7 Diseño de Bases de Datos Relacionales: Normalización

7.1 Problemas derivados del diseño de una Base de Datos Relacional

7.2 Dependencias funcionales. 1ª, 2ª y 3ª Formas Normales

7.3 Dependencias multivaluadas y 4ª Forma Normal

7.4 Otras Formas Normales y ejemplos de diseño

¡ ES SÓLO UNA INTRODUCCIÓN PRÁCTICA !

introducción al problema de diseño de una B.D. Relacional

Metodología de Diseño $\Rightarrow$

- Diseño Conceptual $\rightarrow$ esquema E/R
- reglas transformación
- Diseño Lógico $\rightarrow$ esquema Relacional

buen diseño conceptual y transformaciones correctas y adecuados $\Rightarrow$ ***buen diseño de B.D. Relacional***

- ☛ *¿ Cómo saber si el diseño es "bueno" ?*
- ☛ *¿ Qué problemas ocasiona un "mal" diseño ?*

→ ***no debe haber <u>redundancias</u> en la información***

ejemplo de diseño de una B.D. Relacional: enunciado

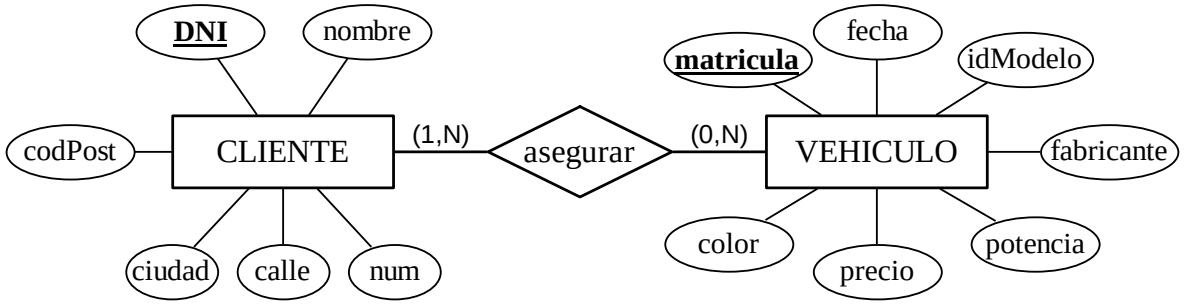
La compañía de seguros PREVISORA necesita una Base de Datos para guardar la información de los vehículos asegurados y de sus propietarios, así como de clientes potenciales (no tienen ningún vehículo asegurado).
De momento, sólo necesita tener de cada vehículo su número de matrícula, fecha de matriculación, precio, modelo, potencia, color y fabricante.
En cuanto a los propietarios, desea guardar su nombre completo, DNI, y dirección (calle, número, ciudad y distrito postal).
Evidentemente, es posible que un cliente tenga asegurados varios vehículos, y que algunos vehículos figuren a nombre de varios propietarios.

Diseñar una Base de Datos relacional conforme a las especificaciones anteriores.

primer intento: esquemas E/R y relacional

1ª propuesta:

hecha en clase por los alumnos



Cliente = (**DNI**: tpDNI; nombre: tpNombre *NO NULO*; ciudad : tpNombre *NO NULO*;
calle: tpNombre *NO NULO*; num: entero *NO NULO*; codPost : entero *NO NULO*);

Vehiculo = (**matricula**: tpMatricula; fecha: tpFecha *NO NULO*; idModelo: tpNombre *NO NULO*;
fabricante: tpNombre *NO NULO*; potencia: entero *NO NULO*;
precio : entero *NO NULO;* color : tpColor);

Asegurar = (**idVehiculo** : tpMatricula; **idCliente**: tpDNI);
clave ajena (idVehiculo), *referencia a* Vehiculo; *borrado* en cascada
clave ajena (idCliente), *referencia a* Cliente; *borrado* en cascada

✓ *Verificar que* $\forall$ *ocurrencia de* matricula *en* vehículo $\exists$ *en* Asegurar $\equiv\not\exists$ $\prod_{matricula}$(Vehiculo) - $\prod_{idVehiculo}$(Asegurar)

• • •

ejemplo de instancia de la B.D.: problemas que aparecen

ejemplo de instancia

cliente

DNI	Nombre	calle	nº	Ciudad	DP
15873564	Muchas Pelas	Principal	25	Zaragoza	50036
25654758	Normal Ito	Mayor	18	Huesca	22022
12365451	Sin Coche	Centro	1	Zaragoza	50051
13655665	Pocas Pelas	Sevilla	33	Teruel	44002
22334455	Aprov Echado	Zaragoza	5	Madrid	28028

asegurar

DNI	matricula
15873564	Z-2345-ZT
15873564	H-2324-AA
15873564	B-2456-HJ
12365451	Z-1234-B
13655665	T-65342
22334455	T-65342

vehículo

matricula	modelo	pot.	fecha	precio	fabricante	color
Z-2345-ZT	Senator Luxe Top	125	22/3/01	15.000.000	GM	dorado
H-2324-AA	Espace VX	90	14/5/01	6.000.000	Renault	gris metal
B-2456-HJ	Senator Luxe Top	125	15/11/00	14.000.000	GM	verde
Z-1234-B	Xara JR	65	6/6/95	2.500.000	Citroen	rojo
T-65342	Fiesta 1000	50	22/9/88	1.800.000	Ford	verde

☛ ***se repite para cada modelo de coche la información básica asociada (potencia, fabricante)***

