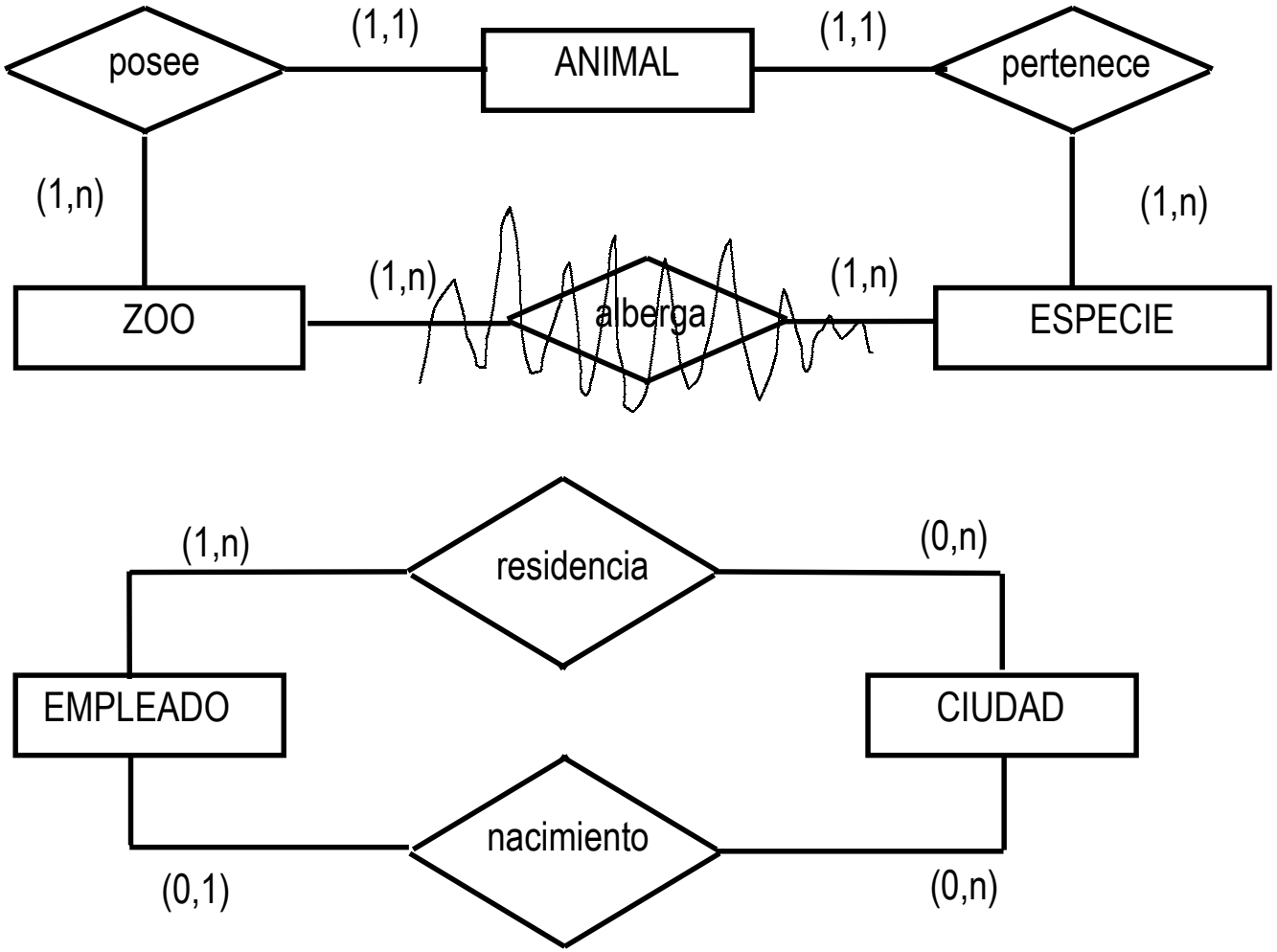
8. Mezclar los esquemas lógicos locales en un esquema lógico global.
9. Validar el esquema lógico global.
10. Estudiar el crecimiento futuro.
11. Dibujar el diagrama entidad/relación final.
12. Revisar el esquema lógico global con los usuarios.

1. Convertir los esquemas conceptuales locales en esquemas lógicos locales

(a) Sustituir cada relación entre tres o más entidades por una entidad intermedia. La cardinalidad de las nuevas relaciones binarias dependerá de su significado. Si la relación sustituida tiene atributos, éstos serán los atributos de la nueva entidad.



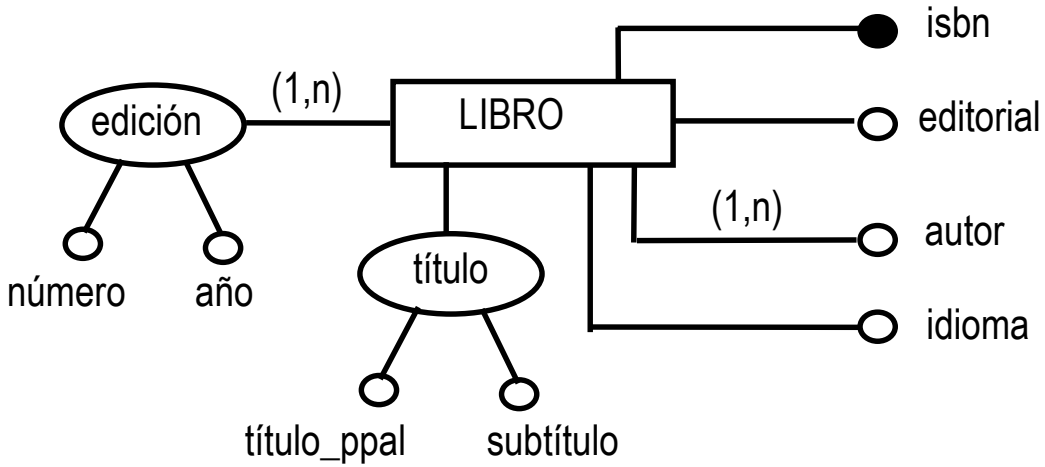
(b) Eliminar las relaciones redundantes.



