Índice:

Estos apuntes incluyen algunas imágenes y textos del libro:

```
Libro - Adoración de Miguel - Diseño de Bases de Datos Relaciónales TEORIA, capítulo 10,
     paginas 349
```

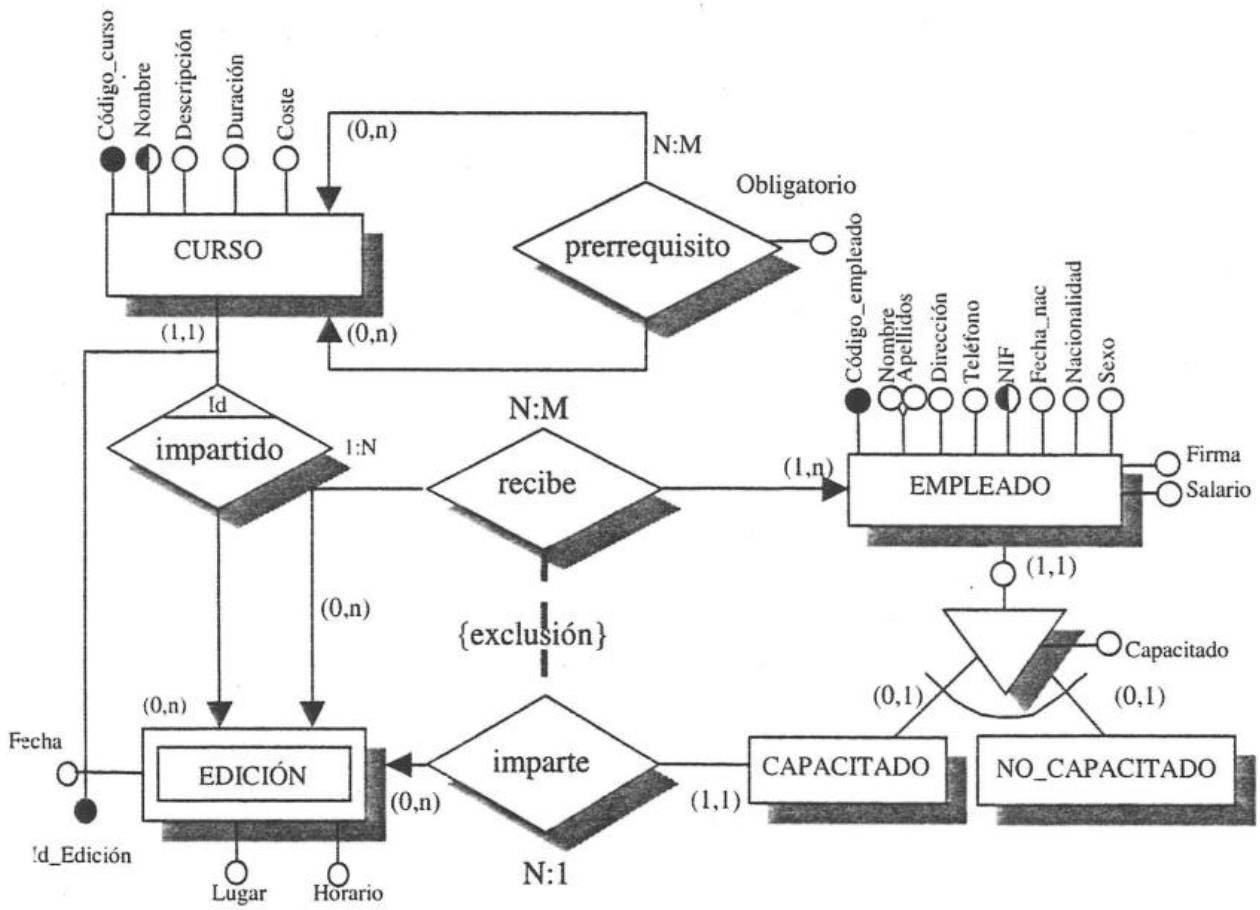
1.1 REGLAS PARA LA TRANSFORMACIÓN DEL MODELO CONCEPTUAL AL MODELO LÓGICO RELACIONAL DE DATOS