- ✓ *aumenta (innecesariamente) la ocupación de memoria*
- ✓ *posibilidad de inconsistencias en la inserción y actualización*
- ✓ *pérdidas de información (anomalías) en el borrado*

no se puede guardar la información de modelos de coches sin matricular

al eliminar el vehículo `T-65342` *se pierde la información de que* `Fiesta 1000` *es un coche de* `Ford` *de* `50 C.V.`

otros problemas de un mal diseño de una B.D. Relacional

✓ *todavía habría sido peor la solución basada en la relación "universal"*

BD_Seguros

DNI	Nombre	calle	nº	Ciudad	DP	matricula	modelo	pot.	fecha	precio
15873564	Muchas Pelas	Principal	25	Zaragoza	50036	Z-2345-ZT	Senator Luxe Top	125	22/3/01	15.000.000
15873564	Muchas Pelas	Principal	25	Zaragoza	50036	H-2324-AA	Espace VX	90	14/5/01	6.000.000
15873564	Muchas Pelas	Principal	25	Zaragoza	50036	B-2456-HJ	Senator Luxe Top	125	15/11/00	14.000.000
25654758	Normal Ito	Mayor	18	Huesca	22022	Z-1234-B	Xara JR	65	6/6/95	2.500.000
12365451	Sin Coche	Centro	1	Zaragoza	50051					
13655665	Pocas Pelas	Sevilla	33	Teruel	44002	T-65342	Fiesta 1000	50	22/9/88	1.800.000
22334455	Aprov Echado	Zaragoza	5	Madrid	28028	T-65342	Fiesta 1000	50	22/9/88	1.800.000

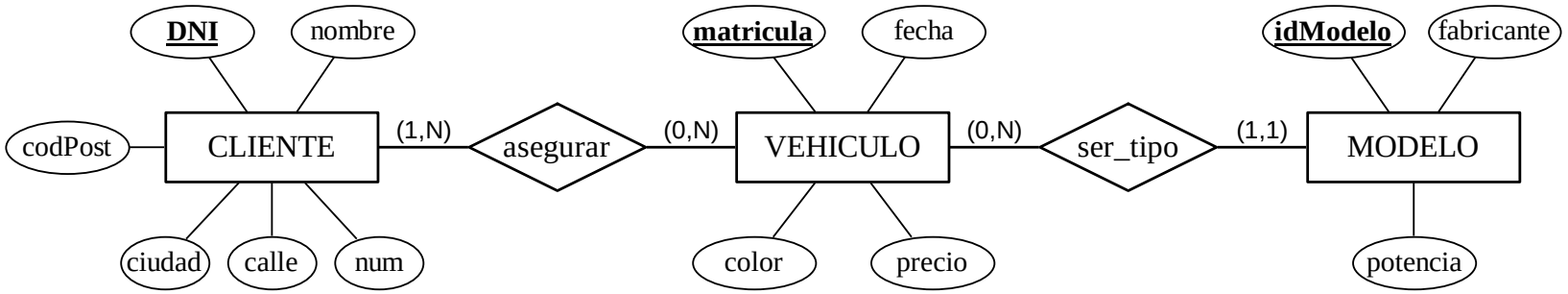
☛ ***además de los problemas anteriores, aparecen los problemas ligados a los valores NULOS***

✓ *definición de claves → atributo "especial" (contador)*
✓ *inconsistencias adicionales (si el señor "Sin Coche", se compra un vehículo ¿?)*
• • •

la causa del error es que hay atributos que proporcionan información de otros atributos ⇒ *éstos deberían ser tipos de entidad*

segundo intento: esquemas E/R y relacional

2ª propuesta:



Cliente = (**DNI**: tpDNI; nombre: tpNombre *NO NULO*; ciudad : tpNombre *NO NULO*;
calle: tpNombre *NO NULO*; num: entero *NO NULO*; codPost : entero *NO NULO*);

Vehiculo = (**matricula**: tpMatricula; fecha: tpFecha *NO NULO*; precio : entero *NO NULO;*
color : tpColor *NO NULO*; idModelo: tpNombre *NO NULO, clave ajena* de Modelo);

Modelo = (**idModelo**: tpNombre; fabricante: tpNombre *NO NULO*; potencia: entero *NO NULO*);

Asegurar = (**idVehiculo** : tpMatricula; **idCliente**: tpDNI);
clave ajena (idVehiculo), *referencia a* Vehiculo; *borrado* en cascada
clave ajena (idCliente), *referencia a* Cliente; *borrado* en cascada

Verificar que $\nexists$ ($\Pi_{matricula}$(Vehiculo) - $\Pi_{idVehiculo}$(Asegurar))

ejemplo de instancia de la B.D.: conclusiones de diseño

cliente

DNI	Nombre	calle	nº	Ciudad	DP
15873564	Muchas Pelas	Principal	25	Zaragoza	50036
25654758	Normal Ito	Mayor	18	Huesca	22022
12365451	Sin Coche	Centro	1	Zaragoza	50051
13655665	Pocas Pelas	Sevilla	33	Teruel	44002
22334455	Aprov Echado	Zaragoza	5	Madrid	28028

asegurar

DNI	matricula
15873564	Z-2345-ZT
15873564	H-2324-AA
15873564	B-2456-HJ
12365451	Z-1234-B
13655665	T-65342
22334455	T-65342

vehículo

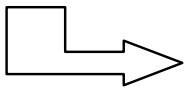
matricula	modelo	fecha	precio	color
Z-2345-ZT	Senator Luxe Top	22/3/01	15.000.000	dorado
H-2324-AA	Espace VX	14/5/01	6.000.000	gris metal
Z-1234-B	Xara JR	6/6/95	2.500.000	rojo
T-65342	Fiesta 1000	22/9/88	1.800.000	verde
B-2456-HJ	Senator Luxe Top	15/11/00	14.000.000	verde

modelo

idModelo	pot.	fabricante
Senator Luxe Top	125	GM
Espace VX	90	Renault
Xara JR	65	Citroen
Fiesta 1000	50	Ford

evidentemente, esta solución es mejor (aunque hay problemas con el Código Postal)

¿¿ sería posible llegar a ella (incluso mejorarla) a partir de las anteriores ??