2. Derivar un conjunto de relaciones para cada esquema lógico local

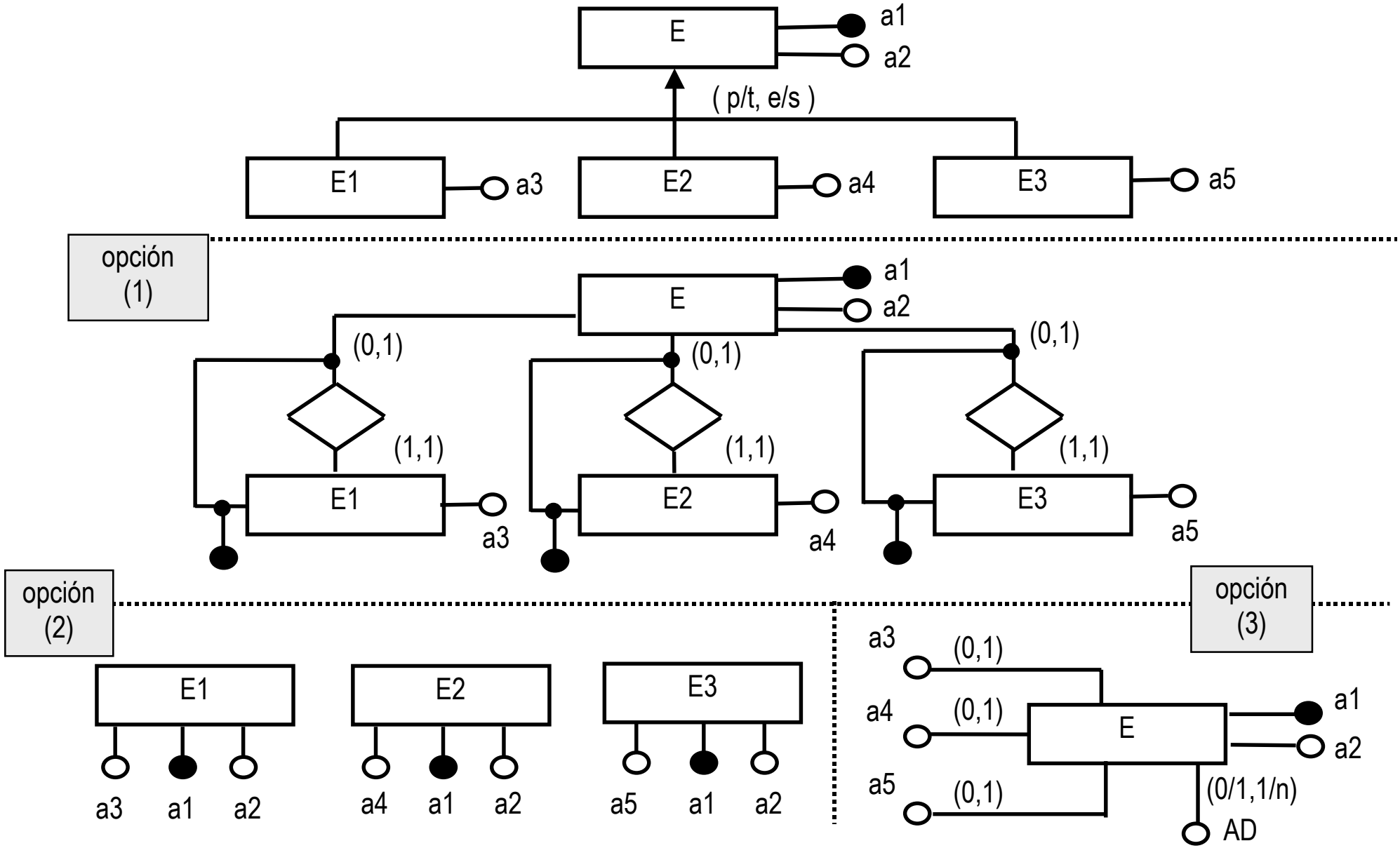
(a) Cada **entidad** del esquema conceptual se transforma en una relación base (tabla).

- Los **atributos** de la entidad se convierten en los atributos de la tabla.
- Cada componente de un **atributo compuesto** se convierte en un atributo de la tabla.
- Por cada **atributo con cardinalidad máxima mayor que uno** se incluye una tabla dentro de la tabla, como un atributo más.
- De entre los **identificadores** de la entidad se debe escoger uno como **clave primaria** de la tabla.



LIBRO(<u>isbn</u>, editorial, AUTOR(<u>autor</u>), idioma, título_ppal, subtítulo, EDICIÓN(<u>número</u>, año))

(b) Hay tres opciones para representar las **jerarquías de generalización**.



(1) Una tabla por cada entidad. Sirve para cualquier tipo de jerarquía (t/p, e/s).

E(<u>a1</u>, a2) , E1(<u>a1</u>, a3) , E2(<u>a1</u>, a4) , E3(<u>a1</u>, a5)

E1.a1, E2.a1, E3.a1 son claves ajenas a E

Nulos	Borrado

(2) Una tabla por cada subentidad. Sólo sirve para jerarquías totales y exclusivas.

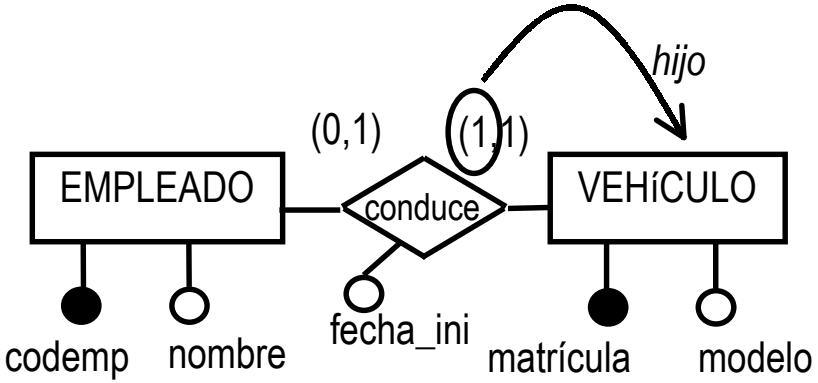
E1(<u>a1</u>, a2, a3) , E2(<u>a1</u>, a2, a4) , E3(<u>a1</u>, a2, a5)

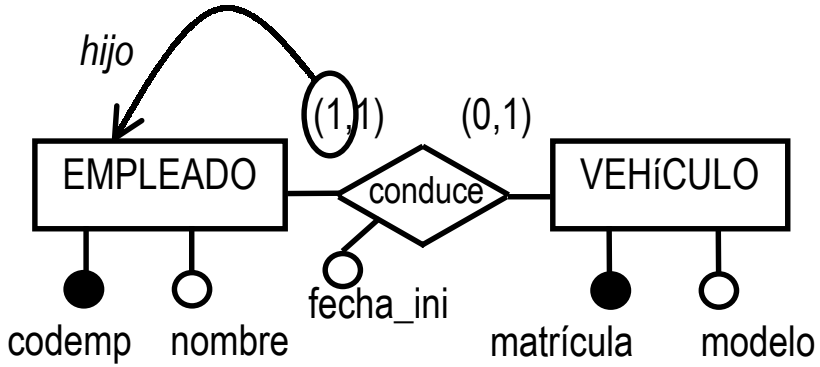
(3) Integrar todas las entidades en una tabla. Sirve para cualquier tipo de jerarquía (t/p, e/s).

