

Bases de Datos

Clase 6: Diseño de Bases de Datos II

Hasta ahora

Sabemos cómo transformar requisitos de usuario en
modelo entidad relación (E/R), pero...
¿Qué hacemos con esto?

Hoy veremos cómo transformar el modelo E/R al
modelo relacional

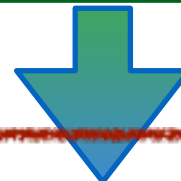
Diseño de base de datos

Análisis de requisitos

Usuarios



Requisitos

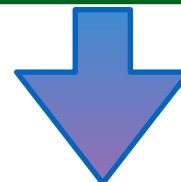


Diseño conceptual de bases de datos

Requisitos



Modelo entidad-relación



Diseño lógico de bases de datos

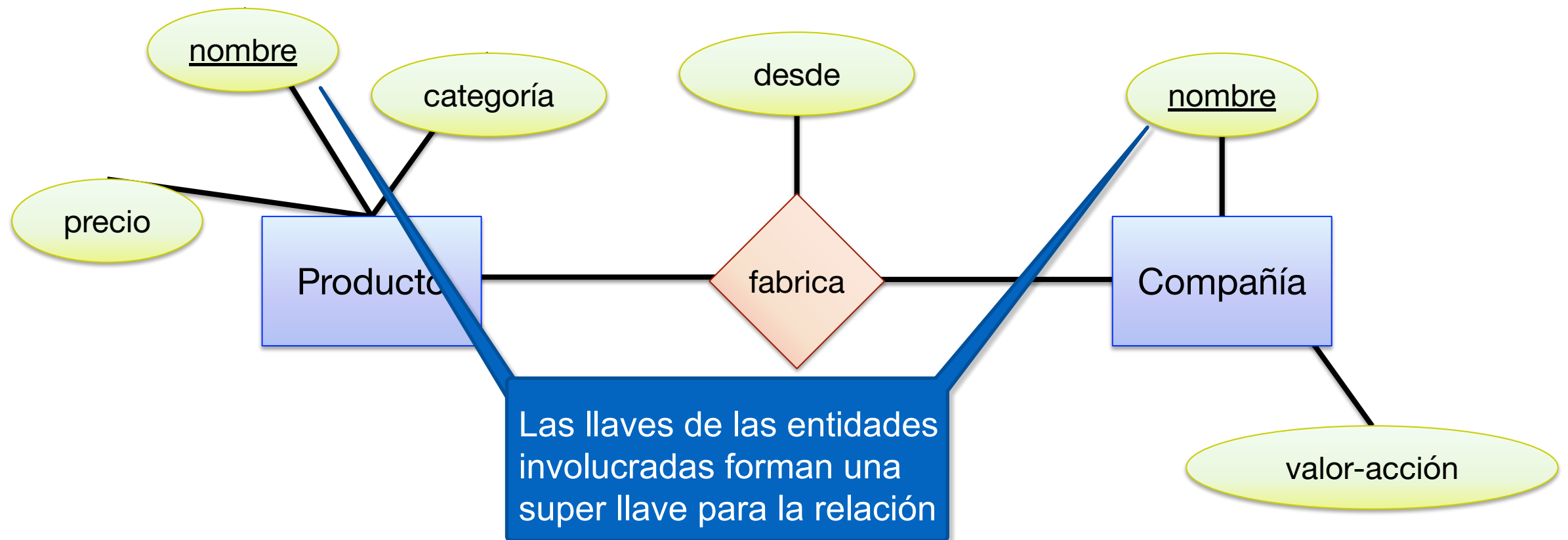
Modelo entidad-relación



Modelo relacional

Del Diagrama E/R al Modelo Relacional

M. E/R → M. Relacional

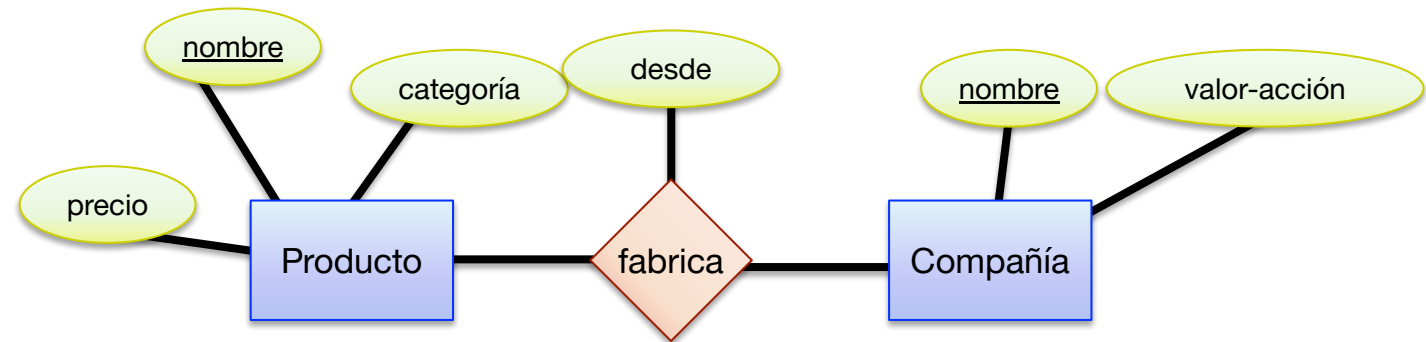


Producto(nombre: string, precio: int, categoría: string)

Compañía(nombre: string, valor-acción: int)

Fabrica(Producto.nombre: string, Compañía.nombre, desde: date)

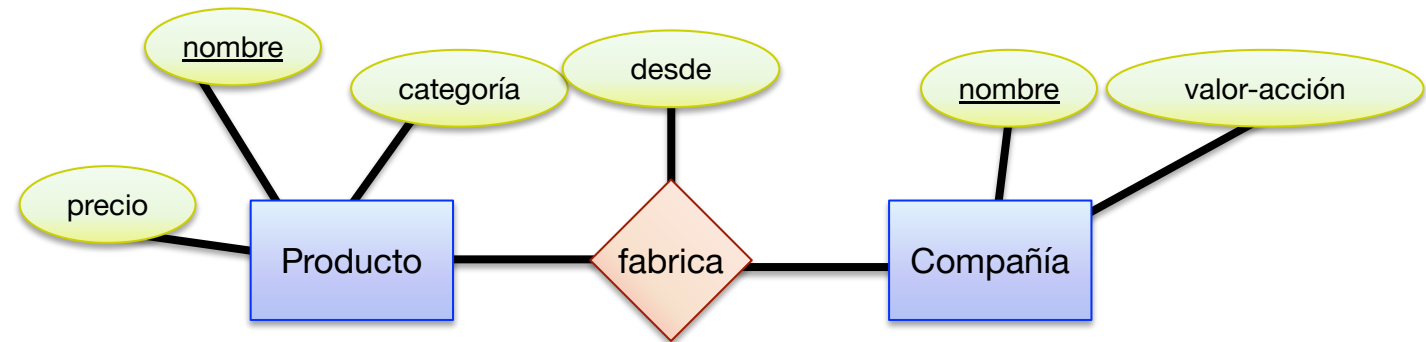
M. E/R → M. Relacional



Producto(nombre: string, precio: int, categoría: string)

```
CREATE TABLE producto(  
    nombre varchar(30),  
    precio int,  
    categoria varchar(30),  
    PRIMARY KEY (nombre)  
)
```

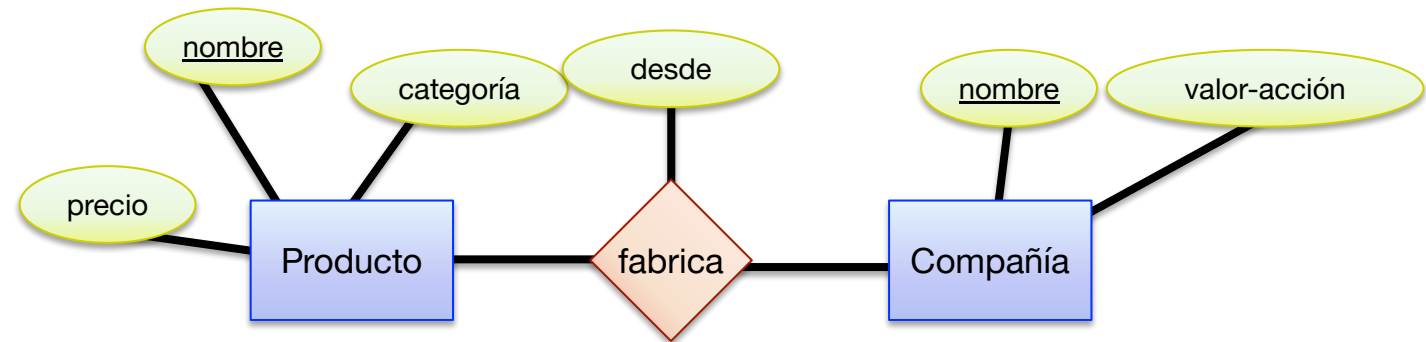
M. E/R → M. Relacional



Compañía(nombre: string, valor-acción: int)

```
CREATE TABLE compania(  
    nombre varchar(30),  
    valor_accion int,  
    PRIMARY KEY (nombre)  
)
```

M. E/R → M. Relacional



Fabrica(Producto.nombre: string, Compañía.nombre, desde: date)

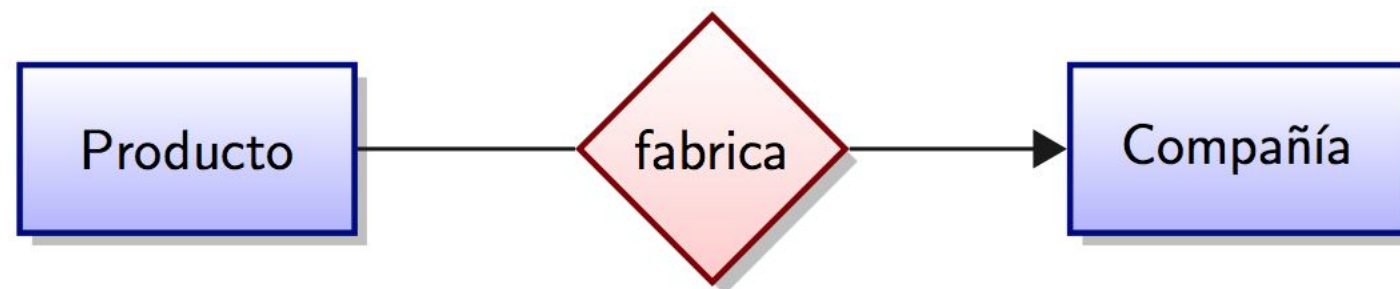
```
CREATE TABLE fabrica(  
    p_nombre varchar(30),  
    c_nombre varchar(30),  
    desde date,  
    PRIMARY KEY (p_nombre, c_nombre),  
    FOREIGN KEY(p_nombre) REFERENCES producto(nombre),  
    FOREIGN KEY(c_nombre) REFERENCES compania(nombre)  
)
```

Llaves foráneas

Paréntesis:
Llaves foráneas

Llaves foráneas

¿Qué pasa aquí?



pid	nombre	precio
1	SonyXZ89	\$100.000
2	MacBook103	\$10.000.000
3	Huawei23	\$1000

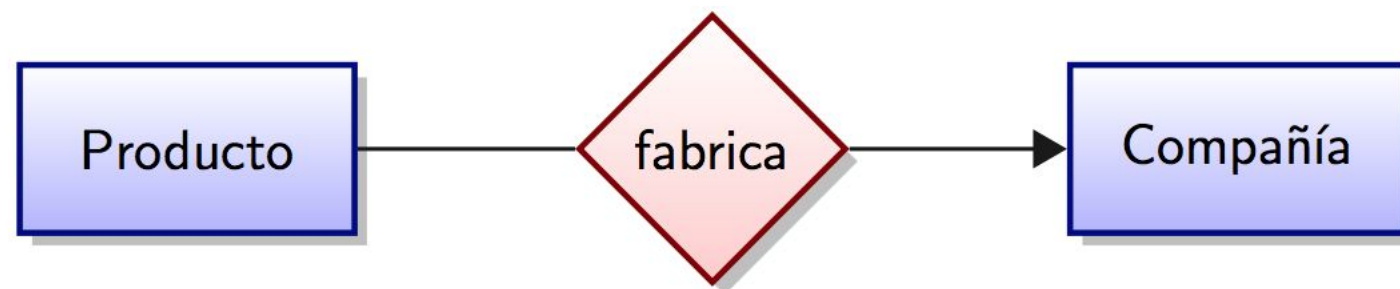
cid	nombre
C1	Sony
C2	Apple

fabrica

pid	cid
1	C1
2	C1
3	C8
89	C2

Llaves foráneas

¿Qué pasa aquí?