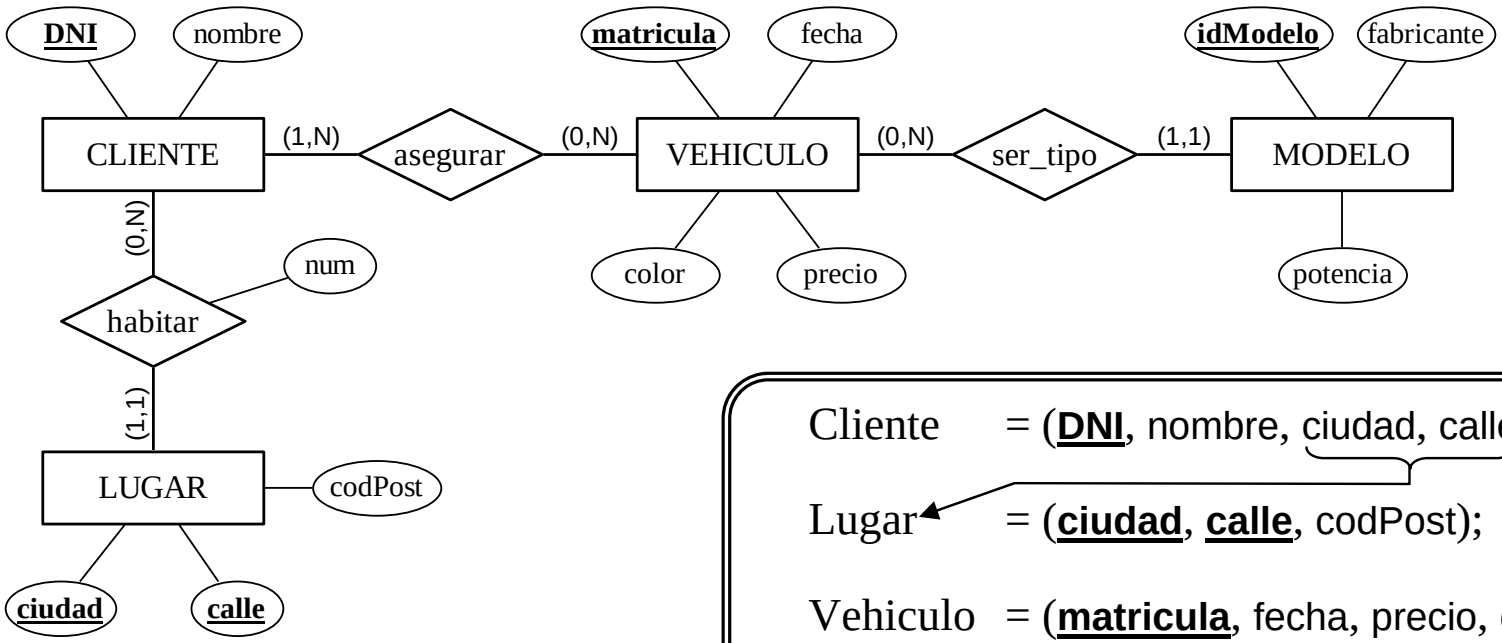


***SI**: realizando una descomposición adecuada de las relaciones originales*

solución final: esquemas E/R y relacional

pensando un poco más, podría haberse llegado a la propuesta "casi" definitiva:

codPost es información asociada a (ciudad,calle)



Cliente = (**DNI**, nombre, ciudad, calle, num);

Lugar = (**ciudad**, **calle**, codPost);

Vehiculo = (**matricula**, fecha, precio, color, idModelo);

Modelo = (**idModelo**, fabricante, potencia);

Asegurar = (**idVehiculo, idCliente**);

Nota: *se ha supuesto, para simplificar, que el código postal depende sólo de la ciudad y de la calle, no del número de la calle.*

planteamientos básicos de una metodología de diseño

Objetivo: ***eliminación de redundancias*** *(anomalías en inserción y eliminación)* ⇨ solución: ***descomposición***

ejemplo:

B1 = (DNI, nombre, calle, numero, codPost);
B2 = (codPostal, ciudad);
B3 = (matricula, fecha, modelo, color, precio);
B4 = (modelo, fabricante, potencia);

→ *no hay (casi) redundancias, pero* ***¡¡¡ se ha perdido información !!!***

☛ ***no es posible reconstruir la relación original***

Sea *R* una relación cualquiera, y *R1*, *R2*, .., *Rn* una *descomposición* de *R*
Se dice que la *descomposición* de *R* es ***sin pérdida***, si R = R1 ⋈ R2 … ⋈ Rn

⇨ *sólo se realizarán* ***descomposiciones sin pérdida***

Primera Forma Normal

Objetivo: ***obtener "buenas" descomposiciones**, a partir de las propiedades de la información*

Teoría de la Normalización

una relación R *está en **1FN** si todo atributo toma un único valor para cualquier ocurrencia de* R.