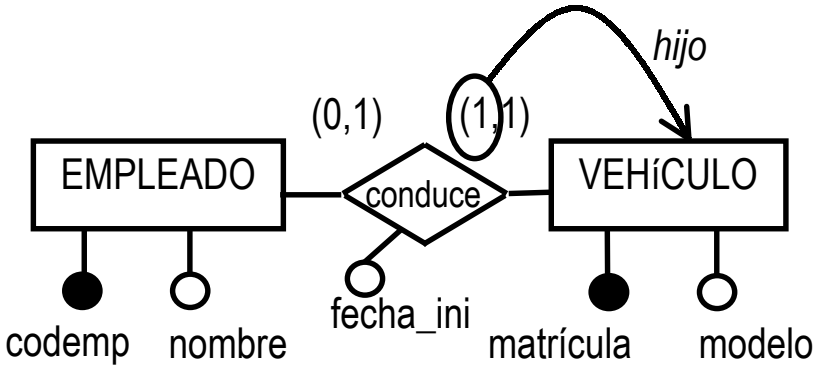
E(<u>a1</u>, a2, a3, a4, a5, tipo) si es exclusiva;
a3, a4, a5 aceptan nulos;
tipo acepta nulos si es parcial.

E(<u>a1</u>, a2, a3, a4, a5, AD(<u>tipo</u>)) si es superpuesta;
a3, a4, a5 aceptan nulos;

(c) Por cada relación binaria (1:1), incluir la clave primaria de la tabla correspondiente a la entidad padre en la tabla de la entidad hijo como una clave ajena. ¿Y los atributos de la relación?







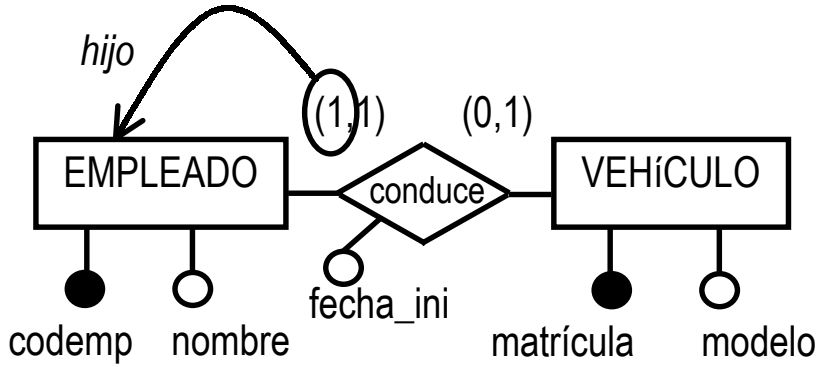
EMPLEADO(<u>codemp</u>, nombre)

¿nulos?

VEHíCULO(<u>matrícula</u>, modelo, codemp, fecha_ini)

VEHíCULO —codemp→ EMPLEADO

Nulos	Borrado



¿son tan diferentes?

VEHíCULO(<u>matrícula</u>, modelo)

¿nulos?

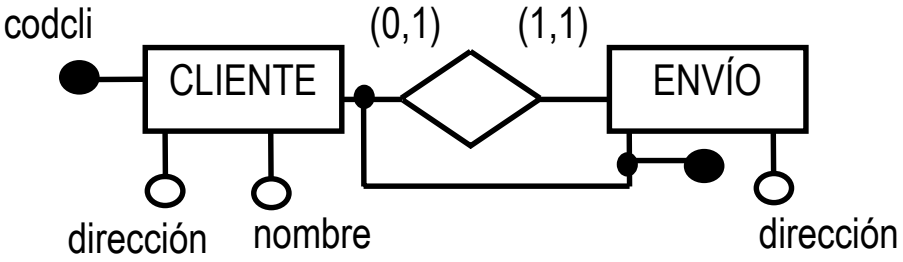
EMPLEADO(<u>codemp</u>, nombre, matrícula, fecha_ini)

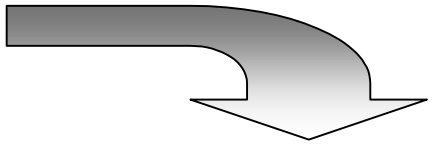
EMPLEADO —matrícula→ VEHíCULO

Nulos	Borrado

¿Y si las dos entidades participan con cardinalidad (0,1)? ¿Y si son ambas (1,1)?

Ojo: Si las entidades relacionadas son sinónimos, integrarlas en una sola tabla.





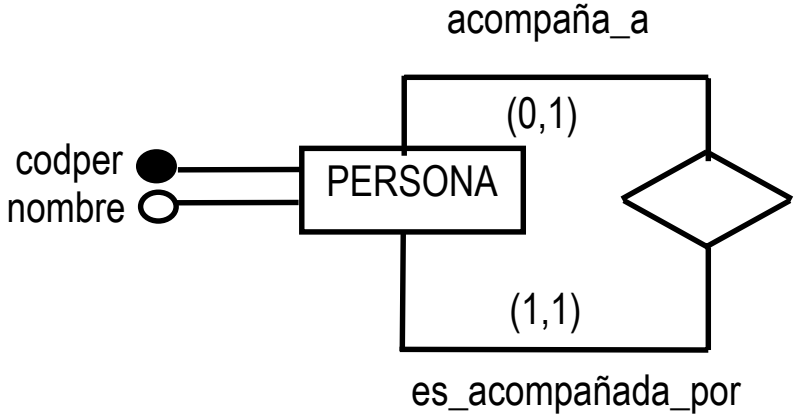
¡¡son sinónimos!!

