Tema 7. Diseño Relacional

1. Introducción

El diseño de una base de datos consiste en definir la estructura de los datos que debe tener un sistema de información determinado. Para ello se suelen seguir por regla general unas fases en el proceso de diseño, definiendo para ello el modelo conceptual, el lógico y el físico.

- En el diseño conceptual se hace una descripción de alto nivel de la estructura de la base de datos, independientemente del SGBD (Sistema Gestor de Bases de Datos) que se vaya a utilizar para manipularla. Su objetivo es describir el contenido de información de la base de datos y no las estructuras de almacenamiento que se necesitarán para manejar dicha información.
- El diseño lógico parte del resultado del diseño conceptual y da como resultado una descripción de la estructura de la base de datos en términos de las estructuras de datos que puede procesar un tipo de SGBD. El diseño lógico depende del tipo de SGBD que se vaya a utilizar, se adapta a la tecnología

que se debe emplear, pero no depende del producto concreto. En el caso de bases de datos convencionales relacionales (basadas en SQL para entendernos), el diseño lógico consiste en definir las tablas que existirán, las relaciones entre ellas, normalizarlas, etc...
- El diseño físico parte del lógico y da como resultado una descripción de la implementación de una base de datos en memoria secundaria: las estructuras de almacenamiento y los métodos utilizados para tener un acceso eficiente a los datos. Aquí el objetivo es conseguir una mayor eficiencia, y se tienen en cuenta aspectos concretos del SGBD sobre el que se vaya a implementar. Por regla general esto es transparente para el usuario, aunque conocer cómo se implementa ayuda a optimizar el rendimiento y la escalabilidad del sistema.

2. Modelado de datos

Según el ciclo de vida clásico, podemos definir dos fases de trabajo (que ya estudiamos en el Tema 1 de Entornos de Desarrollo): Análisis y Diseño (además de la implementación).

- En el **Análisis**, estudiamos las actividades de la organización o sistema para la que queremos desarrollar nuestra base de datos, identificando muy bien qué datos maneja y cuáles son las relaciones entre esos datos. Lo habitual es que como producto de este análisis obtengamos un **esquema conceptual** representado mediante un **diagrama**.
- En el Diseño, creamos un **modelo lógico** para el **esquema conceptual** de la fase anterior. En nuestro caso, será el producto será **grafo relacional**, ya que vamos a implementar la solución con un SGBD relacional.

Llevando a cabo una correcta fase de análisis estaremos dando un paso determinante en el desarrollo de nuestras bases de datos. El hecho de saltarse el esquema conceptual conlleva un problema de pérdida de información respecto al problema real a solucionar. El esquema conceptual debe reflejar todos los aspectos relevantes del mundo real que se va a modelar.

3. Modelo entidad/relación

Es una herramienta de referencia para la representación conceptual de problemas del mundo real. Su objetivo principal, facilitar el diseño de bases de datos permitiendo la especificación de un esquema que representa la estructura lógica completa de una base de datos. Este esquema partirá de las descripciones textuales de la realidad, que establecen los requerimientos del sistema, buscando ser lo más fiel posible al comportamiento del mundo real para modelarlo.

El modelo de datos E-R representa el significado de los datos, es un modelo semántico. De ahí que no esté orientado a ningún sistema físico concreto y tampoco tiene un ámbito informático puro de aplicación, ya que podría utilizarse para describir procesos de producción, estructuras de empresa, etc. Además, las características actuales de este modelo favorecen la representación de cualquier tipo de sistema y a cualquier nivel de abstracción o refinamiento, lo cual da lugar a que se aplique tanto a la representación de problemas que vayan a ser tratados mediante un sistema informatizado, como manual.

Gracias al modelo Entidad-Relación, creado por Peter Chen en los años setenta, se puede representar el mundo real mediante una serie de símbolos y expresiones determinados. El modelo de datos

Entidad/Relación (E/R ó E-R) está basado en una percepción consistente en objetos básicos llamados entidades y relaciones entre estos objetos, estos y otros conceptos se desarrollan a continuación.