→ *≡ no hay grupos repetitivos*

la transformación de una tabla a una relación en 1FN es trivial

Pb. definición de claves

Punto de partida: una relación (o varias) en Primera Forma Normal

Restricción inherente al modelo relacional

concepto de Dependencia Funcional

Sean: $R = (A_1, A_2, \ldots, A_n)$ *un esquema de relación, y*
$X, Y \subseteq \{A_1, A_2, \ldots, A_n\}$ *dos subconjuntos de atributos de* R

Se dice que X ***determina*** Y, *o que* Y ***depende funcionalmente de*** X, $\mathbf{X \rightarrow Y}$
si para todo par de tuplas $t_1, t_2 \in r(R)$, *se verifica que:*

$$\Pi_X(t_1) = \Pi_X(t_2) \Rightarrow \Pi_Y(t_1) = \Pi_Y(t_2)$$

≡ *dado un valor de los atributos del conjunto X, los valores de los atributos del conjunto Y están determinados*

ejemplos:
{ DNI } → { nombre, calle, numero, ciudad }
{ DNI, matricula } → { nombre, color }
{ calle, numero, ciudad } → { codPost }
{ codPost } → { ciudad }
{ matricula } → { modelo, precio }
• • •

propiedades de las Dependencias Funcionales

☛ *dependencia funcional es una generalización del concepto de superclave*

K *es* ***superclave*** *de* **R** **sii** $K \rightarrow R$

toda superclave mínima es una clave candidata

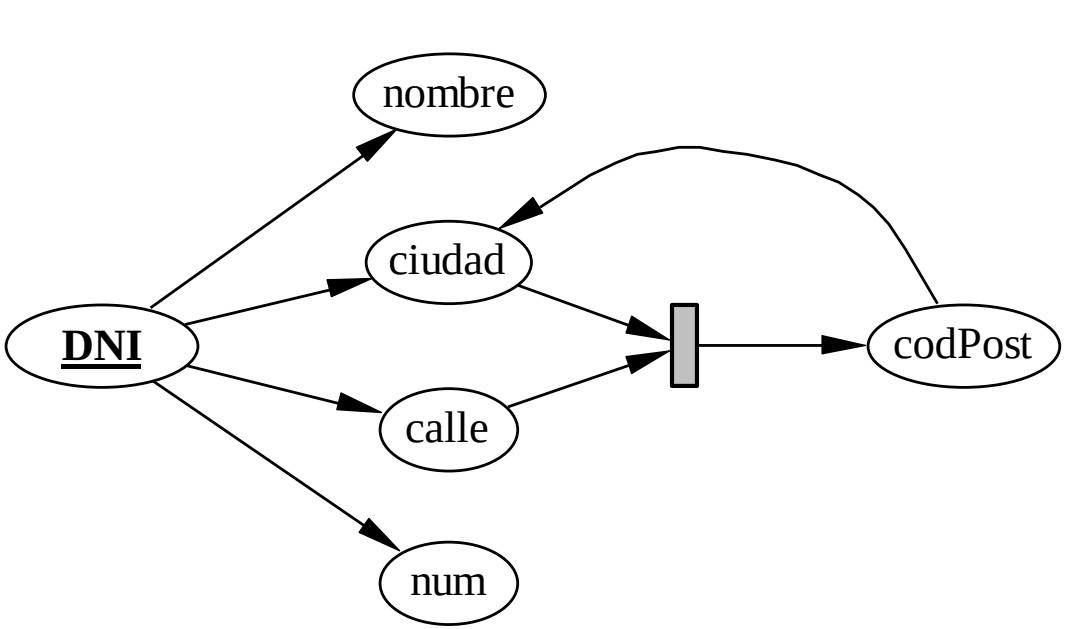
propiedades: ***axiomas de Armstrong***

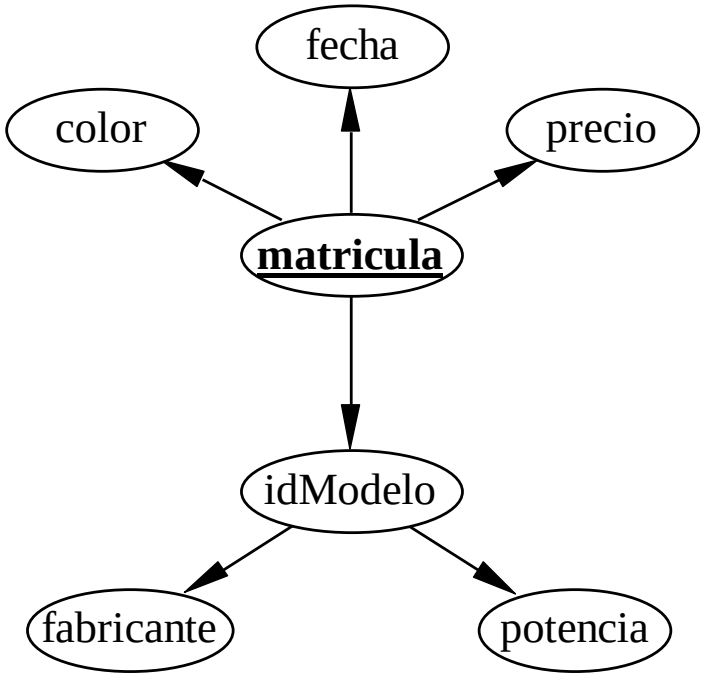
Reflexiva :	$\forall \; Y \subseteq X \Rightarrow X \rightarrow Y$
Aumento :	$X \rightarrow Y \Rightarrow ZX \rightarrow ZY$
Transitiva :	$X \rightarrow Y$ *y* $Y \rightarrow Z \Rightarrow X \rightarrow Z$

facilitan la obtención de todas las dependencias funcionales y, por tanto, de las claves candidatas

grafos de Dependencias funcionales

Se puede construir un **grafo de dependencias funcionales** → *proporciona una visión global*





grafo de dependencias funcionales correspondiente a la BD de vehículos

dependencias funcionales y Segunda Forma Normal

Una relación R *está en* 2FN si y sólo si :

1) Está en **1FN**
2) ningún atributo X $\notin$ clave depende funcionalmente de parte de la clave