¿nulos?

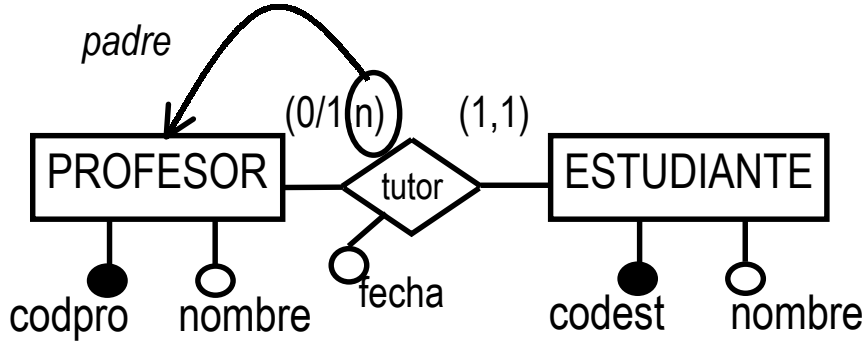
CLIENTE(<u>codcli</u>, dirección, nombre, dirección_envío)

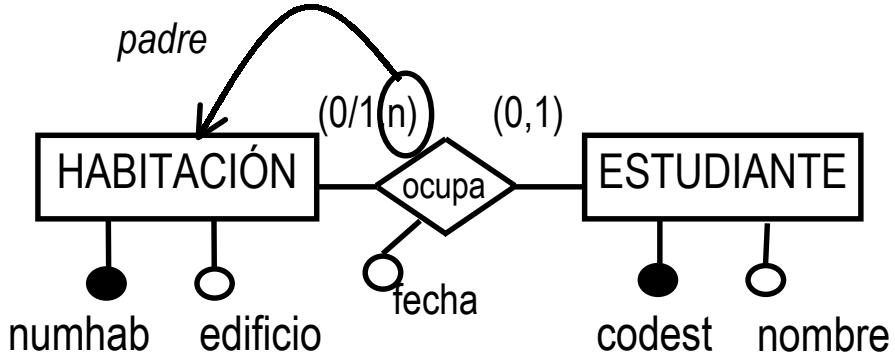
ENVÍO es una **entidad débil** porque no tiene atributos que le sirvan como identificador.

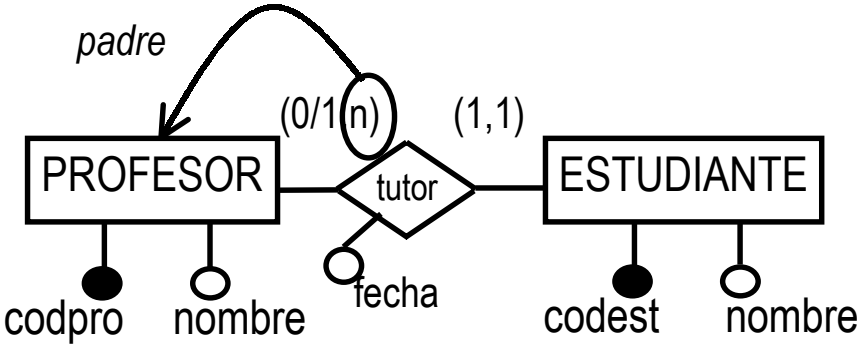
<u>Ejercicio</u>

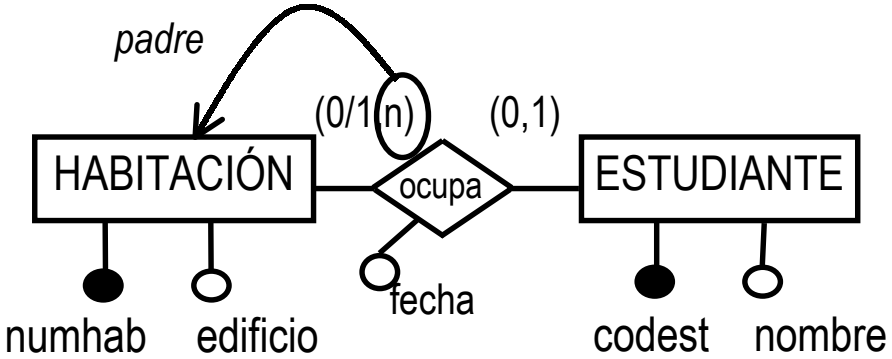


(d) Por cada relación binaria (1:n), incluir la clave primaria de la tabla correspondiente a la entidad padre en la tabla de la entidad hijo (será una clave ajena). ¿Y los atributos de la relación?









PROFESOR(<u>codpro</u>, nombre)

ESTUDIANTE(<u>codest</u>, nombre, codpro, fecha)

¿nulos?

			Nulos	Borrado
ESTUDIANTE	codpro →	PROFESOR		

HABITACIÓN(<u>numhab</u>, edificio)

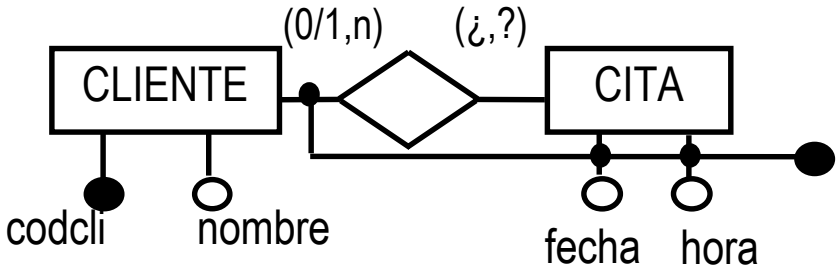
ESTUDIANTE(<u>codest</u>, nombre, numhab, fecha)

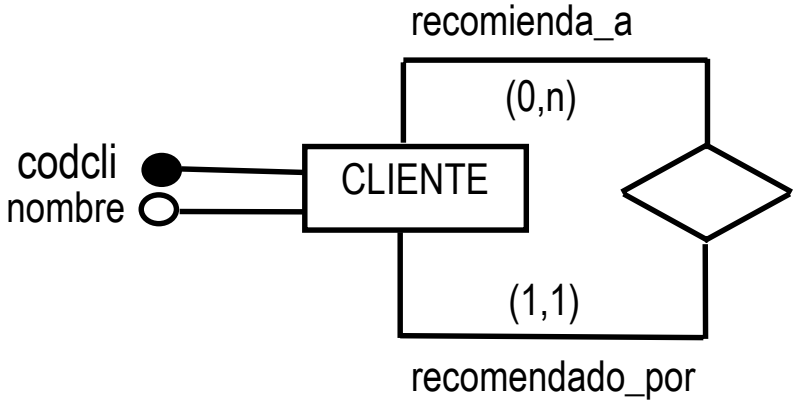
¿nulos?