Elemento del diagrama E/R	**Transformación**
Entidad	Se crea una tabla

Elemento del diagrama E/R			**Transformación**
Relación	*Cardinalidad resumida*	*Cardinalidades extendidas*	
	1:1	(1,1) - (1,1)	Propagación de la clave. 2 Opciones: - De la entidad A a la B - De la entidad B a la A Si la relación tiene atributos propios se propagan al lado elegido.
		(0,1) - (1,1)	Propagación de la clave del lado (1,1) al lado (0,1). Si la relación tiene atributos propios se propagan al lado elegido
		(0,1) - (0,1)	Se crea una tabla que tiene por clave primaria las claves ajenas de ambas entidades
	1:N	(**1**,1) - (x,n)	No se crea una nueva tabla. La clave primaria del lado 1 se propaga como clave foránea al lado N. Si la relación tiene atributos propios se propagan al lado N.
		(**0**,1) - (x,n)	Se crea una nueva tabla que incluye como atributos a las claves primarias de las dos tablas relacionadas. La clave primaria de esta tabla es el atributo del lado N.
	1:N DÉBIL por **Existencia**	(1,1) - (x,n)	Se tratan igual que las 1:N que no son débiles, pero además: ***ON DELETE CASCADE***
	1:N DÉBIL por **Identificación**	(1,1) - (x,n)	Se tratan igual que las 1:N que no son débiles, pero además: La clave foránea pasa a formar parte de la clave primaria del lado N. ***ON DELETE CASCADE***
	M:N	(x,m) - (y,n)	-Se crea una tabla que: 1) Recibe como claves foráneas a las claves primarias de las tablas que relaciona. 2) Recibe los atributos de la relación. -La clave primaria de la nueva tabla estará formada por las claves foráneas que recibe, además hay que analizar si algún atributo de la relación debe formar parte también de esa clave primaria (habitual con información histórica)

Elemento del diagrama E/R			**Transformación**
Relación	*Cardinalidad resumida*	*Cardinalidades extendidas*	
	Ternaria o N'aria **(relación entre 3 o más tablas) N:M:P**	-	Se crea una nueva tabla que recibe como claves foráneas a las claves primarias de las tablas que relaciona. La nueva tabla tiene por clave primaria a las claves foráneas que recibe. A observar: - Si la relación tiene atributos propios, habrá que estudiar si es necesario "ampliar la clave primaria". - Si hay entidades con participaciones (1,1) o (0,1), habrá que estudiar si se puede "reducir la clave primaria" quitando de la clave primaria las claves ajenas de dichas entidades

Elemento del diagrama E/R	Reglas para su transformación
Relaciones jerárquicas	1. La **superentidad** crea una tabla a no ser que posea muy pocos atributos, en cuyo caso desaparecería. Si tiene alguna relación, suele ser preferible crear la tabla, porque si no lo hacemos habría que crear una relación para cada una de las subentidades. 2. Las **subentidades** crearán una tabla si y sólo si tienen atributos propios o bien se relacionan con otras entidades. 3. Las **subentidades** heredan la clave primaria de la superentidad y la reciben como clave ajena (con la regla ON DELETE CASCADE). 4. El carácter T*otal/Parcial* de la jerarquía y el *solapamiento/noSolopamiento* de los subtipos se pude hacer mediante CHECK, disparadores o procedimientos almacenados. 5. En el caso de tener una jerarquía que tenga en el triángulo un **atributo discriminante como *'tipo'***: a. **no Solapamiento**: el atributo *'tipo'* se sube a la superentidad y se le asigna una codificación que identifique a cada una de las subentidades. b. **Solapamiento**: se crea una tabla que almacene los distintos subtipos a que pertenece la superentidad, de la siguiente forma: Tabla_es_un (#clave_superentidad, #*tipo*)



1.2 Atributos de una relación:

- si la relación da lugar a una tabla: entonces los atributos se quedan en esa tabla.
- si la relación no da lugar a una tabla: los atributos se van a la tabla que recibe la clave ajena.