> Nosotros *no vamos a utilizar la notación tradicional* para el modelo entidad/relación, ya que los diagramas suelen ser (en mi opinión) bastante toscos. Vamos a representar este modelo **con notación UML**, que además ya conocemos a través del Tema 5 de Entornos de Desarrollo.

3.1 Entidades

Si utilizamos las bases de datos para guardar información sobre cosas que nos interesan o que interesan a una organización, ¿No crees que hay que identificar esas cosas primero para poder guardar información sobre ellas? Para ello, vamos a describir un primer concepto, el de **Entidad**.

Una **entidad** puede ser un objeto físico, un concepto o cualquier elemento que queramos modelar, que tenga importancia para la organización y del que se desee guardar información. Cada entidad debe poseer alguna característica, o conjunto de ellas, que lo haga único frente al resto de objetos. Por ejemplo, podemos establecer una entidad llamada ALUMNO que tendrá una serie de características. El alumnado podría ser distinguido mediante su número de identificación escolar (NIE), por ejemplo.

> **Entidad**: objeto real o abstracto, con características diferenciadoras capaces de hacerse distinguir de otros objetos.

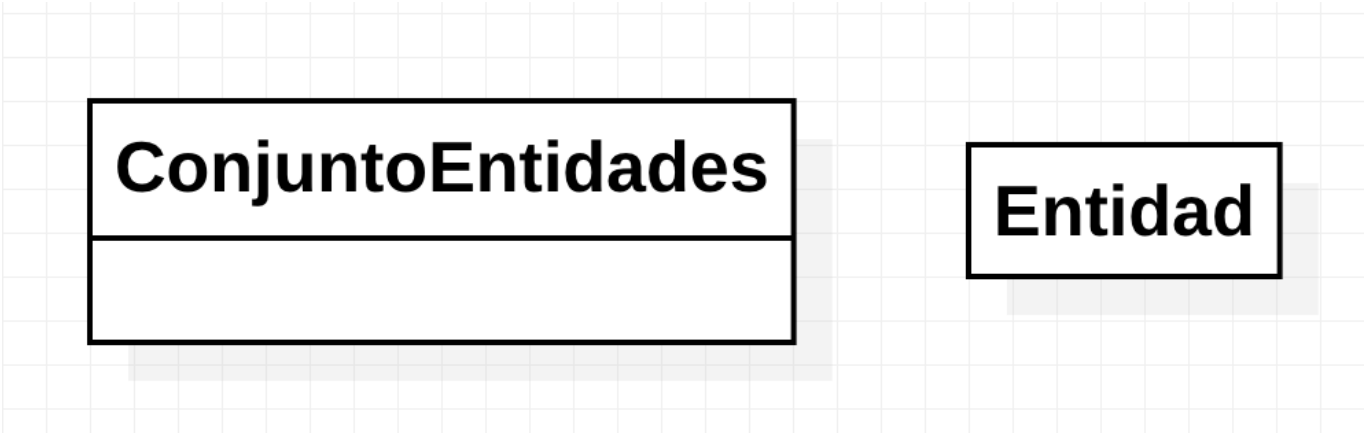
¿Ponemos otro ejemplo? Supongamos que tienes que desarrollar el esquema conceptual para una base de datos de mapas de montaña, los elementos: camping, pista forestal, valle, río, pico, refugio, etc., son ejemplos de posibles entidades. A la hora de identificar las entidades, hemos de pensar en nombres que tengan especial importancia dentro del lenguaje propio de la organización o sistema que vaya a utilizar dicha base de datos. Pero no siempre una entidad puede ser concreta, como un camping o un río, en ocasiones puede ser abstracta, como un préstamo, una reserva en un hotel o un concepto.

Un **conjunto de entidades** serán un grupo de entidades que poseen las mismas características o propiedades. Por ejemplo, al conjunto de personas que realizan reservas para un hotel de montaña determinado, se les puede definir como el conjunto de entidades cliente. El conjunto de entidades río, representará todos los ríos existentes en una determinada zona. Por lo general, se suele utilizar el término entidad para identificar conjuntos de entidades. Cada elemento del conjunto de entidades será una ocurrencia de entidad.

Si establecemos un símil con la Programación Orientada a Objetos, podemos decir que el concepto de entidad es análogo al de instancia de objeto y que el concepto de conjunto de entidades lo es al de clase.