			Nulos	Borrado
ESTUDIANTE	numhab →	HABITACION		

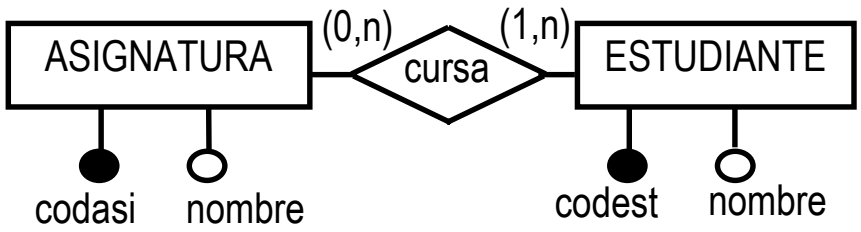
¿Y si hay muy pocos estudiantes que viven en una habitación del campus?

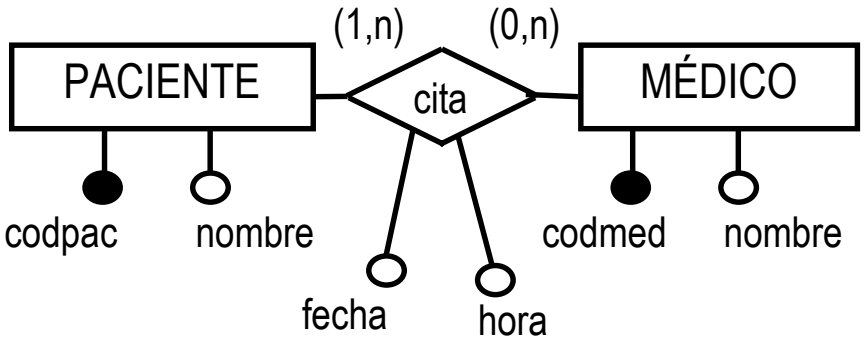
Ejercicios

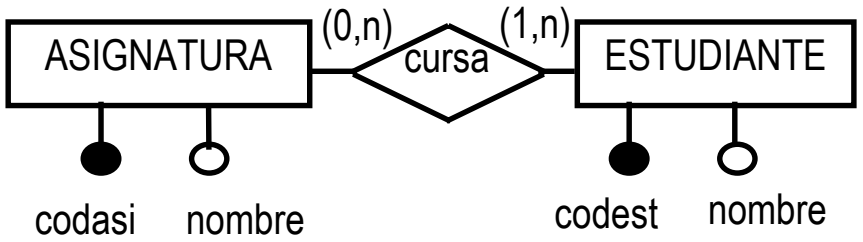

(0/1,n)
(¿,?)
CLIENTE
CITA
codcli
nombre
fecha
hora


recomienda_a
(0,n)
codcli
nombre
CLIENTE
(1,1)
recomendado_por

(e) Por cada relación binaria (m:n), incluir una nueva tabla con una clave ajena a cada una de las tablas correspondientes a las entidades participantes. La clave primaria, la clave primaria ... ¿cuál es la clave primaria? ¿Y los atributos de la relación?





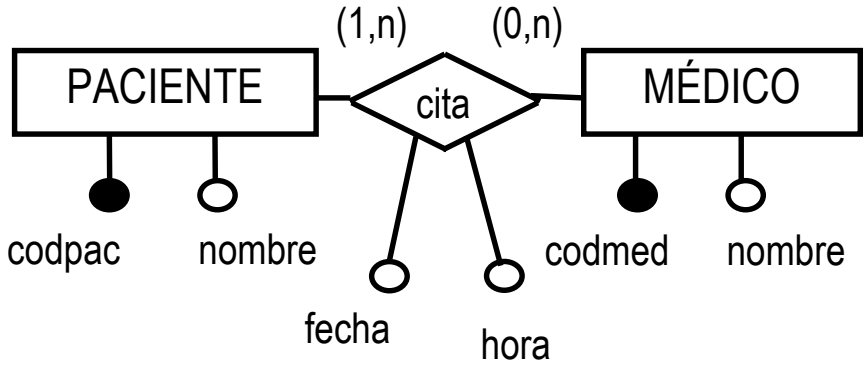


ASIGNATURA(<u>codasi</u>, nombre)

ESTUDIANTE(<u>codest</u>, nombre)

CURSA(<u>codest, codasi</u>)

			Nulos	Borrado
CURSA	codest →	ESTUDIANTE		
CURSA	codasi →	ASIGNATURA		



PACIENTE(<u>codpac</u>, nombre)

MÉDICO(<u>codmed</u>, nombre)

CITA(<u>codmed, fecha, hora</u>, codpac)

			Nulos	Borrado
CITA	codmed →	MÉDICO		
CITA	codpac →	PACIENTE		

Resumen de la correspondencia entre esquemas para las relaciones binarias

	Relación 1:1	***Relación 1:n***	***Relación n:m***
Integrar las dos tablas correspondientes a cada una de las entidades participantes en la relación binaria, en una sola tabla.	Es lo más aconsejable cuando ambas entidades tienen el mismo identificador. Los atributos de la relación binaria también estarán en la tabla. OJO: es posible que algunos atributos deban aceptar nulos.	Para este tipo de relaciones binarias no se puede escoger esta opción.	Para este tipo de relaciones binarias no se puede escoger esta opción.
Poner una clave ajena en la tabla correspondiente a una de las entidades participantes en la relación binaria.	La clave ajena se puede poner en cualquiera de las tablas. La tabla que recibe la clave ajena también recibe los atributos de la relación binaria. OJO: es posible que algunos atributos deban aceptar nulos.	La clave ajena se debe poner en la tabla correspondiente a la entidad que participa en la relación binaria con cardinalidad máxima 1. Los atributos de la relación binaria se ponen como atributos en la tabla que recibe la clave ajena. OJO: es posible que algunos atributos deban aceptar nulos.	Para este tipo de relaciones binarias no se puede escoger esta opción.
Añadir al esquema una nueva tabla en la que se refleje la relación binaria.	Es lo más aconsejable cuando ambas entidades participan en la relación de forma opcional y hay pocas ocurrencias de la misma. Esta nueva tabla tiene una clave ajena a cada una de las dos tablas y también los atributos de la relación binaria.	La nueva tabla tiene una clave ajena a cada una de las dos tablas y también los atributos de la relación binaria. La clave primaria de la nueva tabla será la clave ajena que hace referencia a la tabla de la entidad que participa en la relación binaria con cardinalidad máxima 1.	Esta nueva tabla tiene una clave ajena a cada una de las dos tablas y también los atributos de la relación binaria. La clave primaria variará según el significado de la relación binaria (hay que "meditarla").