$\notin$ *a ninguna clave candidata*

todos los atributos que no pertenecen a ninguna clave candidata suministran información de la clave completa

atributos ***no primos***

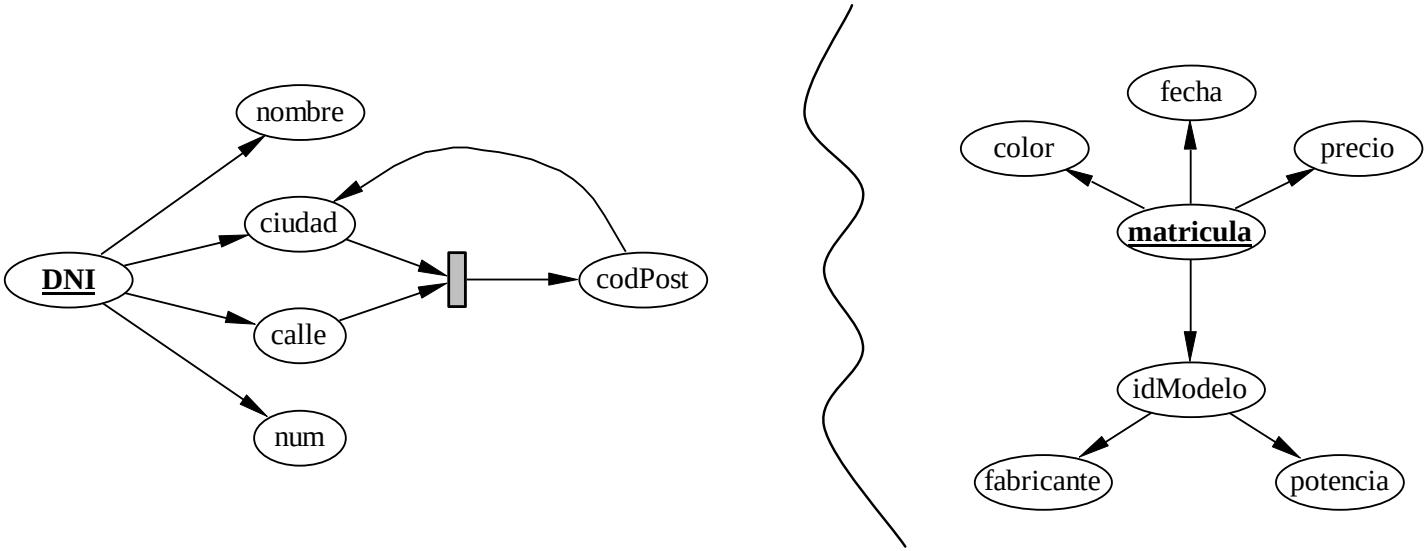
☛ ***toda relación cuyas claves candidatas tienen un único atributo está en 2FN***

Aplicación al diseño $\Rightarrow$ *obtener una descomposición sin pérdida que verifique la 2FN*

- ✓ *una relación con cada uno de los subconjuntos de atributos que no verifican la 2FN*
- ✓ *una relación con la clave primaria y los atributos que dependen de la clave completa*

ejemplo de aplicación de la Segunda Forma Normal

aplicación a la BD de vehículos:



D1 = (DNI, nombre, calle, numero, ciudad, codPost);
D2 = (matricula, fecha, modelo, color, precio, fabricante, potencia);
D3 = (matricula, DNI);

dependencias funcionales y Tercera Forma Normal

Una relación R *está en* 3FN si y sólo si :

1) Está en **2FN**
2) ningún atributo X $\notin$ clave depende funcionalmente de un conjunto de atributos no pertenecientes a la clave → X $\notin$ *a ninguna clave candidata*

☛ *ningún atributo no perteneciente a ninguna clave candidata es transitivamente dependiente de la clave*

☛ *todos los atributos $\notin$ a ninguna clave candidata* ***sólo*** *suministran información de la clave completa*

si y sólo si $\forall$ dependencia funcional $\mathbf{X} \rightarrow \mathbf{A}$ *se verifica una de las siguientes condiciones*:

1) $\mathbf{A} \subseteq \mathbf{X}$ *por lo que* $\mathbf{X} \rightarrow \mathbf{A}$ *es una dependencia funcional trivial*
2) **X** *contiene una clave*
3) **A** *pertenece a una clave candidata*

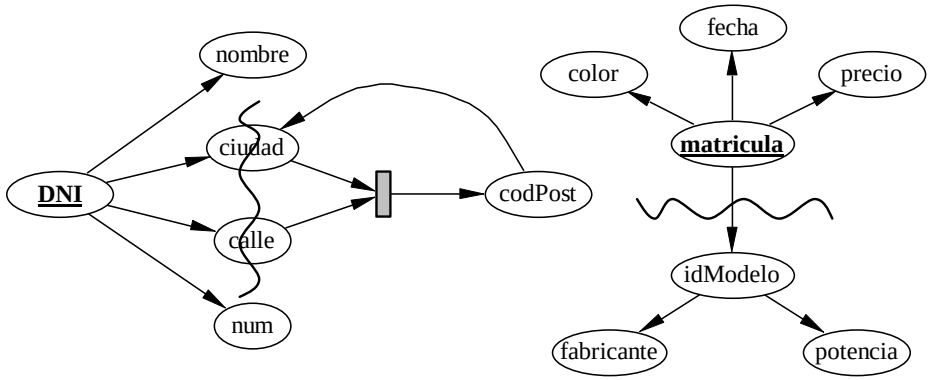
ejemplo de aplicación de la Tercera Forma Normal

***Aplicación al diseño**: obtener una descomposición sin pérdida que verifique la 3FN*

D1 = (DNI, nombre, ciudad, calle, numero, codPost);
D2 = (matricula, fecha, modelo, color, precio, fabricante, potencia);
D3 = (matricula, DNI);