pid	nombre	precio
1	SonyXZ89	\$100.000
2	MacBook103	\$10.000.000
3	Huawei23	\$1000

cid	nombre
C1	Sony
C2	Apple

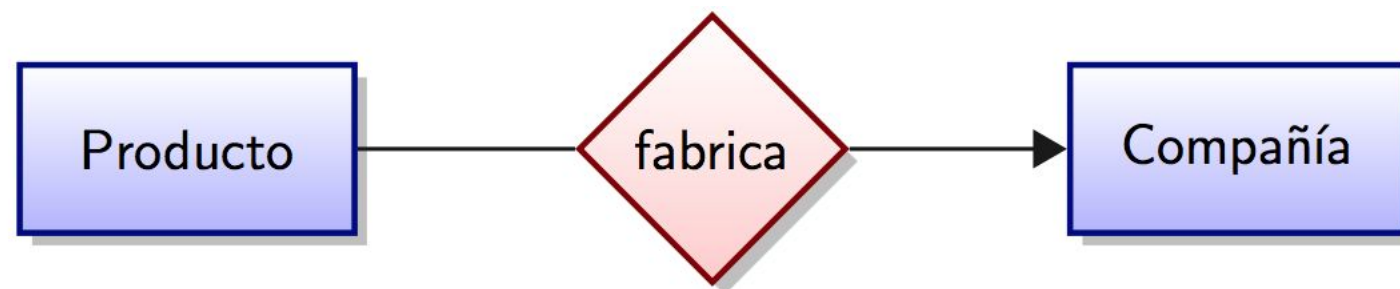
fabrica

pid	cid
1	C1
2	C1
3	C8
89	C2

???

Llaves foráneas

¿Qué pasa aquí?



pid	nombre	precio
1	SonyXZ89	\$100.000
2	MacBook103	\$10.000.000
3	Huawei23	\$1000

cid	nombre
C1	Sony
C2	Apple

fabrica

pid	cid
1	C1
2	C1
3	C8
89	C2

???

???

Llaves Foráneas

Cuando la referencia a la tabla es una llave:

$$R[A_1, \dots, A_n] \subseteq S[B_1, \dots, B_n]$$

Y B_1, \dots, B_n son llave para **S**

La relación R contiene la llave de la relación S

Llaves Foráneas en SQL

Inserciones con llaves foráneas

¿Qué pasa en este caso?

```
CREATE TABLE R(a int, b int, PRIMARY KEY(a));  
CREATE TABLE S(a int, c int, FOREIGN KEY(a) REFERENCES R, ...);  
INSERT INTO R VALUES(1, 1);  
INSERT INTO S VALUES(1, 2);
```

Todo bien hasta ahora...

Llaves Foráneas en SQL

Inserciones con llaves foráneas

¿Qué pasa en este caso?

```
CREATE TABLE R(a int, b int, PRIMARY KEY(a));  
CREATE TABLE S(a int, c int, FOREIGN KEY(a) REFERENCES R, ...);  
INSERT INTO R VALUES(1, 1);  
INSERT INTO S VALUES(1, 2);  
INSERT INTO S VALUES(2, 3);
```



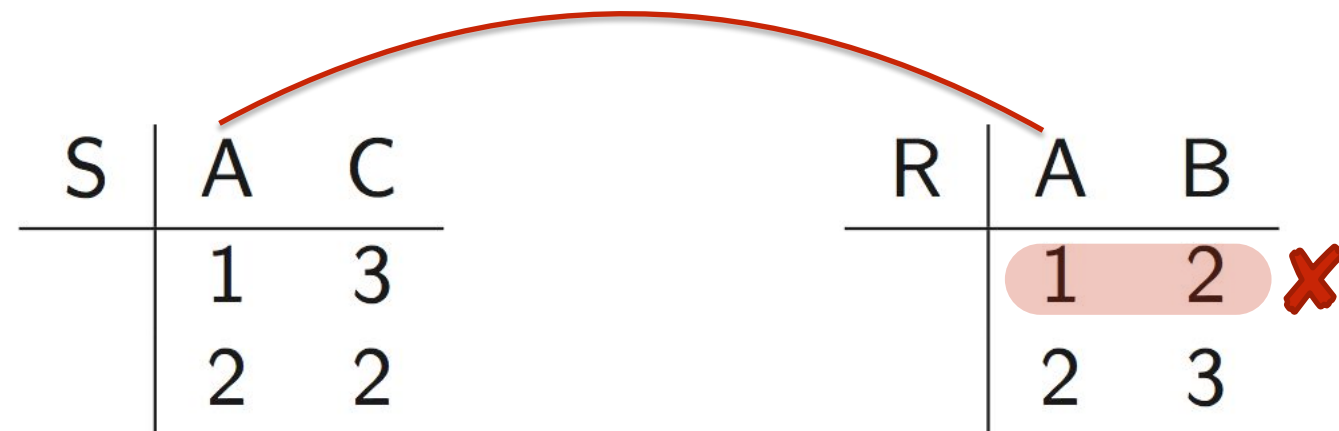
ERROR!

La base de datos no permite que se agreguen filas en que la llave foránea no está en la tabla referenciada!

Llaves Foráneas en SQL

Eliminar con llaves foráneas

Tenemos $\mathbf{S}[a] \subseteq \mathbf{R}[a]$ (llave foránea)



S	A	C
	1	3
	2	2

R	A	B
	1	2
	2	3

Qué ocurre al eliminar (1, 2) en \mathbf{R} ?

Llaves Foráneas en SQL

Eliminar con llaves foráneas

Qué ocurre al eliminar (1, 2) en **R**?

S	A	C
	1	3
	2	2

R	A	B
	1	2
	2	3

Tenemos las siguientes opciones:

- No permitir eliminación
- Propagar la eliminación y también borrar (1,3) de S
- Mantener la tupla en **S** pero dejar en la llave foránea el valor en null.

Llaves Foráneas en SQL

Eliminar con llaves foráneas

Qué ocurre al eliminar (1, 2) en **R**?



S	A	C
	1	3
	2	2

R	A	B
	1	2
	2	3

Opción 1: no permitir la eliminación. **Default en SQL!**

```
CREATE TABLE R(a int, b int, PRIMARY KEY(a))
```

```
CREATE TABLE S(a int, c int, FOREIGN KEY(a) REFERENCES R, ...)
```

Respuesta: obtenemos error

Llaves Foráneas en SQL

Eliminar con llaves foráneas

Qué ocurre al eliminar (1, 2) en **R**?



S	A	C
	1	3
	2	2

R	A	B
	1	2
	2	3

Opción 2: Propagar la eliminación. (**Cascada de eliminaciones**)

```
CREATE TABLE R(a int, b int, PRIMARY KEY(a))
```

```
CREATE TABLE S(a int, c int,
```

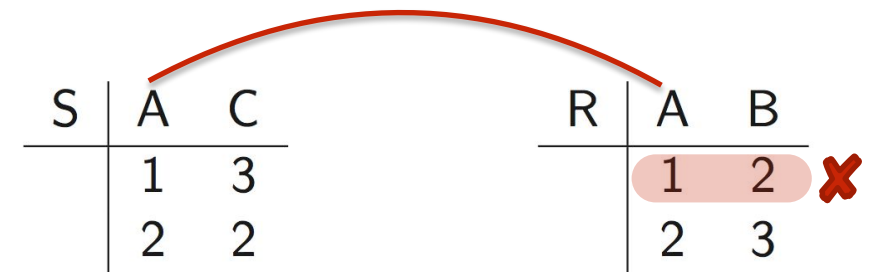
```
FOREIGN KEY(a) REFERENCES R ON DELETE CASCADE, ...)
```

Respuesta: se elimina también (1, 3) en S

Llaves Foráneas en SQL

Eliminar con llaves foráneas

Qué ocurre al eliminar (1, 2) en **R**?



S	A	C
	1	3
	2	2

R	A	B
	1	2
	2	3

Opción 3: **Dejar en nulo**

```
CREATE TABLE R(a int, b int, PRIMARY KEY(a))
```

```
CREATE TABLE S(a int, c int,
```

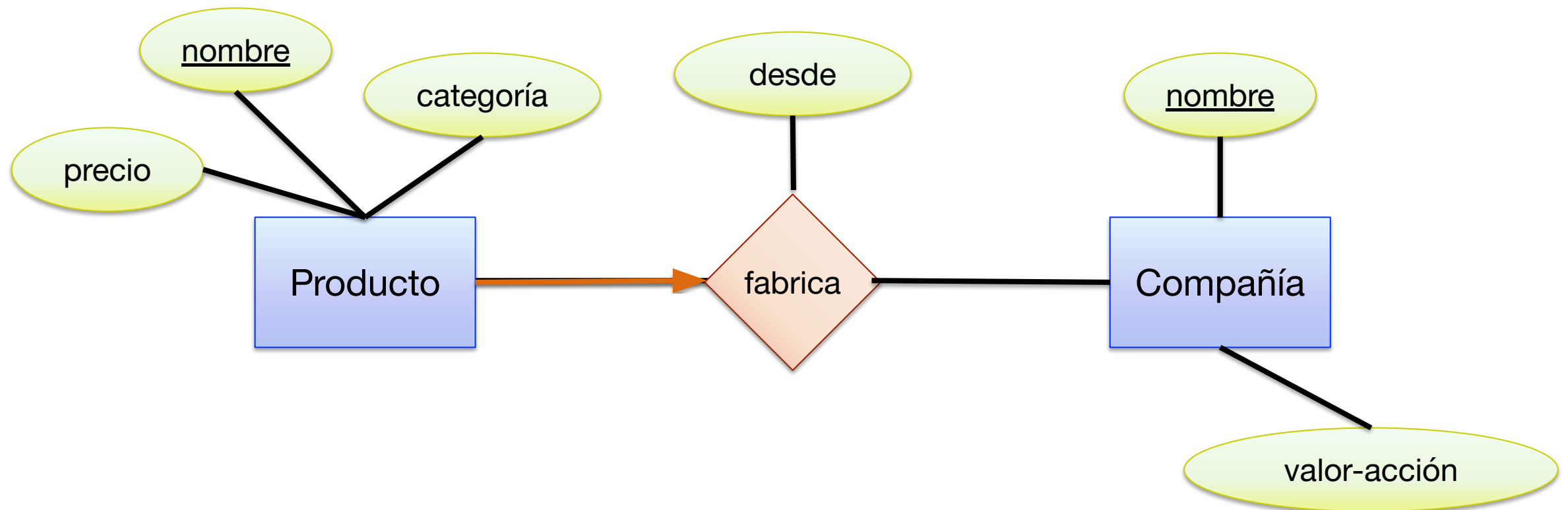
```
    FOREIGN KEY(a) REFERENCES R ON DELETE SET NULL, ...)
```

Respuesta: la tupla (1, 3) en S ahora es (null, 3)

¿Cómo representar E/R?

M. E/R → M. Relacional

¿Qué pasa aquí?