Continuamos con la metodología de diseño lógico ...

3. Validar cada esquema lógico local mediante la normalización.

4. Validar cada esquema frente a las transacciones del usuario.

5. Dibujar el diagrama entidad – relación.

6. Definir las restricciones de integridad.

   (a) Datos requeridos.

   (b) Restricciones de dominios.

   (c) Integridad de entidades.

   (d) Integridad referencial.

      (1) Regla de los nulos (Sí admite / No admite).
      (2) Regla del borrado (Restringir / Propagar / Anular).
      (3) Regla de la modificación (Restringir / Propagar / Anular).

   (e) Reglas de negocio.

Continuamos con la metodología de diseño lógico ...

7. Revisar cada esquema lógico local con el usuario.

Utilizar los DFD para comprobar la consistencia y completitud de los esquemas lógicos.

8. Mezclar los esquemas lógicos locales en un esquema lógico global.

9. Validar el esquema lógico global.

10. Estudiar el crecimiento futuro.

11. Dibujar el diagrama entidad/relación final.

12. Revisar el esquema lógico global con los usuarios.

3. Normalización

- Técnica para diseñar bases de datos relacionales.
- Se debe a Codd (1972).
- No se utiliza como una estrategia de diseño de bases de datos.
- Se utiliza para verificar esquemas relacionales.

Ventajas

- ✓ Evita anomalías en inserciones, modificaciones y borrados.
- ✓ Mejora la independencia de datos.

Fecha: 16/2/99

Pedido nº: 123456
Proveedor nº: 9876
Nombre del proveedor: Productos Surtidos
Dirección del proveedor: Borriol, Castellón

Deseamos envíen:

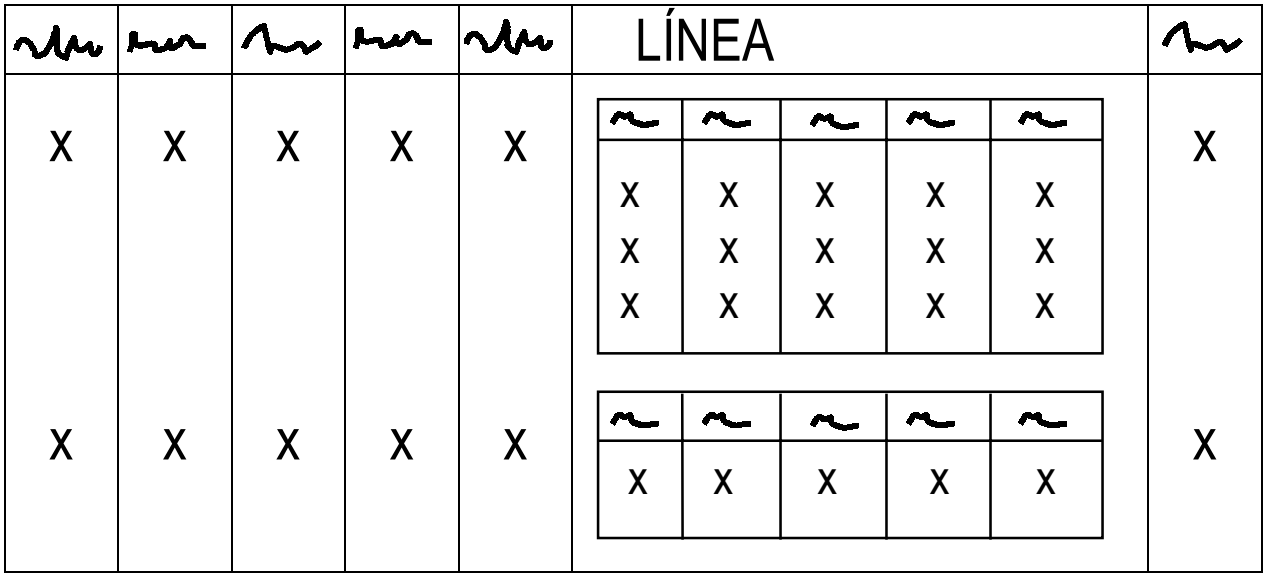
Número producto	Descripción	Precio unitario	Cantidad	Total
511246	Televisión	70.000	1	70.000
124456	Clavija antena	100	10	1.000
124763	Enchufe	150	10	1.500

Importe total: 72.500

PEDIDO (<u>npedido</u>, nprov, nomprov, dirprov, fecha,

LÍNEA (<u>nproducto</u>, descrip, precio, cant, total), importe)