{ciudad, calle} → {codPost}

{modelo} → {fabricante}
{modelo} → {potencia}



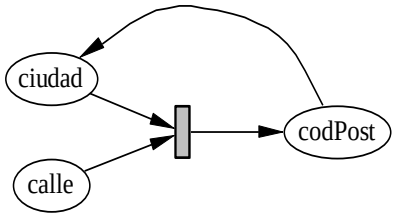
E1 = (DNI, nombre, ciudad, calle, numero);
E2 = (ciudad, calle, codPost);
E3 = (matricula, fecha, modelo, color, precio);
E4 = (modelo, fabricante, potencia);
E5 = (matricula, DNI);

☛ ***toda relación admite, al menos, una descomposición sin pérdida que verifica la 3FN***

Forma Normal de Boyce-Codd (FNBC)

☛ *en la mayor parte de los casos, basta con la 3FN para eliminar las redundancias*

Sólo en algunos casos es necesario aplicar restricciones adicionales → 3FNBC



E2 = (ciudad, calle, codPost);
E2' = (ciudad, calle, codPost);

⇨

F2 = (codPost, calle);
F3 = (codPost, ciudad);

Una relación **R** *está en* Tercera forma normal de Boyce-Codd (3FNBC) si y sólo si :

siempre que se verifica **X → A** , con **A ∉ X,** entonces **X** contiene una clave de **R**

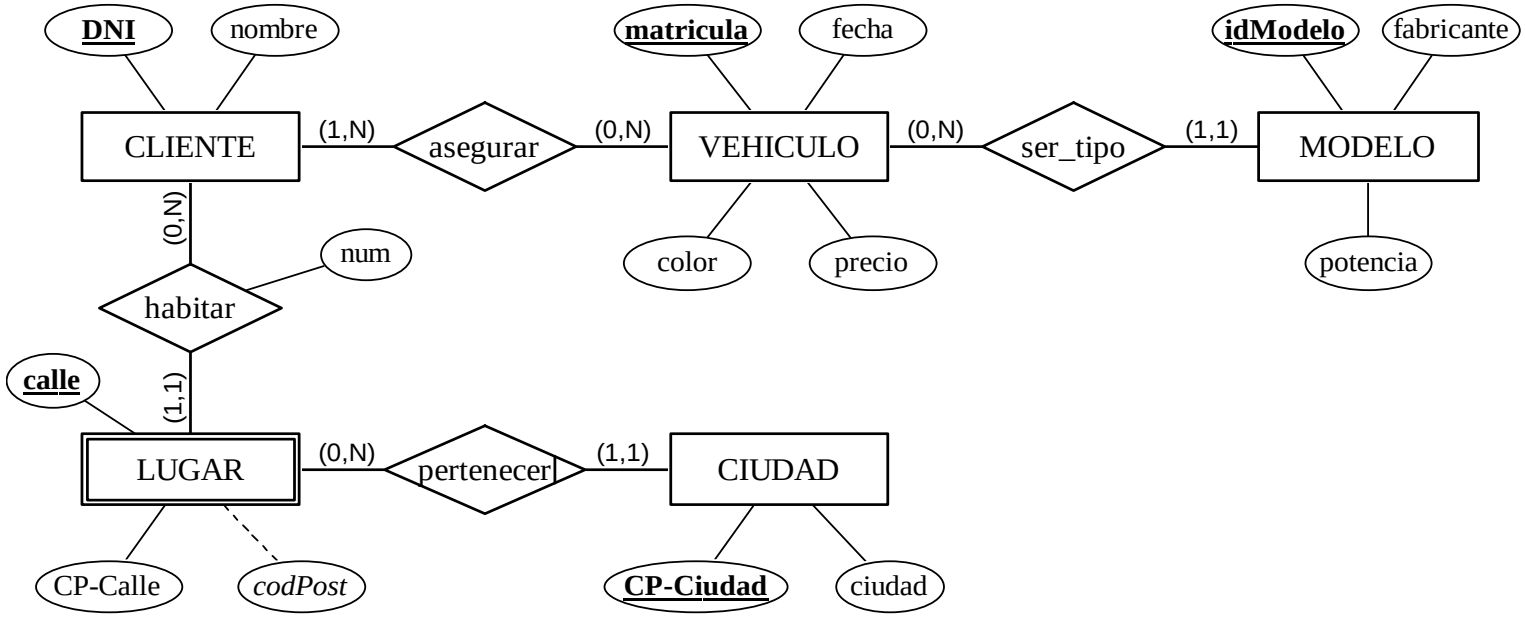
☛ *todas las dependencias funcionales no triviales se deducen del conocimiento de las claves de R*

en el ejemplo, **{ codPost } → { ciudad }** *pero codPost no es una clave*

☛ ***las descomposiciones necesarias para la 3FNBC pueden generar pérdidas de D.Func.***

ejemplo de aplicación de la FNBC

☛ ***considerando el Código Postal como la concatenación de (CP_Ciudad, CP_Calle) :***



suponiendo que toda la calle tiene el mismo código postal

7.3 - Dependencias multivaluadas y 4ª Forma Normal

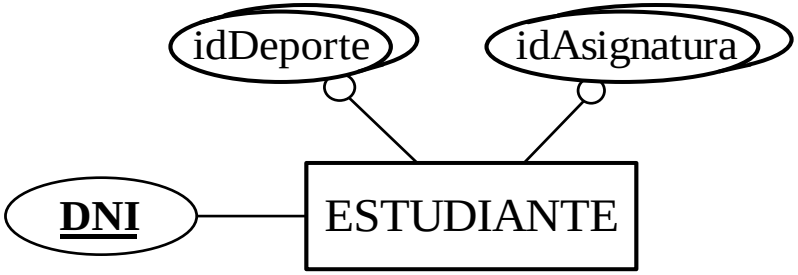
✓ ***la utilización de atributos multivaluados puede generar redundancias no deseadas***

definición de nuevas restricciones (4FN) ←

Diseñar una Base de Datos para guardar la información de las asignaturas en que se ha matriculado un alumno (DNI), así como los deportes que practica.