1.3 Atributos multivaluados:

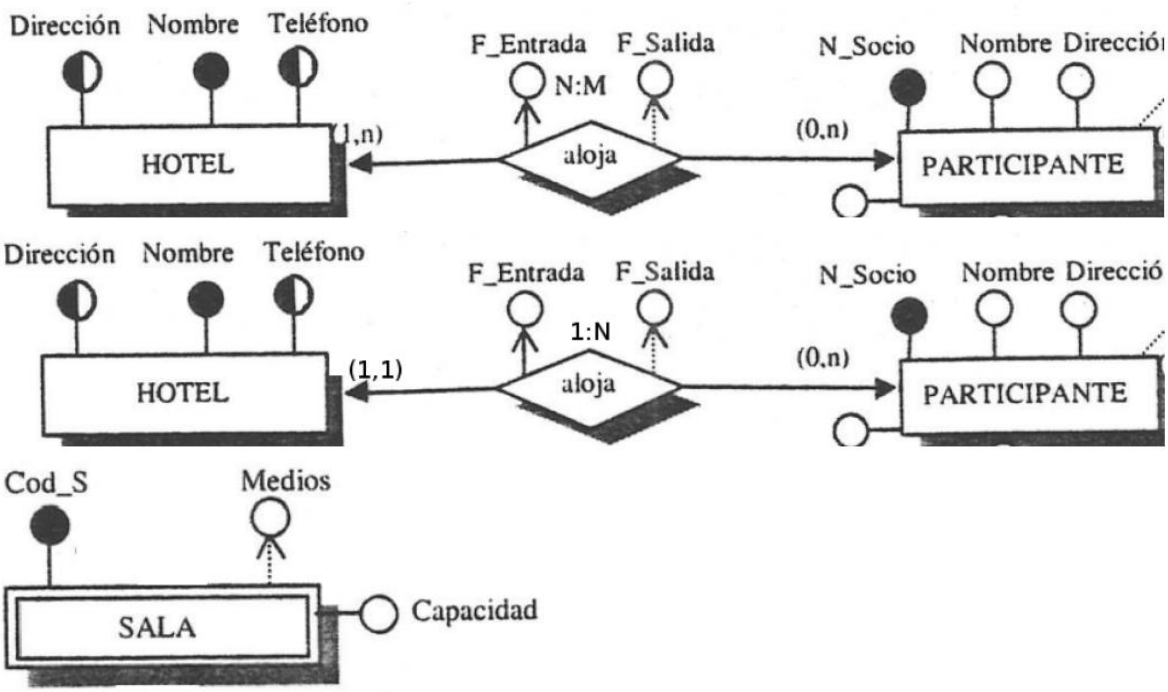
1. **Atributos Multivaluados de una <u>Entidad</u>**: Dan lugar a una nueva tabla cuya clave primaria es la concatenación de la clave primaria de la Entidad en la que se sitúa el atributo multivaluado más el nombre del atributo multivaluado.

   Si el número de valores distintos que se admite para el atributo fuese pequeño (por ejemplo 2 teléfonos máximo) entonces tendríamos la alternativa de crear atributos simples en la tabla original (telefono1, telefono2), aunque en este caso también podríamos aplicar la regla general y crear una tabla para los teléfonos.

   En ocasiones, si el atributo multivaluado no admite repeticiones, es suficiente ese atributo como clave primaria.
2. **Atributos Multivaluados de una <u>Relación N:M</u>** : la relación al ser N:M se transforma en tabla y todos los atributos de la relación (sean multivaluados o no) se añaden a esa tabla.
3. **Atributos Multivaluados de una <u>Relación 1:N</u>** : la relación al ser 1:N no se transforma en tabla , los atributos multivaluados de la relación dan lugar a una nueva tabla y se procede como se indicó antes para el caso de Atributos multivaluados de una Entidad.

```
En ocasiones es interesante perder flexibilidad para ganar simplicidad, debes de
   analizar cada caso para tomar una decisión. Por ejemplo:
- si de cada cliente queremos permitir un número ilimitado de teléfonos, emails y
   direcciones: hacen falta 3 tablas extra para los atributos multivaluados
-si restringimos alguno de esos atributos multivaluados a que tome 3 valores como
   máximo, podríamos optar por ahorrar una tabla si usamos telef1, telef2, telef3.
```

Ejemplos de atributos multivaluados:



1.4 OTROS ATRIBUTOS: obligatorios, clave alternativa, derivados, compuestos, atributos en una relación

- Atributos **Obligatorios**: *NOT NULL*
- Atributos **Clave Alternativa**: *UNIQUE*
- Atributos **derivados**: normalmente no se crean en la tabla (ejemplo: número de empleados de cada departamento), pero podrían crearse si ello ahorra trabajo al realizar las consultas. Por ejemplo, en Wallapop el número de clientes que tienen agregado a un producto como favorito es un atributo derivado y se guarda en la tabla productos.
- Atributos **compuestos**: se descomponen en los atributos que lo forman (dirección se podría descomponer en: calle, portal, piso), por tanto, se crearían 3 atributos (no 4, porque el atributo dirección no se crea, sino que se crean sus partes).
- Atributos **de una relación**: Si la relación tiene atributos y la relación no se convierte en tabla, entonces sus atributos pasan a la tabla que recibe la clave ajena.