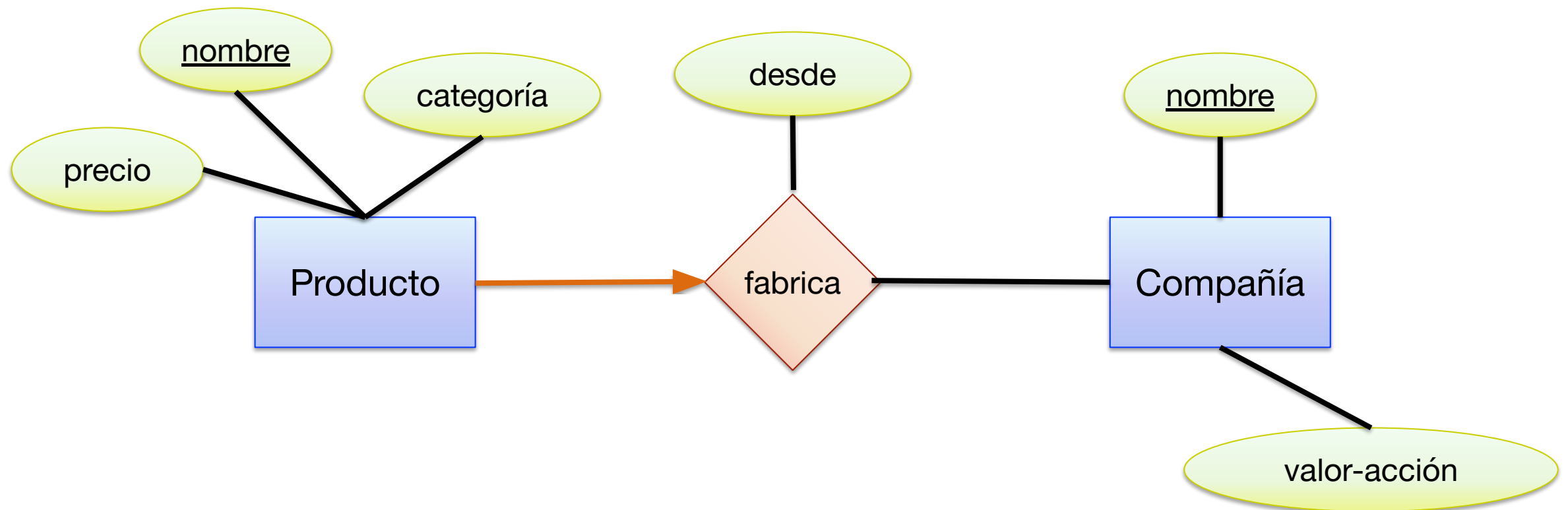
Producto(nombre: string, precio: int, categoría: string)

Compañía(nombre: string, valor-acción: int)

Fabrica(Producto.nombre: string, Compañía.nombre, desde: date)

M. E/R → M. Relacional

¿Qué pasa aquí?



Producto(nombre: string, precio: int, categoría: string)

Compañía(nombre: string, valor-acción: int)

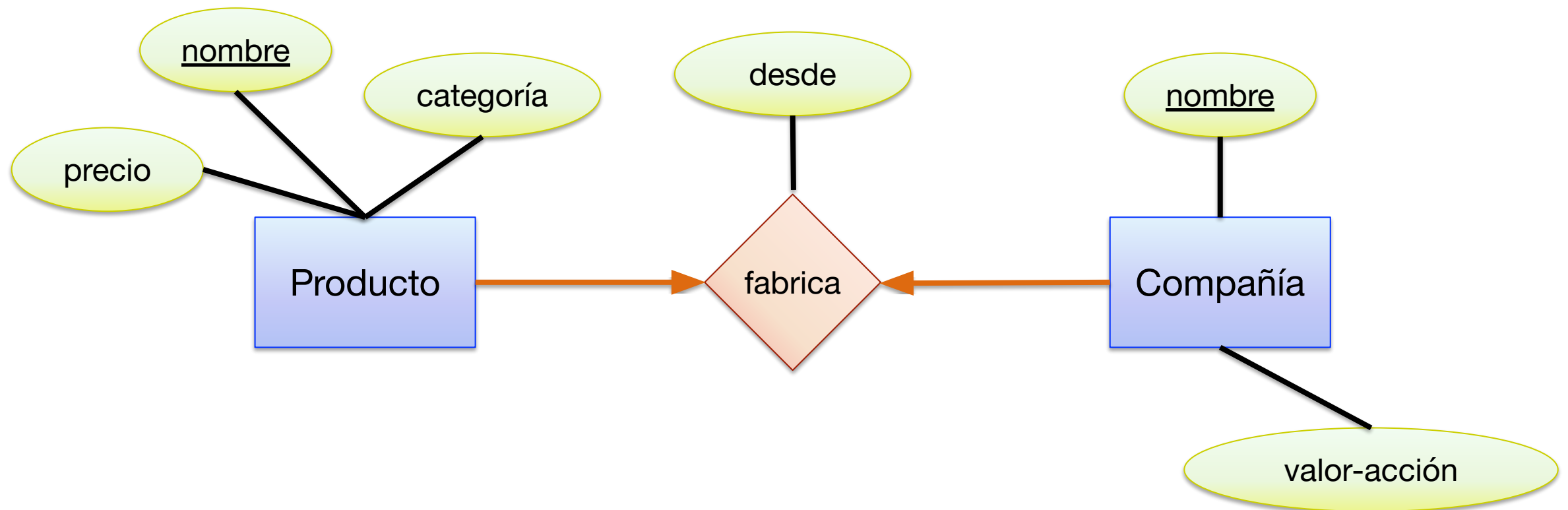
Fabrica(Producto.nombre: string, Compañía.nombre, desde: date)

Producto.nombre forma una llave
candidata

No se necesita que
Compañía.nombre sea llave

M. E/R → M. Relacional

¿Y ahora?



Producto(nombre: string, precio: int, categoría: string)

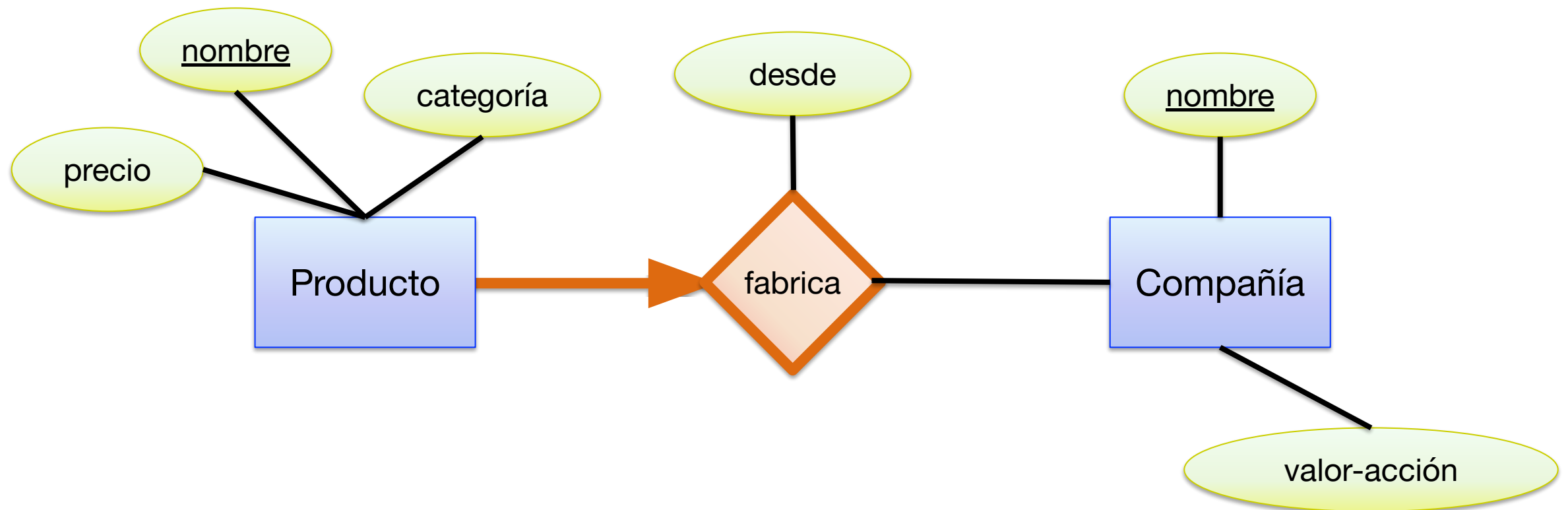
Compañía(nombre: string, valor-acción: int)

Fabrica(Producto.nombre: string, Compañía.nombre, desde: date)

Podemos hacer llave a Producto.nombre o a Compañía.nombre

M. E/R → M. Relacional

¿Y ahora?



Producto(nombre: string, precio: int, categoría: string,
Compañía.nombre: string, desde: date)

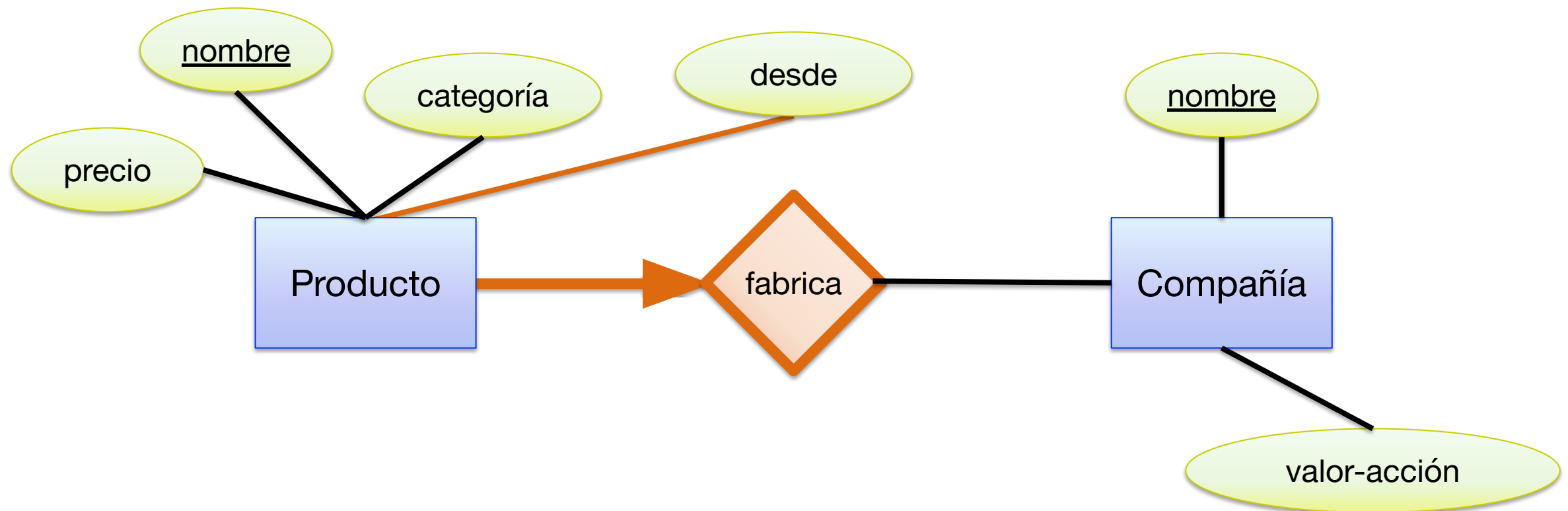
Compañía(nombre: string, valor-acción: int)

Sólo necesitamos una llave foránea en Producto.

Agregamos también el atributo de la relación.