Soluc. 1:

`Estudiante = (DNI, idAsignatura, idDeporte);`



Estudiante

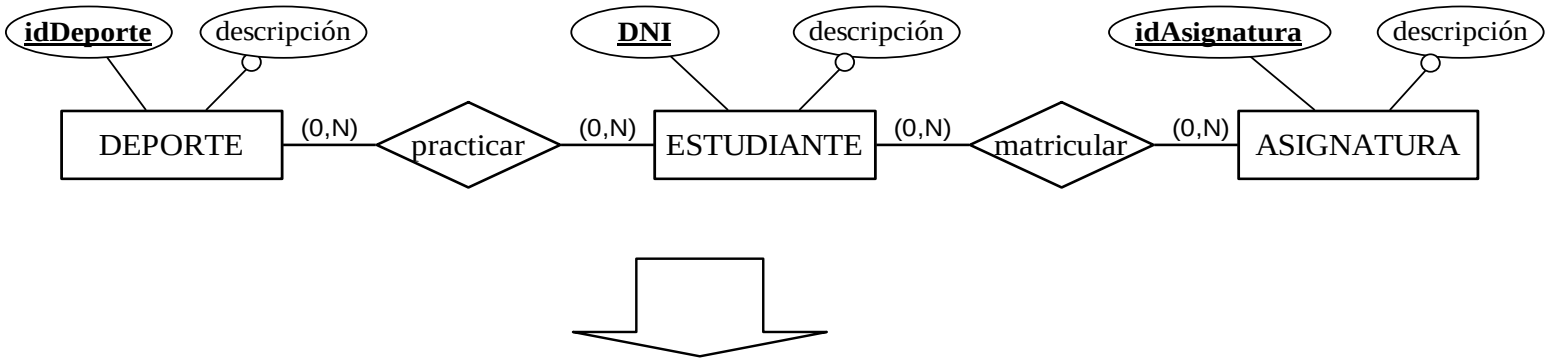
DNI	***idAsignatura***	***idDeporte***
15873564	Matemáticas	Tenis
15873564	Matemáticas	Golf
25654758	Lengua	Ajedrez
25654758	Matemáticas	Ajedrez
25654758	Quimica	Ajedrez

✓ ***pb. con valores nulos***
✓ ***pb. redundancias***

atributo tipo secuencia para la clave primaria

ejemplo de aplicación de la 4ª Forma Normal

Soluc. 2:



Estudiante = (DNI, descripcion);
Deporte = (idDeporte, descripcion);
Asignatura = (idAsignatura, descripcion);

practicar = (DNI, idDeporte);
matricular = (DNI, idAsignatura);

☛ ***se podría haber llegado a una solución similar a ésta aplicando la 4FN***

(pero obsérvese que con un buen diseño, no habría hecho falta)

Nota: Si no se hubieran añadido los atributos descripción, las relaciones Estudiante, Deporte y Asignatura, podrían no ser necesarias

dependencias multivaluadas y 4ª Forma Normal

Sean: $R = (A_1, A_2, \ldots, A_n)$ *un esquema de relación, y*
$X, Y \subseteq \{A_1, A_2, \ldots, A_n\}$ *dos subconjuntos de atributos de* R

Se dice que existe una ***dependencia multivaluada*** $\mathbf{X} \rightarrow\!\!\!\rightarrow \mathbf{Y}$, *o que* X ***multidetermina*** **Y**, *si, dados unos valores de* **X**, *los valores que toman los atributos* **Y** *son independientes de los que toman el resto de los atributos,* **R-X-Y**

☛ *las dependencias funcionales son un caso particular de dependencias multivaluadas*

Una relación **R** *está en* cuarta forma normal (4FN) si y sólo si :

Todas las dependencias multivaluadas son dependencias funcionales

Aplicación al diseño ⇒ *obtener una descomposición sin pérdida que verifique la 4FN*

4FN ⇨ ***3FNBC*** ⇨ ***3FN*** ⇨ ***2FN*** ⇨ ***1FN***

7.4 - Otras Formas Normales y ejemplos de diseño

En algunas (muy raras) ocasiones puede ser necesario verificar la quinta forma normal (5FN)

☛ ***si se ha prestado atención en el diseño E/R ⇒ no será necesario verificarla***

✓ ***se va a ilustrar el problema con el siguiente ejemplo*** (ver DATE)***:***

Diseñar una Base de Datos para guardar la información de las piezas que son utilizadas para fabricar los distintos equipos, junto con los vendedores de dichas piezas. Deberá considerarse que todas las piezas que suministra un vendedor podrán ser utilizadas en todos los proyectos en que se empleen y el vendedor suministre material (está acreditado)

a) si se propone como solución la relación: suministrar = (<u>vendedor</u>, <u>pieza</u>, <u>equipo</u>);

☛ ***aunque está en 4FN, presenta anomalías en la inserción y eliminación***

concepto de dependencia de JOIN

considérese la situación inicial de un sólo proveedor, Pérez, que suministra tuercas y tornillos para los dos únicos equipos fabricados, un motor y un generador:

suministrar

Vendedor	*Piezas*	*Equipo*
Pérez	tornillos	motor
Pérez	tuercas	generador

✓ ***para fabricar el generador se necesitan tornillos, que suministrará el señor López***

¡¡ se necesitan añadir 2 tuplas !!

suministrar

Vendedor	*Piezas*	*Equipo*
Pérez	tornillos	motor
Pérez	tuercas	generador
López	*tornillos*	*generador*
Pérez	***tornillos***	***generador***

es consecuencia de la restricción del enunciado:

Si Pérez suministra tornillos y
Si los tornillos se utilizan en el generador y
Si Pérez suministra piezas para el generador,
entonces Pérez suministra tornillos para el generador

dependencia de JOIN

dependencia de JOIN y 5ª Forma Normal

hay dependencia de JOIN (dependencia de proyección-combinación):

vender

Vendedor	*Piezas*
Pérez	tornillos
Pérez	tuercas
López	*tornillos*

⋈

componer

Piezas	*Equipo*
tornillos	motor
tuercas	generador
tornillos	*generador*

⋈

acreditar

Vendedor	*Equipo*
Pérez	motor
Pérez	generador
López	*generador*

=

=

suministrar

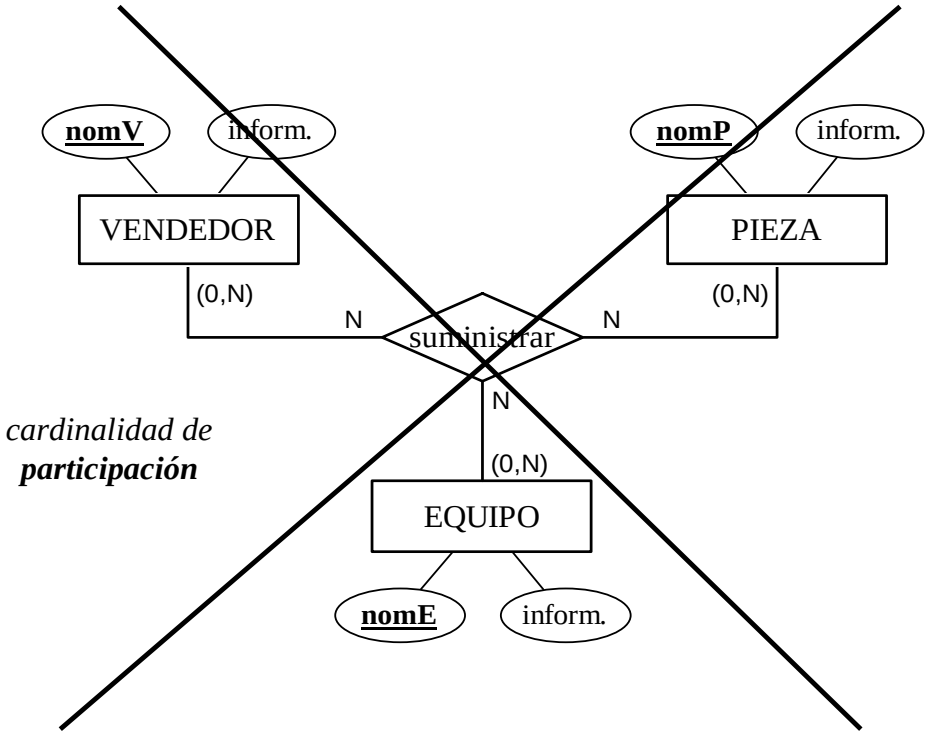
Vendedor	*Piezas*	*Equipo*
Pérez	tornillos	motor
Pérez	tuercas	generador
López	*tornillos*	*generador*
Pérez	***tornillos***	***generador***

Una relación* R *está en* 5FN *si toda dependencia de JOIN en* R *es una consecuencia de las claves candidatas

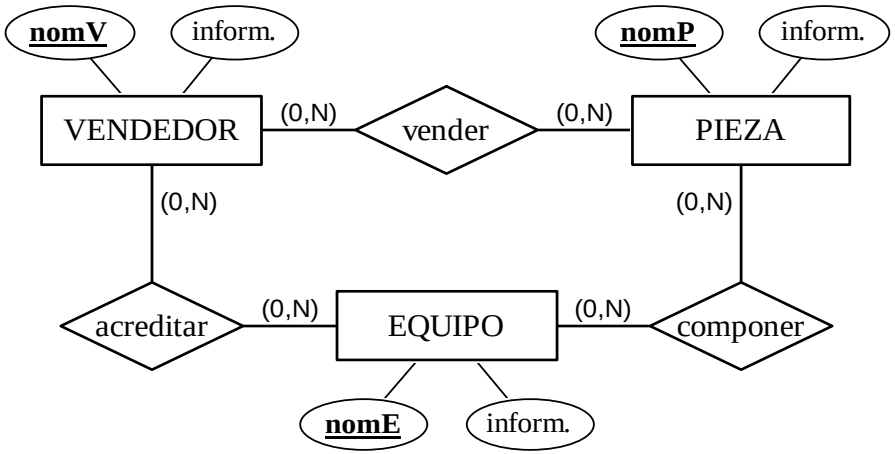
5FN ⇨ ***4FN*** ⇨ ***3FNBC*** ⇨ ***3FN*** ⇨ ***2FN*** ⇨ ***1FN***

5ª Forma Normal: consideraciones para el diseño E/R (1)

Obsérvese que con el enunciado anterior la solución correcta estaría basada en tres interrelaciones binarias, y no en una ternaria:



cardinalidad de ***participación***



5ª Forma Normal: consideraciones para el diseño E/R (2)

si el enunciado hubiera sido:

> *Diseñar una Base de Datos para guardar la información de las piezas que son utilizadas para fabricar los distintos equipos, junto con los vendedores de dichas piezas, de modo que se pueda saber quién ha suministrado cada pieza de cada equipo.*

En el ejemplo anterior, al añadir López como suministrador de tornillos para el generador,

Pérez suministra tornillos
se utilizan tornillos en el generador
Pérez suministra piezas para el generador $\not\Rightarrow$ *Pérez suministre tornillos para el generador*

en este caso NO hay dependencia de JOIN ⇒ no es válida la descomposición

☛ ***la solución correcta habría sido la interrelación ternaria (o equivalente)***

→ *aunque además haya relaciones binarias (significando otras cosas)*