1.5 FICHEROS ADJUNTOS:

si en la base de datos deben de guardarse ficheros adjuntos como: foto producto (jpg, gif, png...), folleto producto (pdf, doc,...) o cualquier otro tipo de fichero, lo mejor es guardar en el atributo el nombre del fichero (y posiblemente parte del PATH) y guardar físicamente los ficheros en carpetas fuera de la Base de Datos, por ejemplo:

```
./data/images/products/id_1287.jpg
```

Es cierto que los SGBD permiten guardar ficheros en su interior (tipo de datos BLOB, ...) pero en general no es lo más recomendable).

1.6 TRIGGER (disparador)

- La cardinalidad mínima o máxima de cada una de las entidades que participan en la interrelación se puede hacer mediante restricciones, aserciones o **TRIGGER**. Por ejemplo, si nos dicen que un vendedor puede atender como máximo a 15 clientes.
- Las restricciones de exclusión, exclusividad, inclusión, inclusividad se realizan mediante clausulas*CHECK* (como MySQL no dispone de ellas habrá que recurrir a TRIGGER).
- Transformación de tipos y subtipos: la exclusividad y la totalidad se realizan mediante clausulas*CHECK* (como MySQL no dispone de ellas habrá que recurrir a TRIGGER).

1.7 Utiliza siempre el mismo criterio al nombrar los elementos:

- Recuerda que algunos sistemas distinguen mayúsculas de minúsculas.
- Los nombres de los atributos, tablas y bases de datos y objetos en general, no deben de incluir espacios en blanco ni caracteres no disponibles en inglés (ñ, tildes, ...). Un atributo como "**direccion de envio**" se podría escribir como **direccionEnvio** o **direccion_envio**, escoge tu criterio.

Utiliza nombres comprensibles para cualquiera porque puede que en el futuro otra persona tenga que manipular tu BD: no ahorres tantas letras que se pierda la comprensibilidad de los nombres: *CurPrefEst* no debería emplearse en lugar de *CursosPreferidosEsstdudiantes*.

1.8 GUARDAR VALORES CONSTANTES EN TABLAS

Cuando los valores de un atributo están limitados a un conjunto predefinido es interesante crear una tabla que contenga todos los valores considerados. De este modo el usuario no tendrá que escribir, simplemente escogerá un valor de la lista. Ejemplos:

```
-provincias de España: A Coruña, Almeria, ...
-países de Europa: España, Grecia, Italia, ...
-estado civil: soltero, casado, viudo, ...
-tipo de carné de conducir: A1, B1, ...
-sexo: H, M
-talla de ropa: XXS, XS, S, ...
```

Podemos guardar en una tabla con 2 atributos (**clave : valor**) valores constantes como estos: `número máximo de matriculaciones en un módulo, límite máximo de crédito de un cliente cualquiera, plazo máximo de renovación de un libro en la biblioteca`. De este modo facilitamos el trabajo del programa que usa la BD. Ejemplo de **TABLA DE CONSTANTES**:

```
-numeroMaximMatriculacionesEnUnModulo: 3
-limiteMaximoCreditoCliente: 1500
-plazoMaximoRenovacionLibro: 15
```

1.9 CÓMO ALMACENAR INFORMACIÓN HISTÓRICA:

El siguiente ejemplo resuelto proviene del libro (página 27):

```
LIBRO - Diseño De Bases De Datos, Problemas Resueltos, Ed. Rama, Adoración de Miguel
```

Debemos fijarnos en las cardinalidades de la relación aloja: Ciertamente cada día el participante se aloja en un solo hotel, pero se está guardando información histórica (se va guardando información de los distintos hoteles en que se ha ido alojando durante el torneo, con sus fechas), por ello la cardinalidad reflejada es:*

```
Un participante se relaciona con (1,n) hoteles porque: se ha alojado en al menos 1 y como
     máximo n a lo largo del tiempo.
```

Es necesario añadir a este diagrama un texto que indique que cada día puede estar en un solo hotel, porque esta restricción no se puede representar en el diagrama

F_Entrada y F_Salida son multivaluados en el Diagrama de la imagen, pero realmente no tendría por qué indicarse así, podrían ser atributos normales (univaluados).