Por tanto, dado que hemos indicado que vamos a representar el modelo entidad/relación con *notación UML*, para representar una entidad usaremos el símbolo de un objeto, y para representar al conjunto de entidades el símbolo de una clase.

> Habitualmente, se suprime el espacio para operaciones, ya que no tiene sentido en este modelo.



Las entidades dentro del modelo E/R se suelen dividir entre:

- **Entidades Fuertes o Regulares**: Son aquellas que tienen existencia por sí mismas, es decir, su existencia no depende de la existencia de otras entidades. Por ejemplo, en una base de datos hospitalaria, la existencia de instancias concretas de la entidad DOCTOR no depende de la existencia de instancias u objetos de la entidad PACIENTE. En el modelo E/R las entidades fuertes se representan como hemos indicado anteriormente, con el nombre de la entidad encerrado dentro de un rectángulo. La forma de representar es la que hemos visto anteriormente
- **Entidades débiles**: Son aquellas cuya existencia depende de la existencia de otras instancias de entidad. Por ejemplo, consideremos las entidades EDIFICIO y AULA. Supongamos que puede haber aulas identificadas con la misma numeración pero en edificios diferentes. La numeración de cada aula no identificará completamente cada una de ellas. Para poder identificar completamente un aula es necesario saber también en qué edificio está localizada. Por tanto, la existencia de una instancia de una entidad débil depende de la existencia de una instancia de la entidad fuerte con la que se relaciona.

> ¿Cómo representamos las entidades débiles con la notación UML? Tenemos varias alternativas, aunque si nos fijamos:
>
> - La existencia de la entidad débil se supedita a la existencia de una entidad fuerte.
> - Por tanto, ambas estarán asociadas.
> - Si se elimina una entidad fuerte, desaparecen todas las entidades débiles asociadas a ellas. En UML, todas estas premisas anteriores se representan mediante una **asociación de composición**.

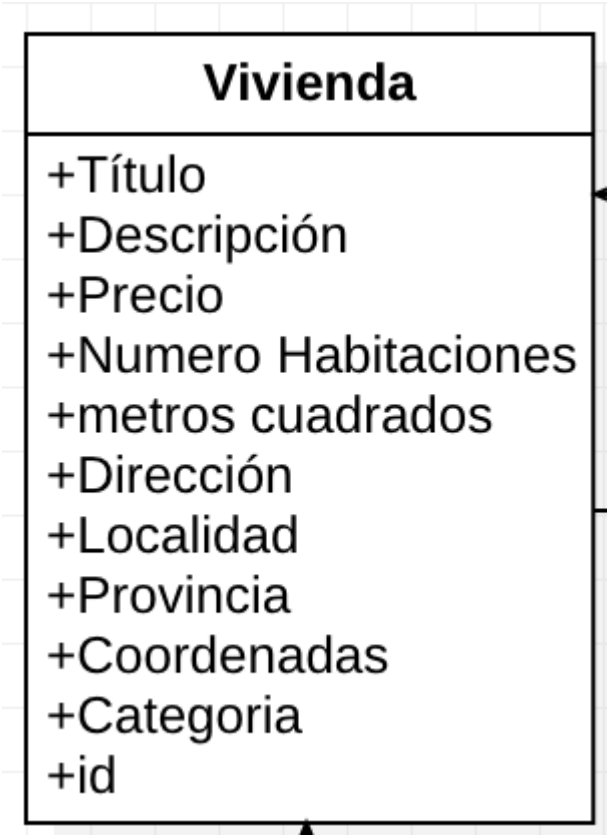
3.2 Atributos

¿Cómo guardamos información de cada entidad? A través de sus **atributos**. Las entidades se representan mediante un conjunto de **atributos**. Éstos describen características o propiedades que posee cada miembro de un conjunto de entidades. El mismo atributo establecido para un conjunto de entidades o, lo que es lo mismo, para un tipo de entidad, almacenará información parecida para cada ocurrencia de entidad. Pero, cada ocurrencia de entidad tendrá su propio valor para cada atributo.