M. E/R → M. Relacional

Un mejor diagrama



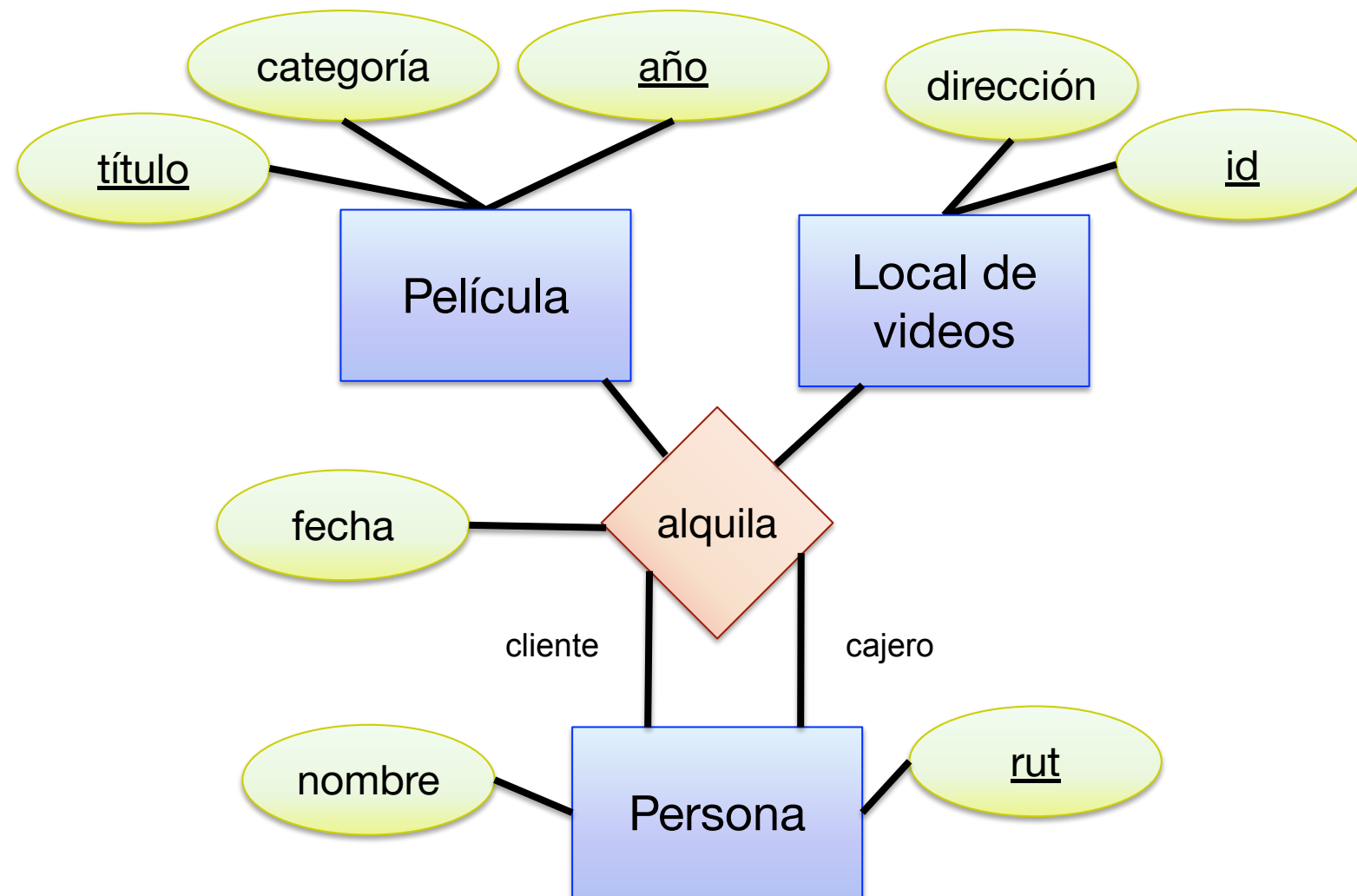
Producto(nombre: string, precio: int, categoría: string,
Compañía.nombre: string, desde: date)

Compañía(nombre: string, valor-acción: int)

Sólo necesitamos una llave foránea
en Producto.

Agregamos también el atributo de la
relación.

M. E/R → M. Relacional



Pelicula(titulo: string, año: int, categoría: string)

Local de videos(id: int, dirección: string)

Persona(rut: string, nombre: string)

Alquila(Pl.titulo: string, Pl.año: int, Pr.rut-cl: string, Pr.rut-ca: string, L.id: int, fecha: date)

M. E/R → M. Relacional

Jerarquía de clases

Opción 1: Tablas solo para las subclases

Vino(nombre: string, origen: string, año: string)

Cerveza(nombre: string, origen: string)

Opción 2: Tabla para la superclase

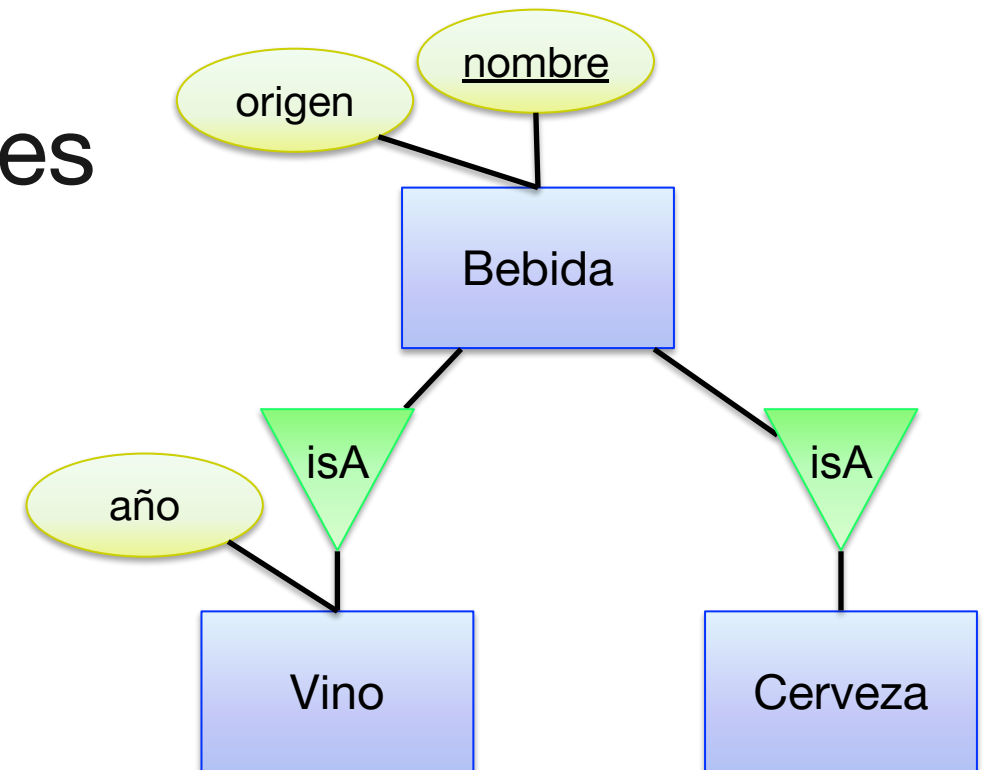
Bebida(nombre: string, origen: string)

Vino(nombre: string, año: string)

Cerveza(nombre: string)

¿Cuál es mejor?

Se requieren joins para acceder a todos los datos



Si hay mucho solapamiento: opción 2.
De lo contrario tendríamos mucha repetición de datos.

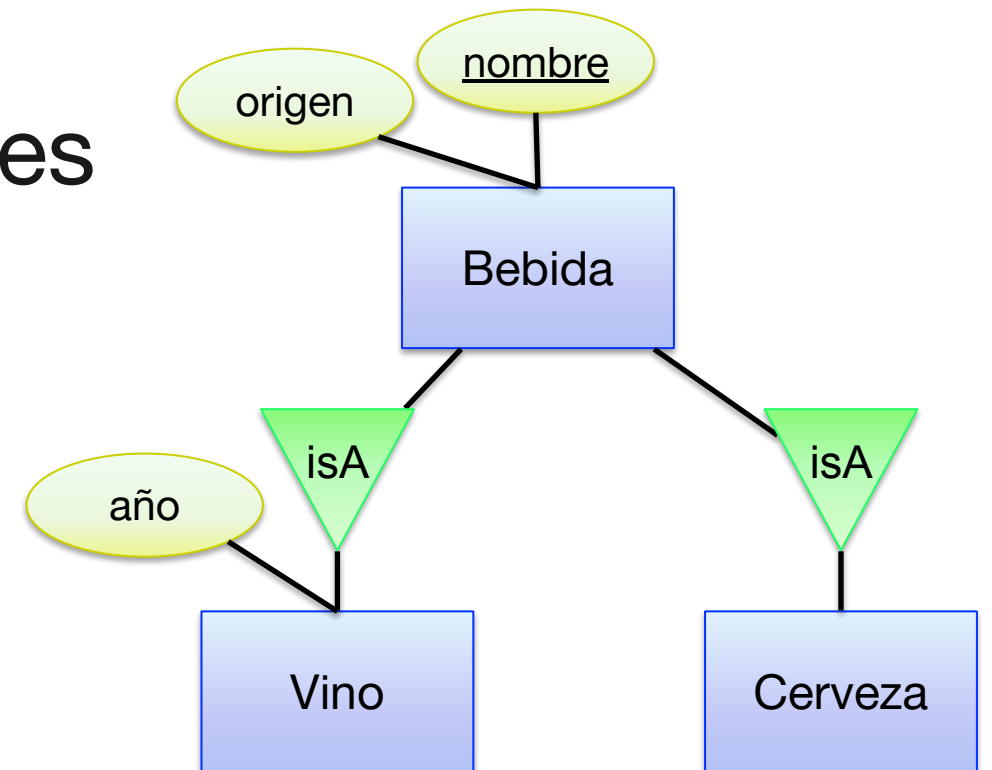
M. E/R → M. Relacional

Jerarquía de clases

Opción 1: Tablas solo para las subclases

Vino(nombre: string, origen: string, año: string)

Cerveza(nombre: string, origen: string)



Opción 2: Tabla para la superclase

Bebida(nombre: string, origen: string)

Vino(nombre: string, año: string)

Cerveza(nombre: string)

Se requieren joins para acceder a todos los datos

¿Cuál es mejor?

Si no hay cobertura: opción 2.
No hay otra opción o no podríamos guardar el whisky :(

M. E/R → M. Relacional

Jerarquía de clases

Opción 1: Tablas solo para las subclases

Vino(nombre: string, origen: string, año: string)

Cerveza(nombre: string, origen: string)

Opción 2: Tabla para la superclase

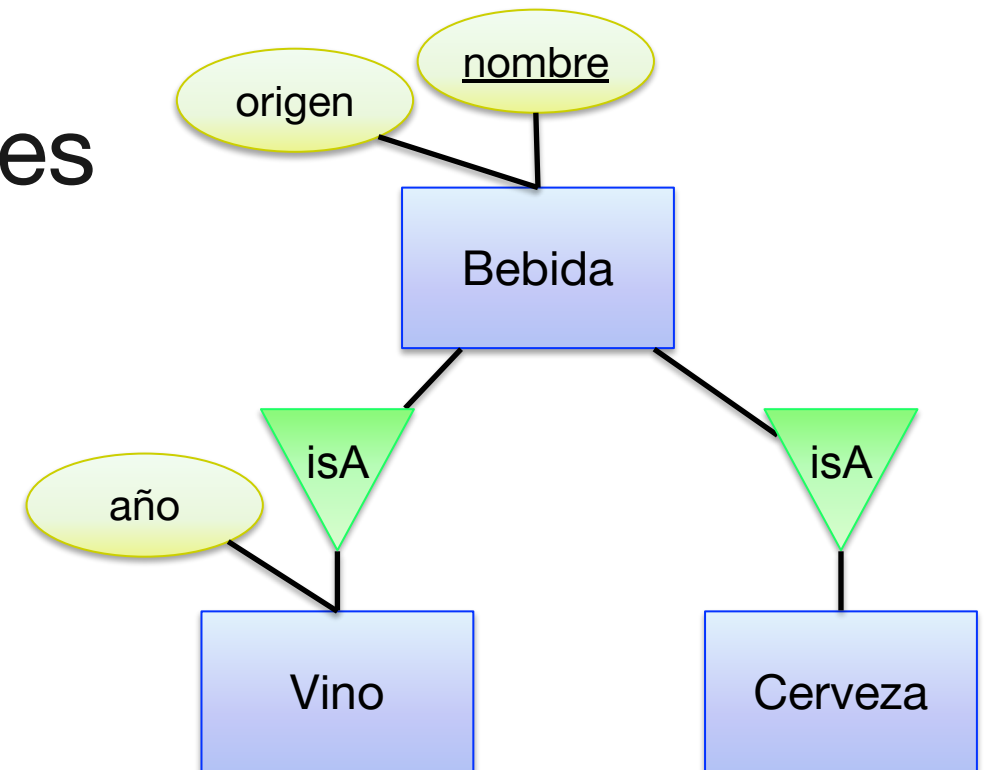
Bebida(nombre: string, origen: string)

Vino(nombre: string, año: string)

Cerveza(nombre: string)

¿Cuál es mejor?