1) TEXTO DEL PROBLEMA

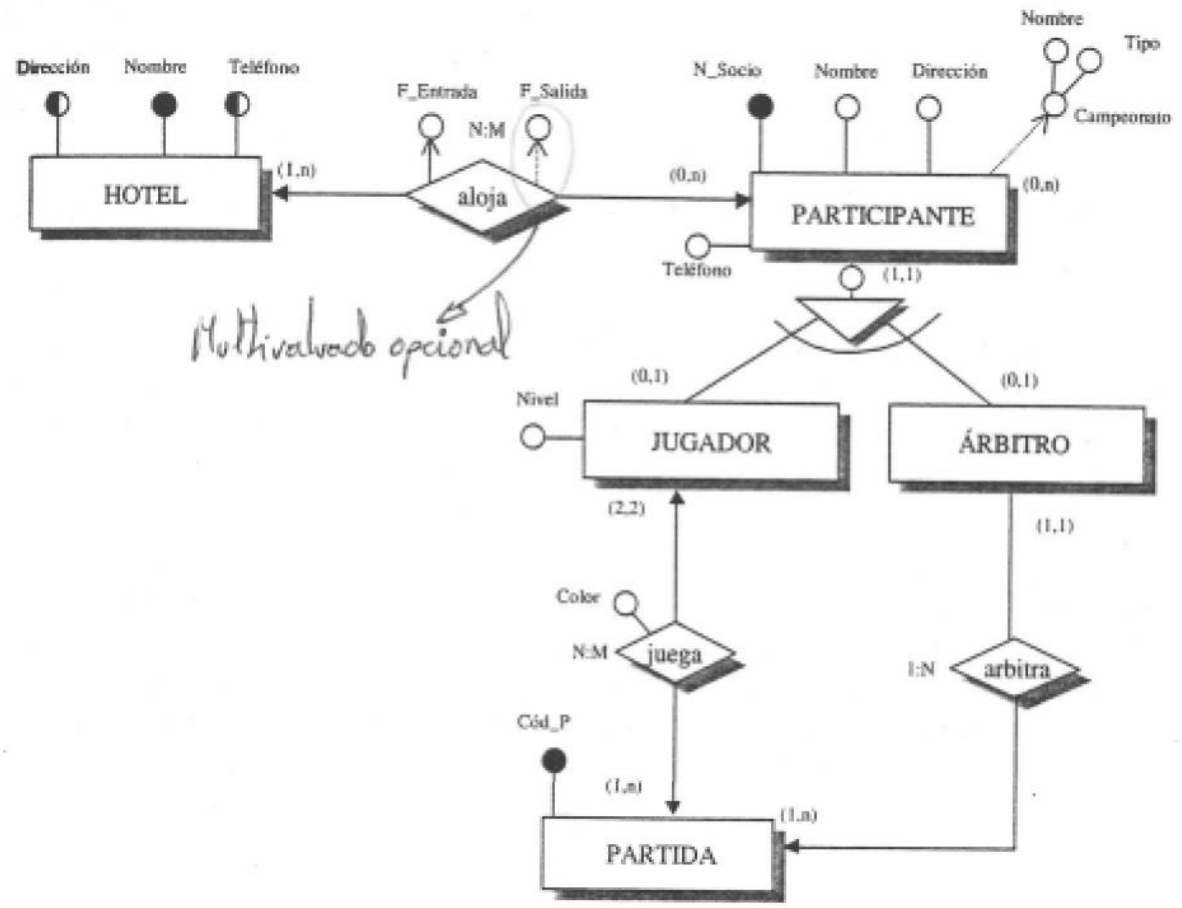
"Tanto jugadores como árbitros se alojan en uno de los hoteles en los que se desarrollan las partidas, se desea conocer en qué hotel y en qué fechas se ha alojado cada uno de los participantes. Los participantes pueden no permanecer en Villatortas durante todo el campeonato, sino acudir cuando tienen que jugar alguna partida alojándose en el mismo o distinto hotel. De cada hotel, se desea conocer el nombre, la dirección y el número de teléfono."