Atributo: Cada una de las propiedades o características que tiene un tipo de entidad o un tipo de relación se denomina atributo; los atributos toman valores de uno o varios dominios.

Por tanto, un atributo se utilizará para guardar información sobre alguna característica o propiedad de una entidad o relación. Ejemplos de atributos pueden ser: altura, color, peso, DNI, fecha, etc. todo dependerá de la información que sea necesaria almacenar.

> En el modelo entidad/relación utilizando la notación de UML, los atributos se representan como si fueran las propiedades de una clase.

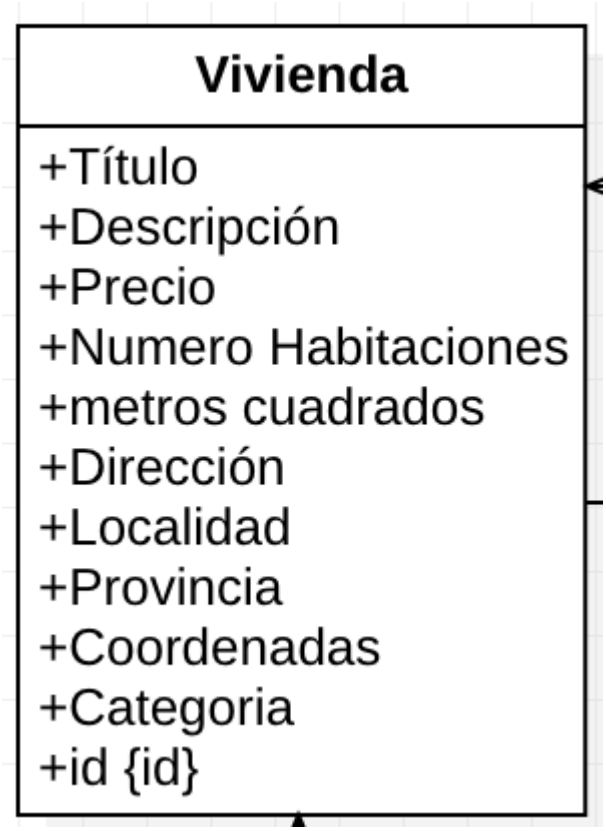


3.3 Claves

En el Tema 2, sobre modelo relacional, estuvimos trabajando ampliamente los conceptos de superclave, clave candidata y clave primaria. Si recordamos las definiciones:

- Superclave: Es cualquier conjunto de atributos que permite identificar de forma única a una ocurrencia de entidad (una fila). Una superclave puede tener atributos no obligatorios, es decir, que no identificarían por si solos una ocurrencia de entidad.
- Clave candidata: Si de una superclave no es posible obtener ningún subconjunto que sea a su vez superclave, decimos que dicha superclave es clave candidata.
- **Clave primaria**: De todas las claves candidatas, el diseñador de la base de datos ha de escoger una, que se denominará clave principal o clave primaria. La clave primaria es un atributo o conjunto de ellos, que toman valores únicos y distintos para cada ocurrencia de entidad, identificándola unívocamente. No puede contener valores nulos.

> En el modelo entidad/relación utilizando notación UML no hay una forma muy clara de establecer los atributos como clave primaria. Se fuera posible, se deberían indicar en negrita. Si no es posible, se le puede añadir una almohadilla delante a aquel (o aquellos) atributo que conforme la clave primaria. StarUML tiene la una opción entre las propiedades de un atributo que es *isID*, el cuál también podríamos utilizar para indicar que un atributo es o forma parte de la clave primaria



4. Interrelaciones o asociaciones

> Se suele utilizar el término interrelación en lugar de relación para no tener ambigüedad entre la asociación entre dos entidades y una tabla del modelo relacional.

¿Cómo interactúan entre sí las entidades? A través de las interrelaciones. La interrelación es un elemento del modelo Entidad/Relación que permite asociar datos entre sí. En una interrelación se asocia un elemento de una entidad con otro de otra entidad.

Interrelación: es una asociación entre diferentes entidades. En una interrelación no pueden aparecer dos veces relacionadas las mismas ocurrencias de entidad.

> En el modelo entidad/relación con notación UML, las interrelaciones se representan como asociaciones entre las clases que representan las entidades.