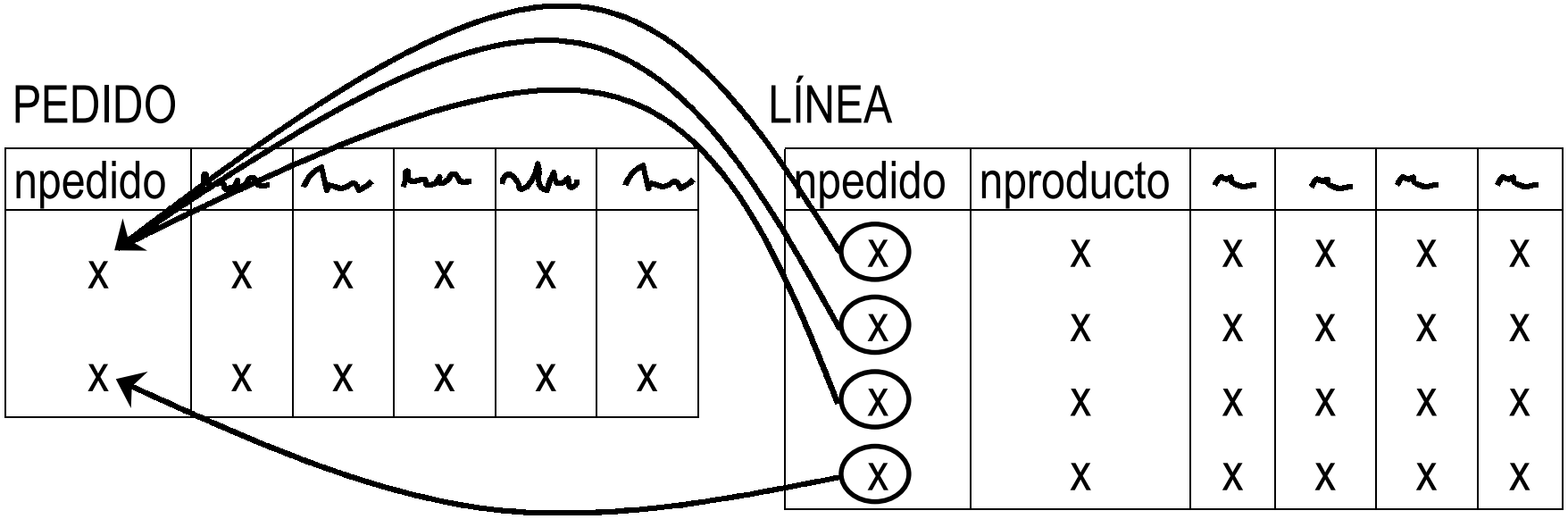
PEDIDO



Hay atributos que tienen valores de tipo relación (tabla).

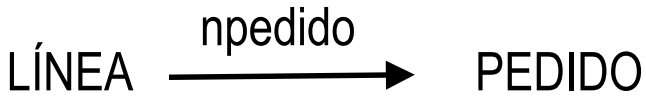
PEDIDO (<u>npedido</u>, nprov, nomprov, dirprov, fecha, importe)

LÍNEA (<u>npedido, nproducto</u>, descrip, precio, cant, total)



PEDIDO (<u>npedido</u>, nprov, nomprov, dirprov, fecha, importe)

LÍNEA (<u>npedido, nproducto</u>, descrip, precio, cant, total)



- Guardar nuevo producto.
  Producto nº 511944, Reproductor de vídeo, 35.000 pesetas.
- Modificar el precio de un producto.
  Producto nº 511246, Televisión, 68.000 pesetas.
- Eliminar la única compra de un producto:
  Producto nº 124763, Enchufe, 150 pesetas.

¡Anomalías en las actualizaciones de datos!

PEDIDO (<u>npedido</u>, nprov, nomprov, dirprov, fecha, importe)

LÍNEA (<u>npedido, nproducto</u>, cant, total)

PRODUCTO (<u>nproducto</u>, descrip, precio)

- Guardar nuevo proveedor.
  Proveedor nº 5194, Don Proveedor, Játiva.
- Modificar la dirección de un proveedor.
  Proveedor nº 9876, Productos Surtidos, Castellón de la Plana.
- Eliminar la única compra realizada a un proveedor.

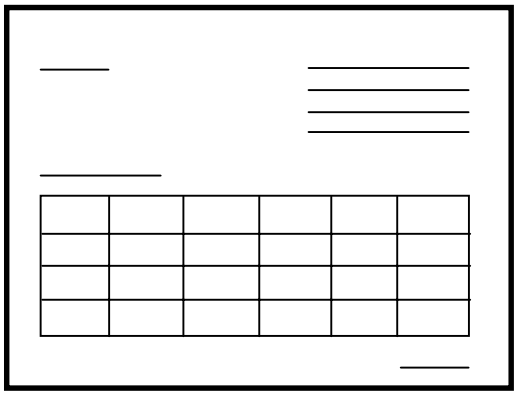
¡Anomalías en las actualizaciones de datos!

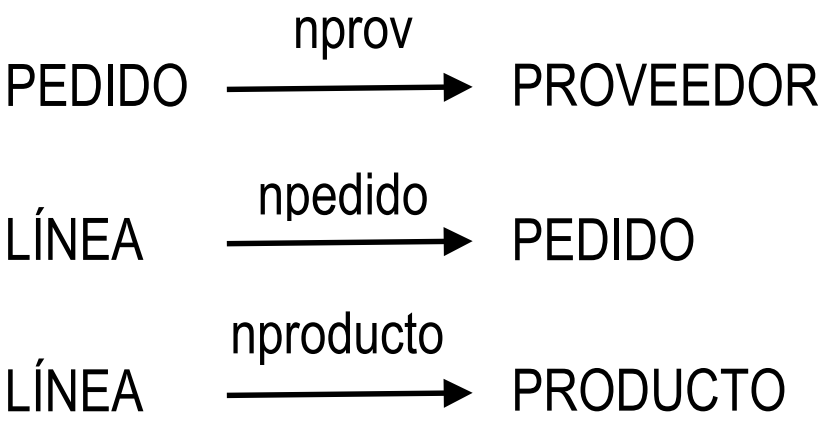
PEDIDO (<u>npedido</u>, nprov, fecha, importe)

LÍNEA (<u>npedido, nproducto</u>, precio, cant, total)

PRODUCTO (<u>nproducto</u>, descrip, precio)

PROVEEDOR (<u>nprov</u>, nomprov, dirprov)





Dependencia funcional

Y es ***funcionalmente dependiente*** de **X,** si **X** determina el valor de **Y**: **X** → **Y**

Ejemplo: CLIENTE(codcli, nombre, codpostal, población)

codpostal → población

Observaciones

- La ***dependencia funcional*** es una noción semántica.
- Cada ***dependencia funcional*** es una clase especial de regla de integridad.
- Cada ***dependencia funcional*** representa una relación de uno a muchos.

Primera forma normal (1FN)

Una relación está en **1FN** si, y sólo si, todos sus dominios contienen valores atómicos.