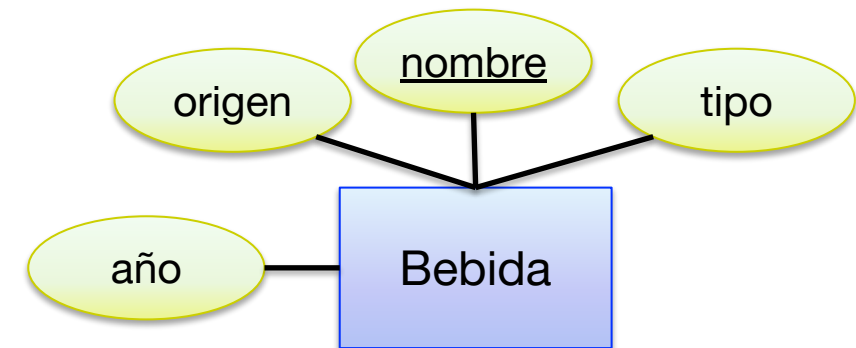
Se requieren joins para acceder a todos los datos



Si hay muchas consultas por **nombre**: opción 2.
Con la opción 1 tendríamos que consultar dos tablas.

M. E/R → M. Relacional

Jerarquía de clases



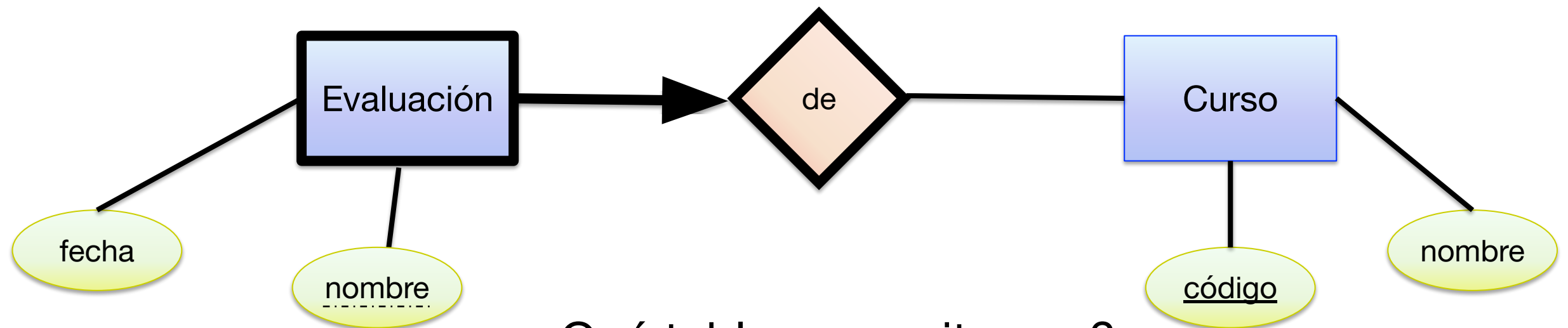
Opción 3: Quitar la jerarquía

Bebida(nombre: string, origen: string, año: string, tipo: string)

- Muchas repeticiones de la columna tipo.
- Puede que no se conozca el tipo (nulls).
- Pero más sencillo (y comprimible)

M. E/R → M. Relacional

Entidades débiles



¿Qué tablas necesitamos?

Curso(codigo: string, nombre: string)

Evaluación(nombre: string, C.código: string, fecha: date)

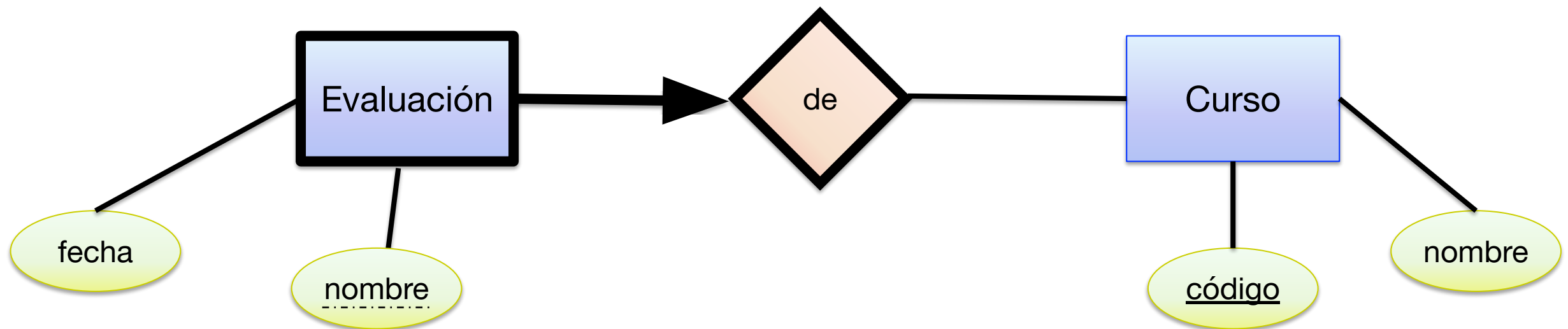
De(E.nombre: string, C.código: string) ❌

¿Está bien esto?

La tabla **De** es redundante (1-a-algo)
(y mal nombre para una tabla)

M. E/R → M. Relacional

Entidades débiles



Curso(codigo: string, nombre: string)

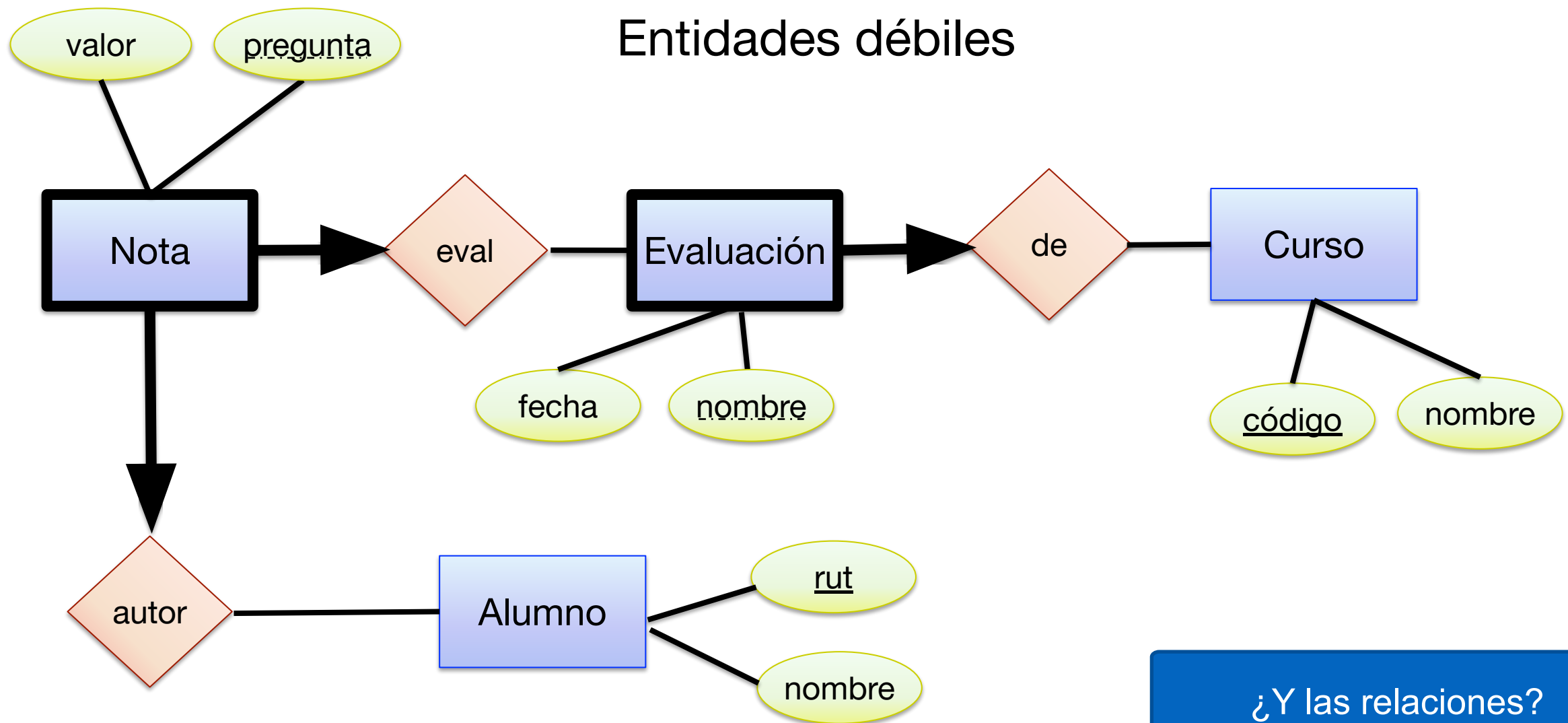
Evaluación(nombre: string, C.código: string, fecha: date)

¡Ahora sí!

```
CREATE TABLE evaluacion(  
    nombre varchar(30) NOT NULL,  
    codigo varchar(30) NOT NULL,  
    fecha date,  
    PRIMARY KEY (nombre, codigo)  
    FOREIGN KEY(codigo) REFERENCES curso(codigo) ON DELETE CASCADE  
)
```

M. E/R → M. Relacional

Entidades débiles



¿Y las relaciones?

Curso(codigo: string, nombre: string)

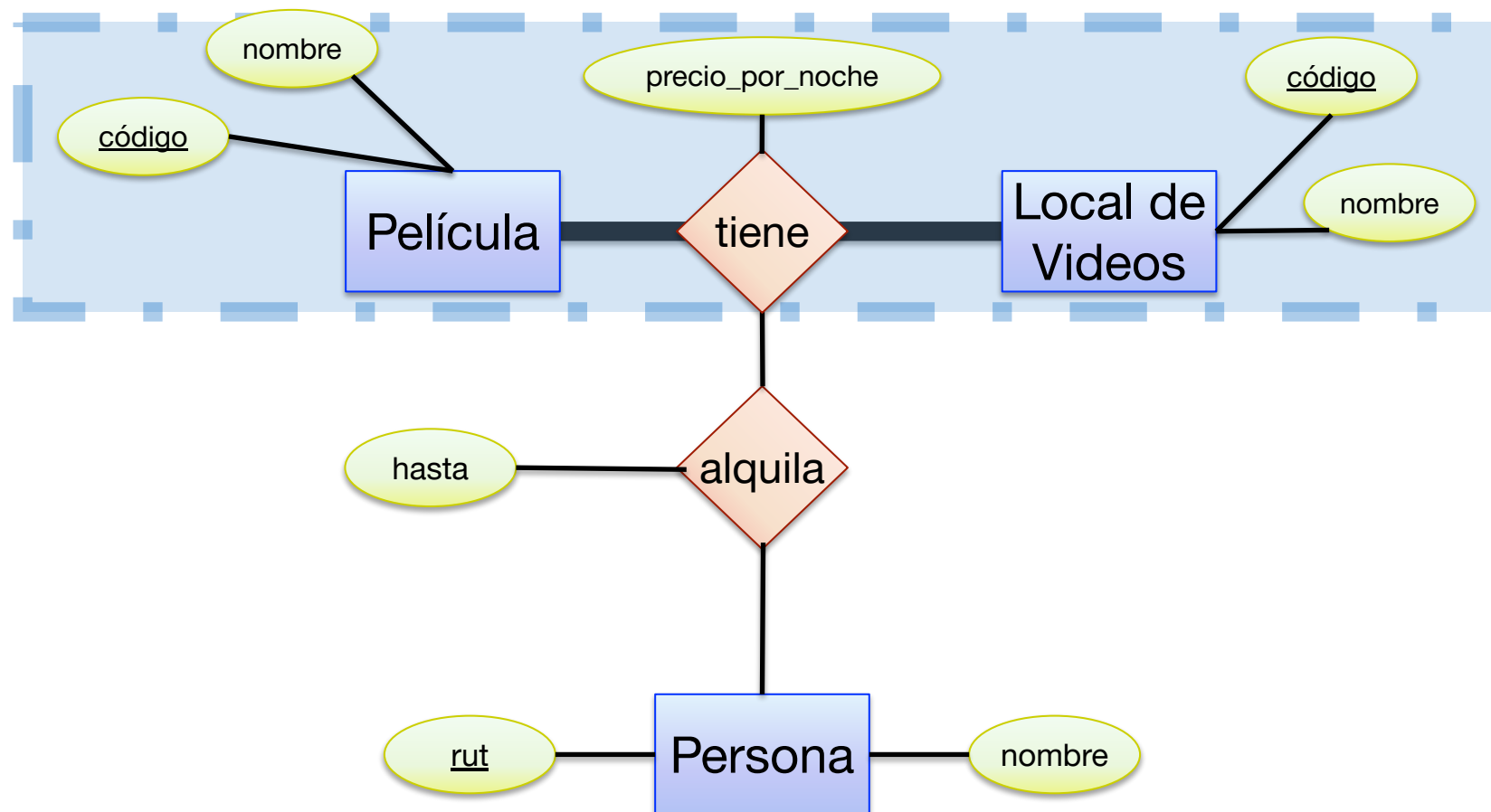
Evaluación(nombre: string, C.código: string, fecha: date)