> Cuando debas dar un nombre a una interrelación procura que éste haga referencia al objetivo o motivo de la asociación de entidades. Se suelen utilizar verbos en singular. Algunos ejemplos podrían ser: forman, poseen, atiende, contrata, hospeda, supervisa, imparte, etc.

En algunas ocasiones, es interesante que en las líneas que conectan las entidades con la interrelación, se indique el papel o rol que desempeña cada entidad en la interrelación. Esto se puede representar utilizan la notación correspondiente de UML, dando un nombre a cada extremo de la asociación.

Al igual que hemos hecho en Entornos de Desarrollo, por lo general **trabajaremos con interrelaciones binarias**, y no de grado superior, ya que son mucho más difíciles de manejar.

4.1 Cardinalidad o multiplicidad

Es el número máximo de ocurrencias de cada entidad que pueden intervenir en una ocurrencia de una interrelación. La cardinalidad vendrá expresada siempre para relaciones entre dos entidades. Dependiendo del número de ocurrencias de cada una de las entidades pueden existir relaciones **uno a uno**, **uno a muchos**, **muchos a uno** y **muchos a muchos**.

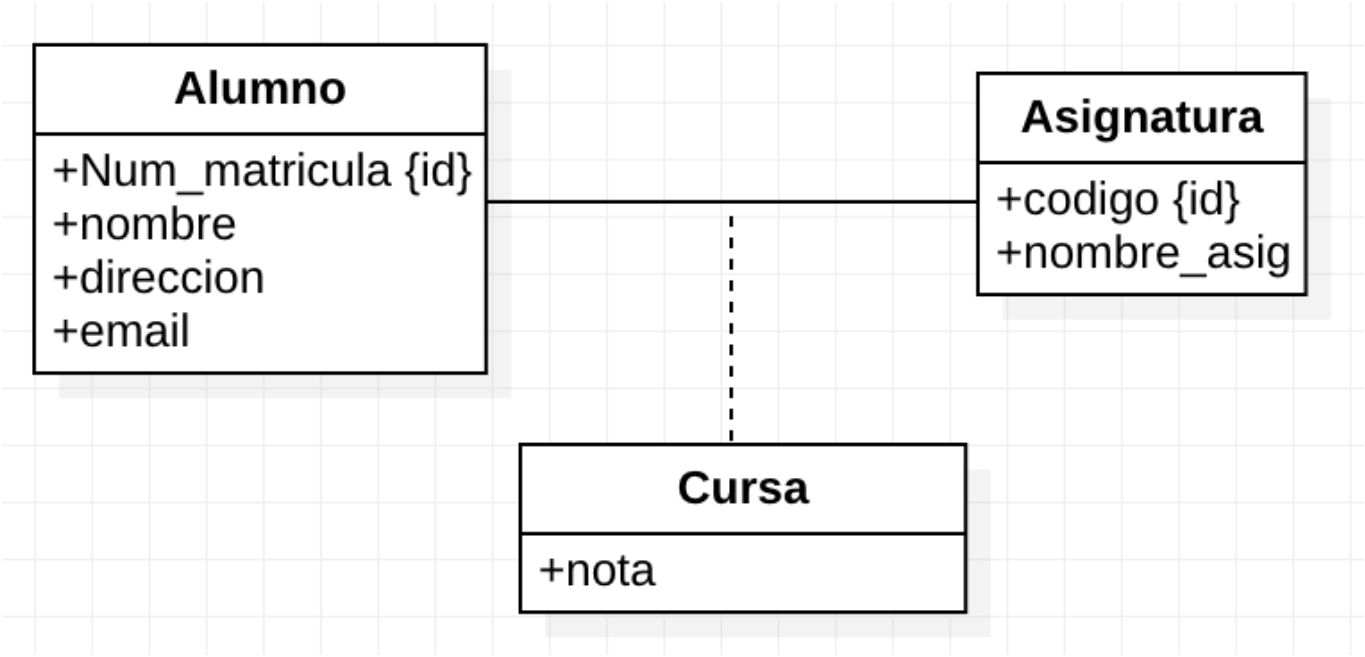
En la notación clásica del modelo entidad/relación suelen representarse como 1:1, 1:N, N:1 y N:M.