PRODUCTO

codprod	nombre	VERSIÓN		
		número	fecha	ventas
LH4	Ladrillo hueco	1	1/3/1996	30.000
		2	1/8/1998	50.000
		3	1/2/2000	13.000
LP7	Ladrillo perforado	1	1/6/1996	70.000
		2	1/12/2000	

grupos repetitivos (valores no atómicos)

PRODUCTO (codprod, nombre, VERSIÓN (número, fecha, ventas)) ~~1FN~~

Se descompone en:

PRODUCTO (codprod, nombre, descripción)

VERSIÓN (codprod, número, fecha, ventas)

hereda la clave primaria

OJO

VERSIÓN —codprod→ PRODUCTO

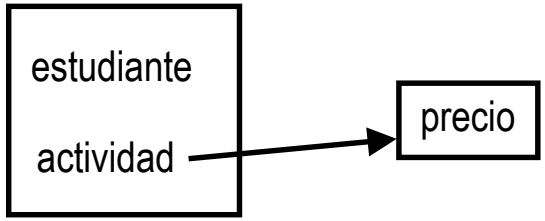
Nulos Borrado

Segunda forma normal (2FN)

Una relación está en **2FN** si, y sólo si, está en 1FN y, además, cada atributo no clave depende completamente de la clave primaria (no depende de algún subconjunto).

INSCRIPCIÓN (estudiante, actividad, precio) ~~2FN~~

actividad → precio



estudiante	actividad	precio
100	Tenis	1500
100	Yoga	1500
200	Tenis	1500
300	Escalada	5000

misma actividad, mismo precio.

Se descompone en las proyecciones:

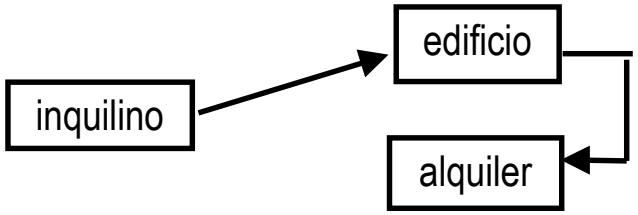
INSCRIPCIÓN (estudiante, actividad) y **ACTIVIDAD (actividad, precio)**

Tercera forma normal (3FN)

Una relación está en **3FN** si, y sólo si, está en 2FN y, además, cada atributo no clave no depende transitivamente de la clave primaria.

INQUILINO (inqulino, edificio, alquiler) ~~3FN~~

edificio → alquiler



inquilino	edificio	alquiler
100	E04	50.000
200	E13	50.000
300	E09	65.000
400	E04	50.000

mismo edificio, mismo alquiler.

Se descompone en las proyecciones:

INQUILINO (inqulino, edificio) **y** **EDIFICIO (edificio, alquiler)**

INQUILINO —edificio→ EDIFICIO

Nulos ☐ Borrado ☐

Ejercicio de normalización

estudiante	nombre	apellido	DNI	dirección	codbeca	nombeca	requisito	fecha
0123	Carlos	Gil	159357	C/ Paz, 23	A223	EEUU	Ing. Sup.	10/10/98
7636	Paula	Tena	913752	C/ Río Po, 1	B567	ERASMUS	Ing. Téc.	12/11/98
7636	Paula	Tena	913752	C/ Río Po, 1	A223	EEUU	Ing. Sup.	14/10/98
7636	Paula	Tena	913752	C/ Río Po, 1	G654	DRAC	Ing. Sup.	15/09/99
0123	Carlos	Gil	159357	C/ Paz, 23	G654	DRAC	Ing. Sup.	17/09/98
9516	Andrés	Calpe	682432	Plz. Sol, 40	G654	DRAC	Ing. Sup.	12/09/99
0123	Carlos	Gil	159357	C/ Paz, 23	B567	ERASMUS	Ing. Téc.	12/11/98
9516	Andrés	Calpe	682432	Plz. Sol, 40	B567	ERASMUS	Ing. Téc.	23/11/99
0123	Carlos	Gil	159357	C/ Paz, 23	A223	EEUU	Ing. Sup.	12/10/99
3361	Lucía	Porcar	243115	Plz. Sol, 26	A223	EEUU	Ing. Sup.	12/10/99
...	...	...	...	...	...	...	...	...

SOLICITUD (<u>estudiante, codbeca, fecha</u>, nombre, apellido, DNI, dirección, nombeca, requisito)

4. Desnormalización, partición de relaciones y optimización

A partir del esquema lógico obtenido y teniendo en cuenta el modelado de la carga ...

- Se pueden fundir varias relaciones en una si se usan juntas con frecuencia mediante operaciones de JOIN → **Desnormalización.**

- Se pueden dividir algunas relaciones con el objeto de reorganizar la distribución de los casos → **Partición Horizontal**, o de los atributos → **Partición Vertical**, de manera que una relación incluya atributos o casos a los que se requiera acceso simultáneo con frecuencia.

- Se pueden introducir otros cambios para conseguir un acceso más eficiente → **Optimización.**

Desnormalización

Por ejemplo, se pueden fusionar las relaciones:

CLIENTE(codcli, nombre, codpostal) y **CODPOSTAL(codpostal, codpueblo)**

en una sola relación: **CLIENTE(codcli, nombre, codpostal, codpueblo)**

Así se mejora el funcionamiento frente a la necesidad de hacer el JOIN de las dos tablas. Se notará más la mejora cuanto más frecuentes sean los accesos. Pero mucho OJO: se han introducido redundancias que ahora será necesario controlar ¿alguna idea sobre cómo hacerlo?

Partición de tablas

Por ejemplo, se puede descomponer la siguiente relación:

EMPLEADO(codemp, nombre, teléfono, fecha_eval, aspecto1, aspecto2)

en las relaciones:

EMPLEADO(codemp, nombre, teléfono)
EVALUACION(codemp, fecha_eval, aspecto1, aspecto2)

porque no se accede con frecuencia a los datos de la evaluación de los empleados, o bien porque se quiere preservar la seguridad de los mismos. ¿Y qué hacemos para el usuario que necesita ver la tabla tal y como estaba?

Optimización

UNIVERSIDAD(universidad, director, vicedirector)

Cada universidad tiene un director y de uno a tres vicedirectores ¿clave primaria?

Hay una dependencia funcional no deseada:

universidad ⟶ director

UNIVERSIDAD no se encuentra en 2FN → debe descomponerse en:

UNIVERSIDAD(<u>universidad</u>, director)

ASISTENTE (<u>universidad, vicedirector</u>)

Siempre que una aplicación necesite información de la universidad, debe leer entre dos y cuatro filas de datos.

Una alternativa que consigue mayor eficiencia es:

UNIVERSIDAD(<u>universidad</u>, director, vicedirector1, vicedirector2, vicedirector3)