Nota(pregunta: int, E.nombre: string, C.código: string, A.rut: string, valor: float)

Alumno(rut: string, nombre: string)

M. E/R → M. Relacional

Agregación



Película(código: string, nombre: string)

LocalDeVideos(código: string, nombre: string)

Tiene(P.código: string, L.código: string, precio_por_noche: int)

Persona(rut: string, nombre: string)

Alquila(T.p código: string, T.l código: string, Pr.rut, hasta: date)

M. E/R → M. Relacional

Un mejor diagrama

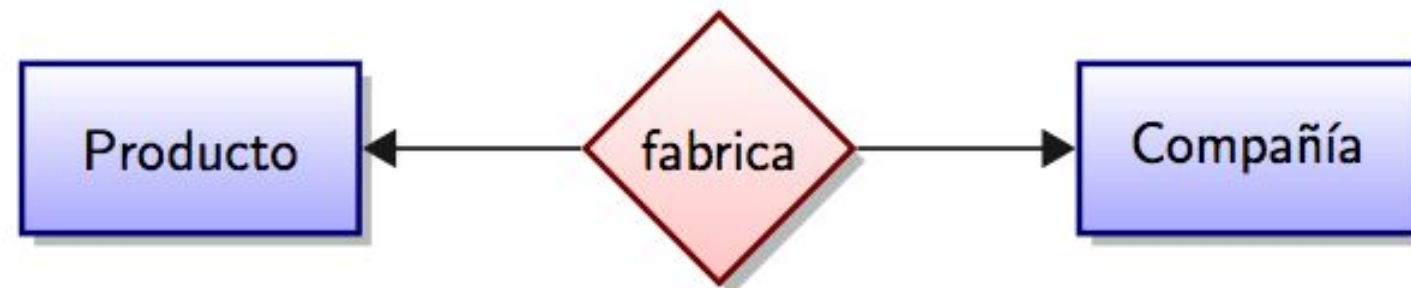
`Producto`(`nombre`: string, `precio`: int, `categoría`: string,
`Compañía.nombre`: string, `desde`: date)

Agregamos además una restricción de integridad de unicidad a la llave, para que la relación se mantenga 1:1. Hablaremos más de restricciones de integridad en un rato.

```
CREATE TABLE producto(  
    nombre varchar(30),  
    precio int,  
    categoria varchar(30),  
    c_nombre varchar(30),  
    desde date,  
    PRIMARY KEY (nombre),  
    UNIQUE(c_nombre),  
    FOREIGN KEY(c_nombre) REFERENCES compania(nombre)  
)
```

Llaves Foráneas

Relación 1:1



¿Cómo modelamos con llaves foráneas?

Sólo necesitamos una llave foránea en cualquiera de las dos tablas (Por qué??).

Agregamos además una restricción de integridad de unicidad a la llave, para que la relación se mantenga 1:1. Hablaremos más de restricciones de integridad en un rato. En SQL:

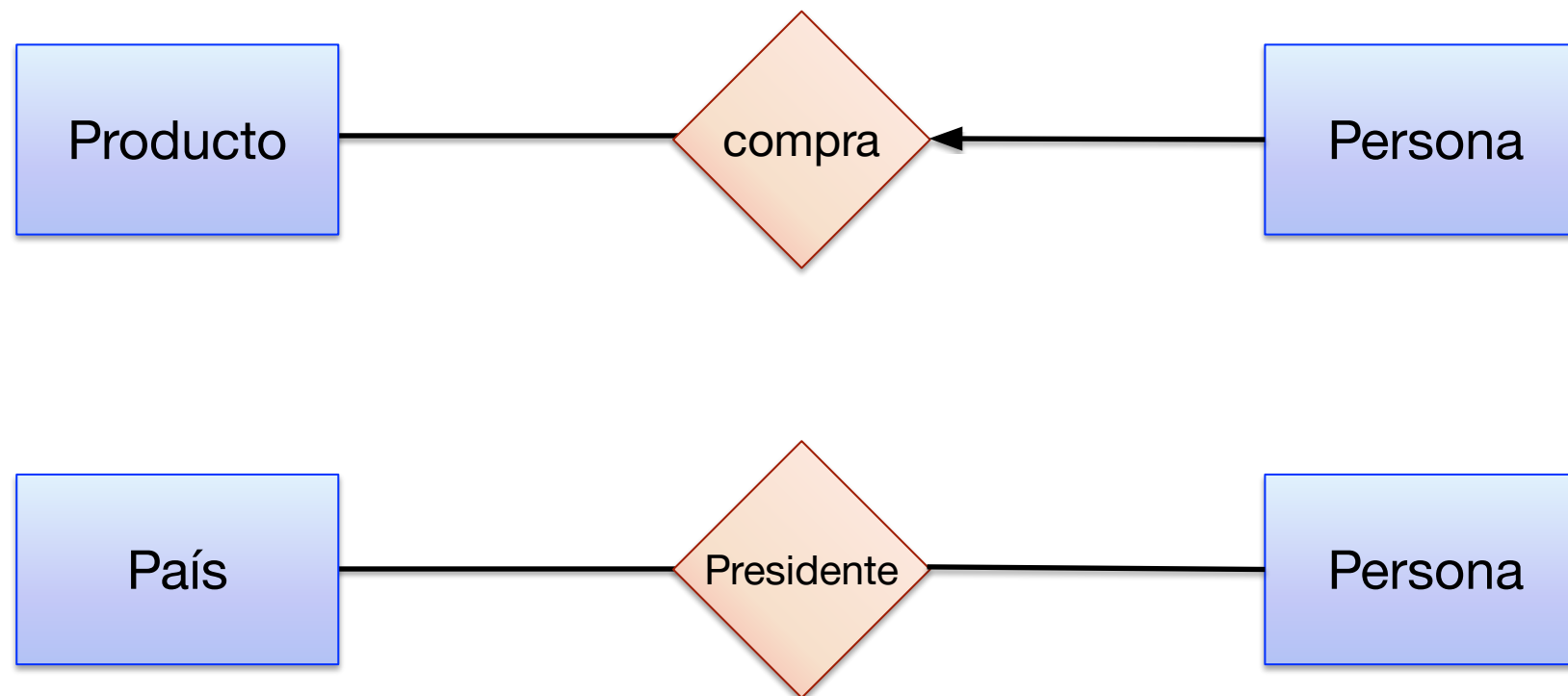
```
CREATE TABLE Producto(  
    id int,  
    nombre varchar(30),  
    categoria varchar(30),  
    precio int,  
    id_compañía int UNIQUE,  
    PRIMARY KEY (id),  
    FOREIGN KEY(id_compañía) REFERENCES Compañía(id)  
)
```

Principios básicos del diseño

Principios básicos del diseño

Fidelidad al problema

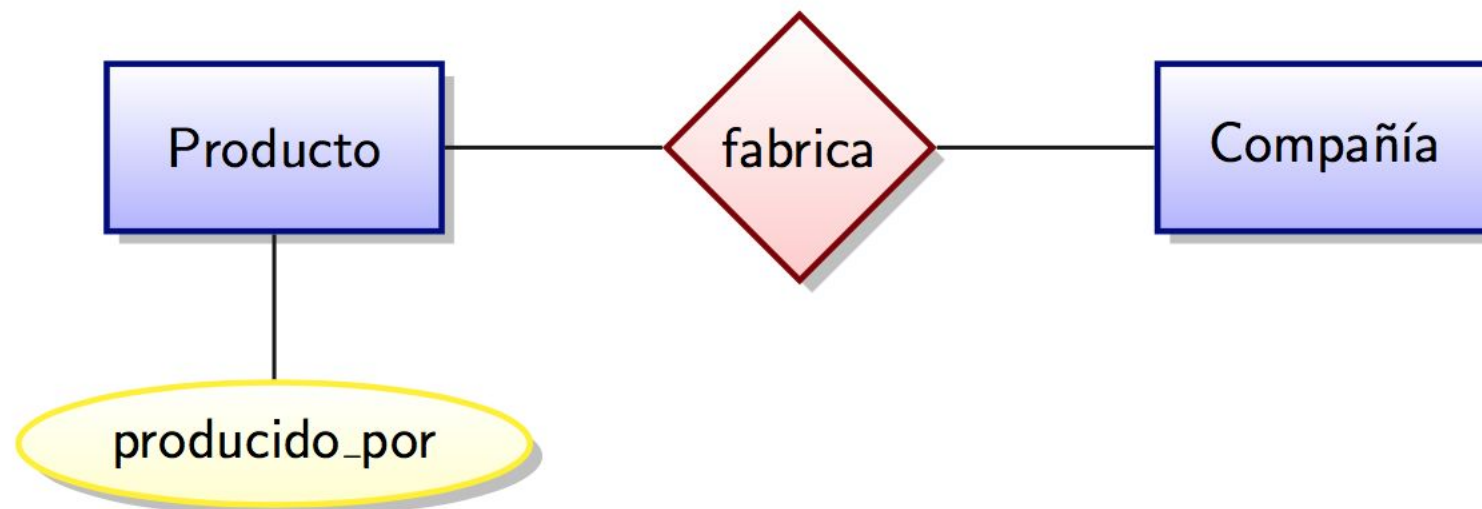
¿Qué está mal?



Principios básicos del diseño

Evitar redundancia

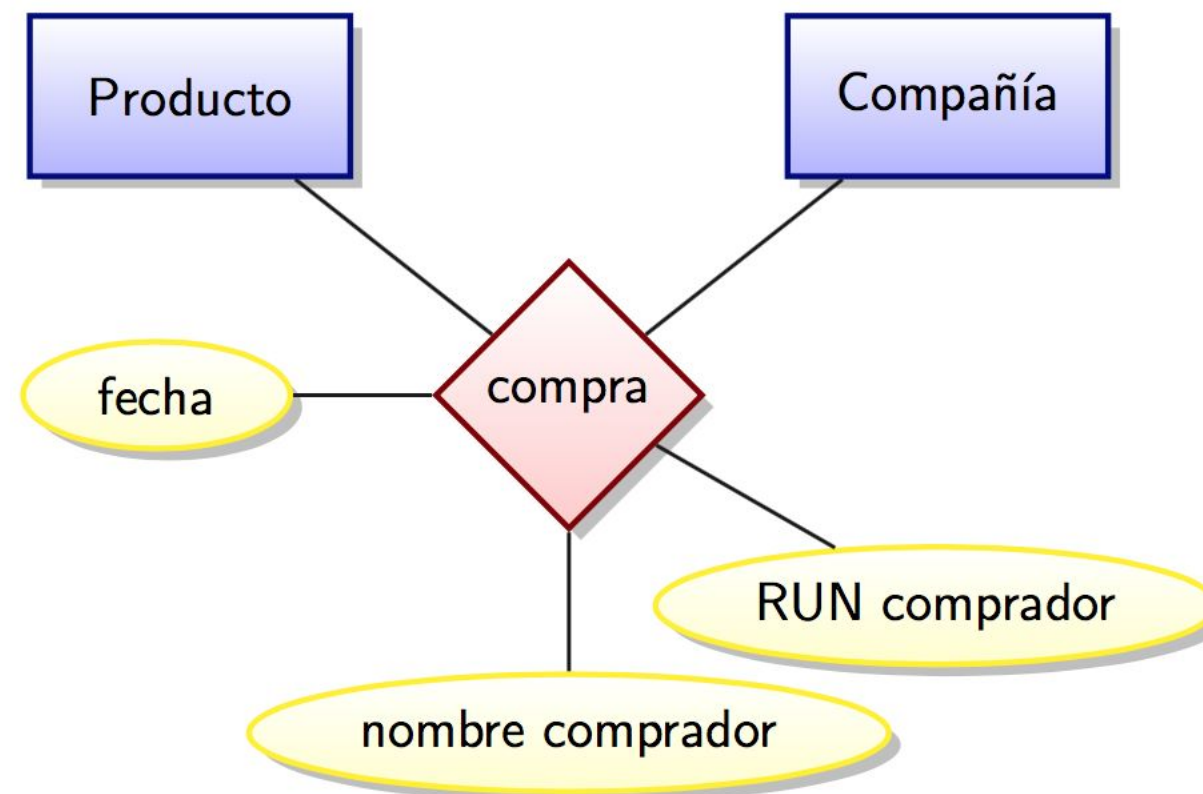
Algo como esto, puede generar **anomalías**