Como los participantes se alojan en uno o varios hoteles durante el campeonato, aunque no es obligatorio que permanezcan en él durante todo el campeonato, se creará una entidad *HOTEL* cuyo identificador principal será "Nombre" y como atributos alternativos se presentarán "Dirección" y "Teléfono". Y se relacionarán *HOTEL* y *PARTICIPANTE* mediante una interrelación **Aloja** cuyas cardinalidades asociadas serán (0,n) y (1,n), respectivamente. Con la cardinalidad mínima 0 para *HOTEL* se contemplará que en un momento dado puede que no existan participantes alojados en el mismo. Además, las estancias de un participante pueden ser intermitentes, con lo cual la interrelación posee dos atributos multivaluados, "F_Entrada" y "F_Salida", que recogerán este supuesto. Para reflejar las instancias acutales, en las cuales, evidentemente no habrá una fecha de salida, el atributo "f_salida", será opcional.

Como restricción adicional deducida de la interpretación del Universo de Discurso se debería asegurar que la fecha de salida sea mayor que la de entrada y que los periodos de tiempo no se solapen entre sí.

La semántica anterior aparece reflejada en el esquema de la figura 1.15.

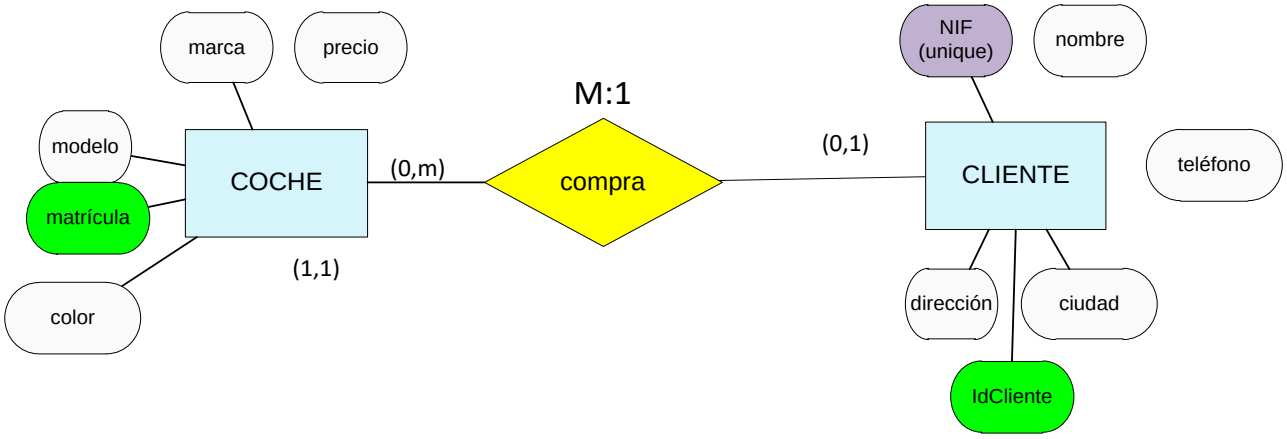
2) **MODELO ER resultante:**


Dirección
Nombre
Teléfono
HOTEL
(1,n)
F_Entrada
F_Salida
N:M
aloja
(0,n)
N_Socio
Nombre
Dirección
Nombre
Tipo
Campeonato
PARTICIPANTE
(0,n)
Teléfono
(1,1)
Multivaluado opcional
(0,1)
(0,1)
Nivel
JUGADOR
ÁRBITRO
(2,2)
(1,1)
Color
N:M
juega
1:N
arbitra
Cód_P
(1,n)
(1,n)
PARTIDA

2. Apéndice

2.1 Como transformar una relación 1:N

Figura V.119



- **Opción 1:**

  Esta manera de pasar el modelo ER a tablas es la preferible porque evita valores nulos.

COCHE
matricula
1111AAA
2222BBB
3333CCC
4444DDD

COMPRA	
matricula	***idCliente***
1111AAA	2
3333CCC	1

CLIENTE
idCliente
1
2

- **Opción 2:**

  Esta manera de pasar el modelo ER a tablas es válida, pero no es recomendable, porque genera valores nulos y eso ralentiza al servidor cuando tiene que realizar tareas como la búsqueda de datos (SELECT) y las operaciones de INNER JOIN.

COCHE	
matricula	***idClienteComprador***
1111AAA	2
2222BBB	**NULL**
3333CCC	1
4444DDD	**NULL**

CLIENTE
idCliente
1
2
3