Principios básicos del diseño

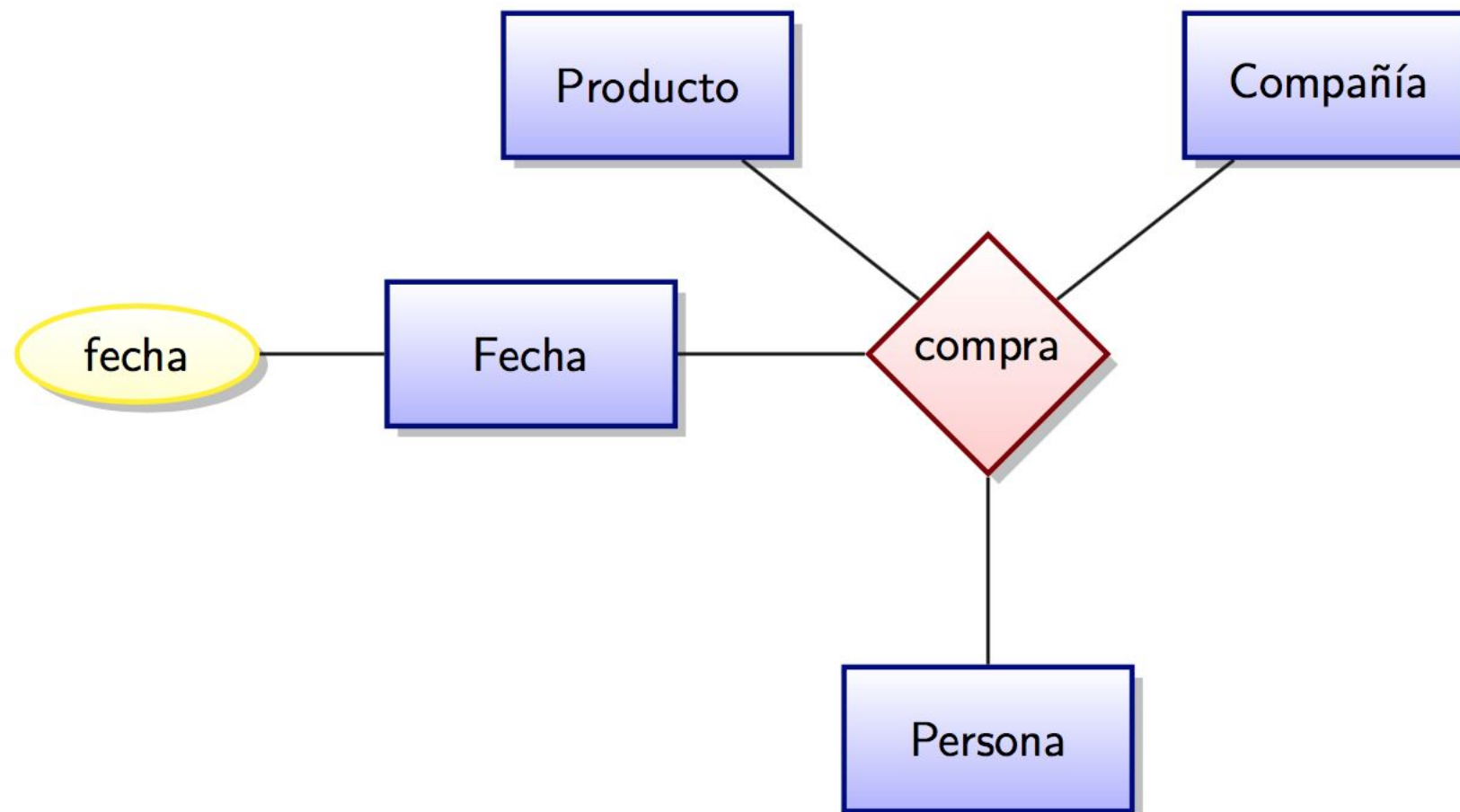
Elegir entidades y relaciones correctamente

¿Qué está mal?



Principios básicos del diseño

No complicar más de lo necesario



Principios básicos del diseño

Elección de llave primaria.

Al momento de diseñar siempre queremos identificar todas los atributos de las entidades que son candidatos a ser llave de la tabla, a estos les llamamos *natural key*, porque son columnas que naturalmente tienen el comportamiento de una llave. Por ejemplo de la siguiente tabla:

Usuario(email, rut, username, nombre, tipo, fecha_de_inscripcion)

rut, email y username son posibles *natural keys*.

...pero en la práctica el 99% de las veces es mejor usar una columna inventada, sin significado que sea autogenerada por el RDBMS. A esto le llamamos [surrogate key](#).

Principios básicos del diseño

Elección de llave primaria.

Bueno en realidad es algo medio opinionado...

Surrogate vs. natural/business keys [closed]

Asked 12 years, 7 months ago Active 2 months ago Viewed 71k times



Closed. This question is [opinion-based](#). It is not currently accepting answers.

Principios básicos del diseño

Elección de llave primaria.

Bueno en realidad es algo medio opinionado...

Pero en la práctica los frameworks de desarrollo web modernos esperan una *surrogate key* llamada *id* como llave primaria e incluso la generan por defecto.

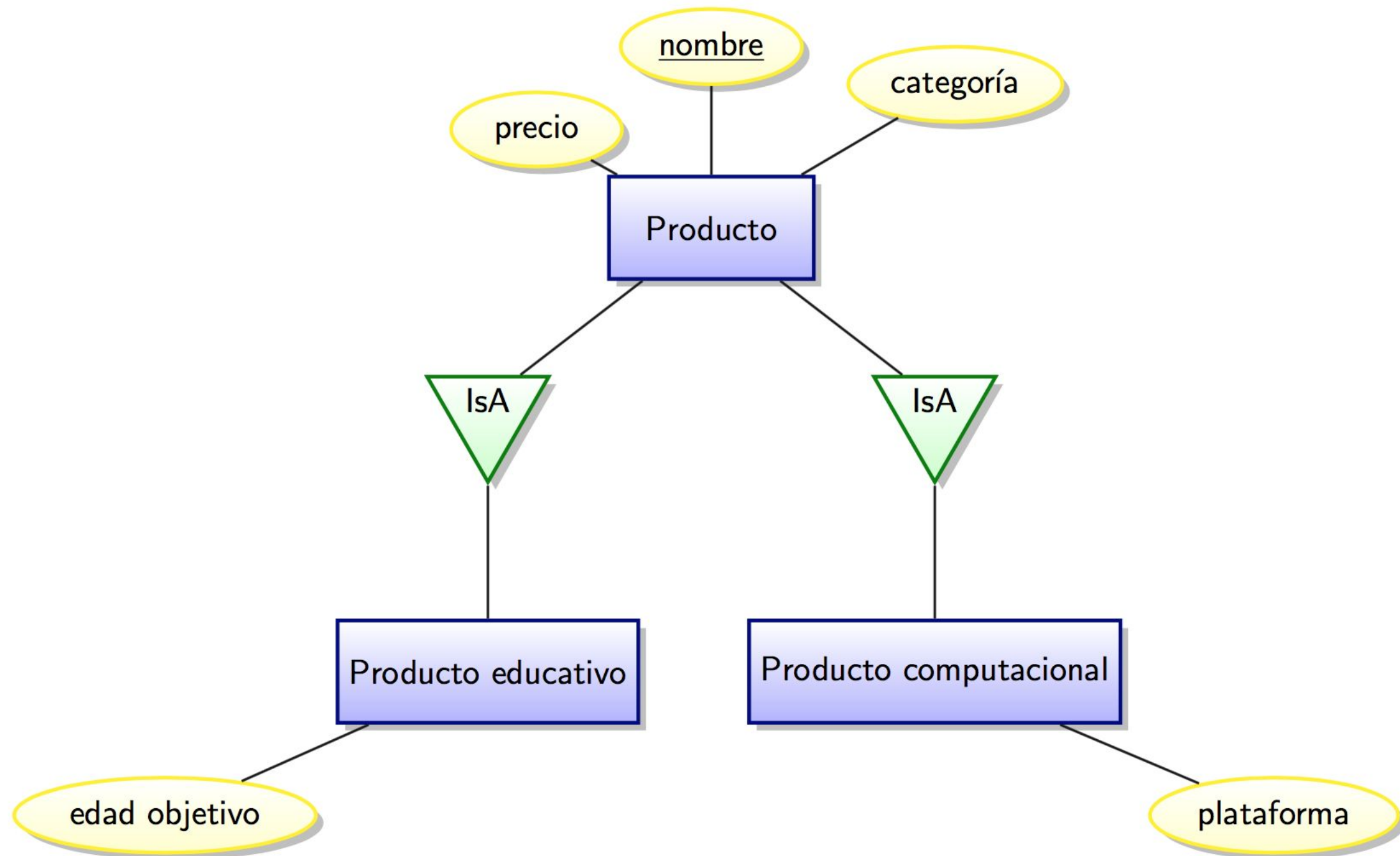
La tabla anterior deberíamos generarla así:

```
CREATE TABLE usuario(  
    id SERIAL,  
    email nombre varchar(30) UNIQUE NOT NULL,  
    RUT varchar(30) UNIQUE NOT NULL,  
    fecha date,  
    PRIMARY KEY (id)  
)
```

Subclases (IsA)

- Algunas entidades son casos especiales de otra
- Similar a la herencia en orientación a objetos
 - Ej. Todo estudiante es también una persona
- En E/R usamos **IsA** (EsUn en Inglés)

Subclases (IsA)



Subclases (IsA)

Sin herencia

Mantener sólo las entidades hijo con todos los atributos de la entidad padre:

ProductoEducativo(id, nombre, edad_objetivo, precio, categoría)

ProductoComputacional(id, nombre, plataforma, precio, categoría)

Ventajas: No se necesitan joins para acceder todas las columnas

Desventaja: No puedo tener productos genéricos ni solapados (en la práctica si podríamos querer un producto educativo y computacional a la vez)

Subclases (IsA)

Herencia con tablas múltiples.

- Se hacen tablas para todas las entidades y las entidades hijo tienen una referencia a la entidad padre.
- Las entidades hijo tienen sólo los atributos adicionales
- Para obtener la versión completa hago un join

Ventaja: puedo tener productos genéricos y solapados.

Desventaja: se requieren joins para acceder a todos los datos.

Subclases (IsA)

Herencia con tablas múltiples.

Producto(id, nombre, precio, categoría)

ProductoEducativo(id, nombre, edad_objetivo, id_producto)

ProductoComputacional(id, nombre, plataforma, id_producto)

Subclases (IsA)

Herencia con una sola tabla.

- Se hace una sola tabla que tiene las columnas tanto del padre como de los hijos.
- Se puede agregar una columna tipo para guardar a qué entidad pertenece cada fila.

Ventaja: No se requieren joins y se pueden tener productos solapados.

Desventaja: Pueden quedar muchos nulos en las tablas y estos no se pueden evitar, aunque la entidad hija a la que corresponda el atributo nunca lo tenga vacío.

Subclases (IsA)

Herencia con una sola tabla.

Producto(nombre, precio, categoría, edad_objetivo, plataforma, tipo)

Restricciones de integridad

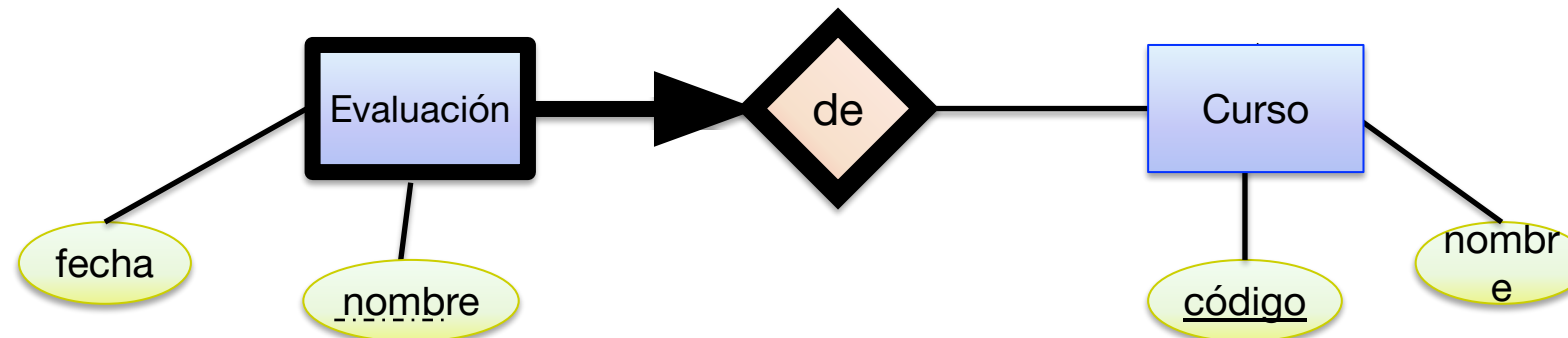
Restricciones de integridad

Son **restricciones** formales que imponemos a **un esquema** que todas sus instancias deben satisfacer. Algunas son:

- **De valores nulos:** El valor puede o no ser nulo.
- **Unicidad:** Dado un atributo, no pueden haber dos tuplas con el mismo valor.
- **De llave:** el valor es único y no puede ser null.
- **De referencia:** si se trabaja en una compañía, esta debe existir (Llaves foráneas).
- **De dominio:** la edad de las personas debe estar entre 0 y 150 años.

M. E/R → M. Relacional

Entidades débiles



Curso(codigo: string, nombre: string)

Evaluación(nombre: string, C.código: string, fecha: date)

No puede ser nulo

```
CREATE TABLE evaluacion(
```

```
  nombre varchar(30) NOT NULL
```

Tiene que tener a lo más 30 caracteres

```
  codigo varchar(30) NOT NULL,
```

No puede ser nulo

```
  fecha date DEFAULT NOW(),
```

Tiene que ser una fecha, y su valor por defecto es la fecha actual

```
  PRIMARY KEY (nombre, codigo)
```

La llave son "nombre" y "código"

```
  FOREIGN KEY(codigo) REFERENCES curso(codigo) ON DELETE CASCADE
```

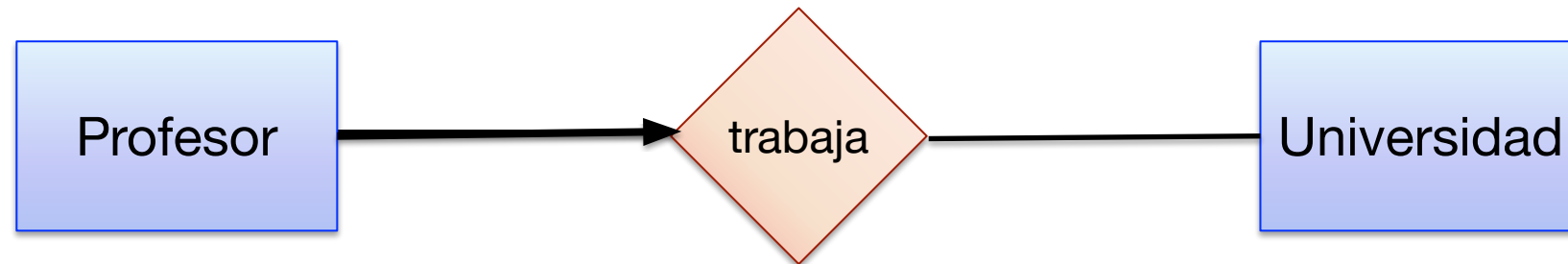
```
)
```

"codigo" es una llave foránea a la tabla curso

Borrar las tuplas de evaluación que dependan de un código en la tabla curso que fue eliminado.

Restricciones de integridad

Integridad de la entidad



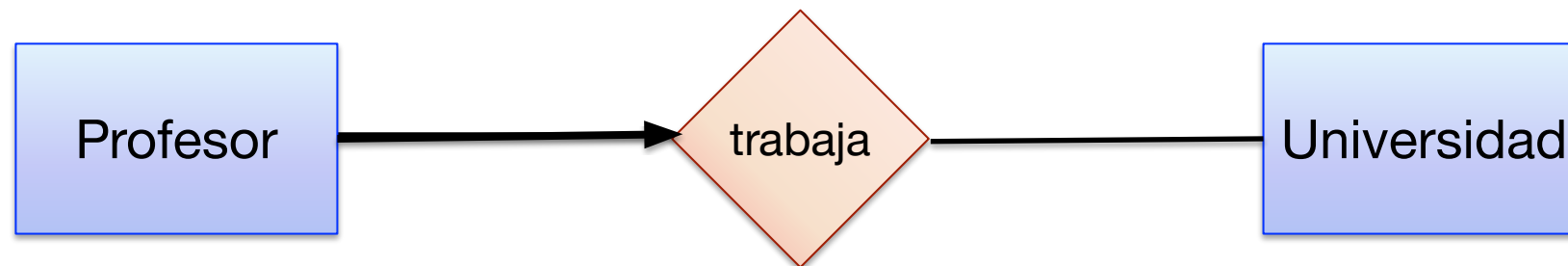
Un ejemplo de tabla de profesor con algunas restricciones:

```
CREATE TABLE Profesor(  
    id int PRIMARY KEY,  
    nombre varchar(30) NOT NULL,  
    apellidos varchar(30) NOT NULL,  
    telefono varchar(30) NOT NULL,  
    id_universidad int,  
    nivel varchar(20) DEFAULT 'Pregrado'  
    FOREIGN KEY(id_universidad) REFERENCES Universidad(id)  
)
```

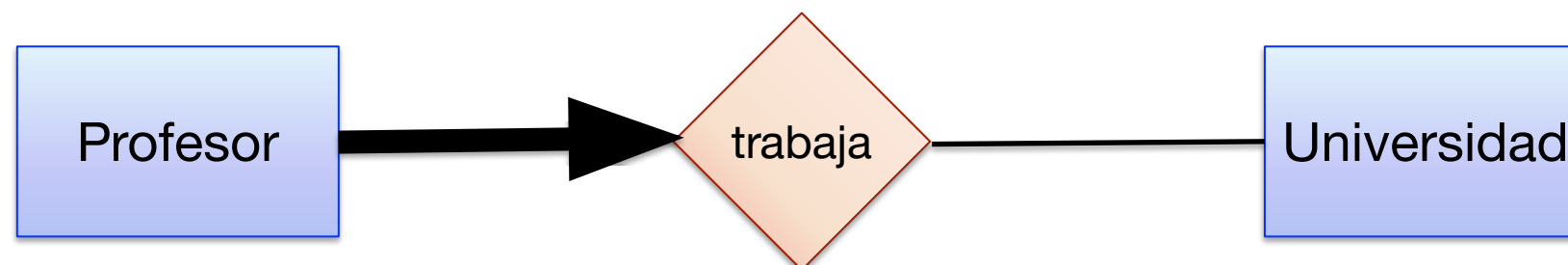

Restricciones de integridad

Participación

Cada profesor puede trabajar en una única universidad (pero puede estar sin trabajo!):



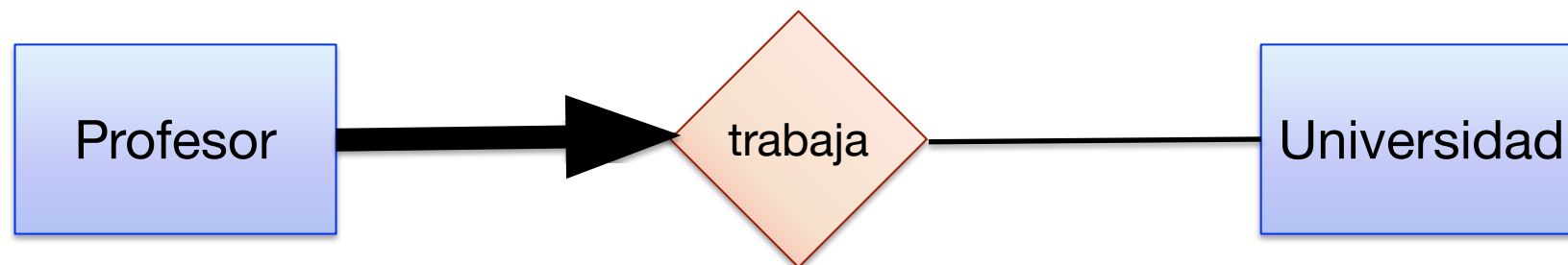
Cada profesor necesariamente trabaja en una única universidad:



Restricciones de integridad

Participación

Cada profesor necesariamente trabaja en una única universidad

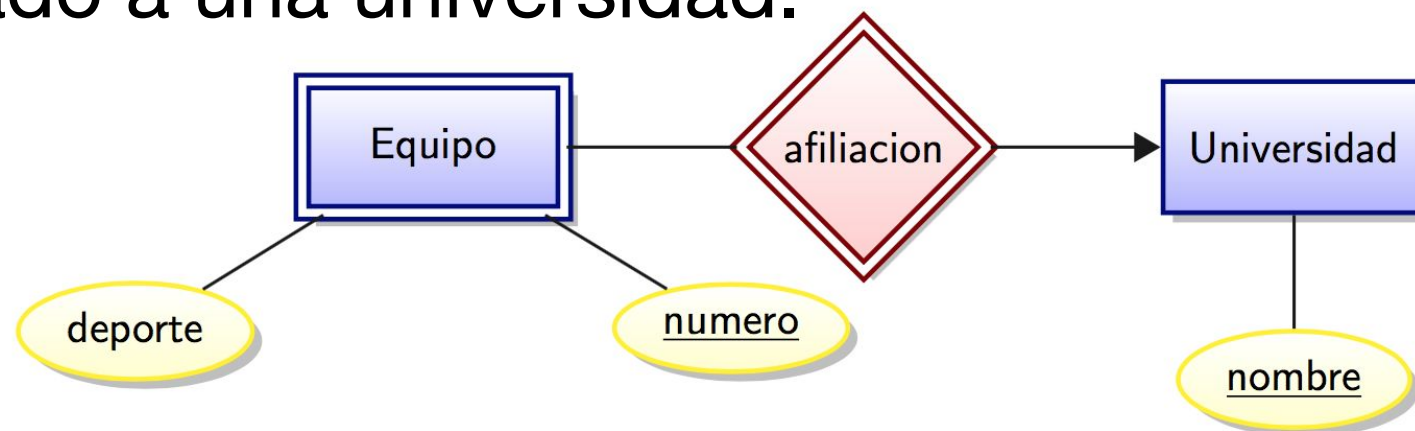


```
CREATE TABLE Profesor(  
    id int PRIMARY KEY,  
    nombre varchar(30) NOT NULL,  
    apellidos varchar(30) NOT NULL,  
    telefono varchar(30) NOT NULL,  
    id_universidad int NOT NULL,  
    nivel varchar(20) DEFAULT 'Pregrado'  
    FOREIGN KEY(id_universidad) REFERENCES Universidad(id)  
)
```

Restricciones de integridad

Entidades débiles

Le llamamos entidad débil a aquellas que necesitan de una relación para existir. En este caso, el equipo no puede existir sin estar asociado a una universidad.



```
CREATE TABLE Equipo(  
    id int PRIMARY KEY,  
    nombre varchar(30) NOT NULL,  
    id_universidad int NOT NULL,  
    FOREIGN KEY(id_universidad) REFERENCES Universidad(id) ON DELETE CASCADE  
)
```

Restricciones de integridad

Dominio

Queremos restringir el dominio de las columnas. Una forma simple de hacer esto en SQL es con **CHECK**:

```
CREATE TABLE Festival(  
    id int PRIMARY KEY,  
    nombre varchar(30) NOT NULL,  
    fecha_inicio date NOT NULL,  
    fecha_fin date NOT NULL,  
    precio int NOT NULL,  
    CHECK( precio BETWEEN 10000 AND 1000000 ),  
    CHECK( fecha_fin > fecha_inicio)  
)
```