> En el modelo entidad/relación con notación UML, usaremos los símbolos habituales, como son los números o el *.

4.2 Atributos de una interrelación

Una interrelación puede también tener atributos que la describan. Para ilustrar esta situación, observa el siguiente ejemplo:

Consideremos la interrelación CURSA entre las entidades ALUMNO y ASIGNATURA. Podríamos asociar a la interrelación CURSA un atributo nota para especificar la nota que ha obtenido un alumno/a en una determinada asignatura.



> La forma de representar esta situación usando notación UML es mediante el uso de una **clase de asociación**, añadiendo a la misma los atributos de la interrelación.

4.3 Tipos de interrelaciones

Es habitual encontrar interrelaciones (representadas como asociaciones) entre entidades (representadas a través de clases) de los diferentes tipos que hemos visto en los diagramas de clases UML, a saber:

- Composición
- Agregación
- Herencia

> **El tipo de asociación será muy importante a la hora de trasladar nuestro modelo entidad/relación a un grafo relacional**.

En el caso de las asociaciones de herencia, en la literatura de modelo relacional se suelen encontrar como interrelaciones is-a o es-un, y su representación suele ser igual que las asociaciones de herencia habituales en UML.

5. Representación de un grafo relacional

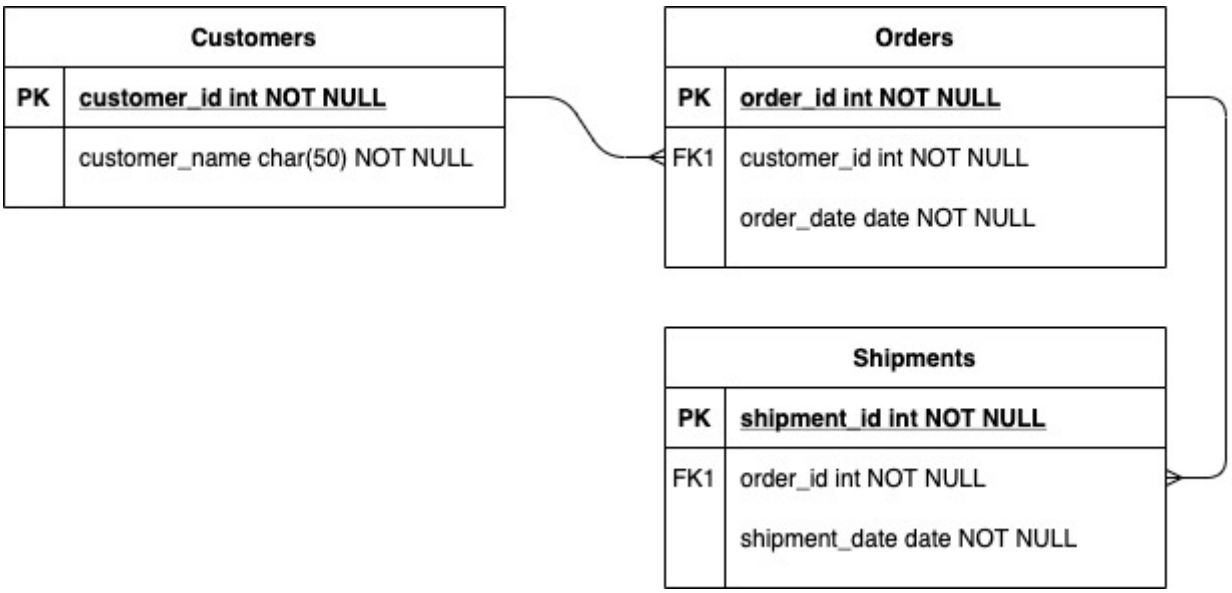
Hasta ahora, la mayoría de los grafos relacionales que hemos representados los hemos hecho en papel o pizarra, lo cual no le quita validez. Las reglas que hemos seguido han sido:

- Cada tabla se representa por su esquema, indicando el nombre de la misma en mayúsculas.
- Los atributos de una tabla se representan (habitualmente en minúsculas) entre paréntesis, separados por comas, a continuación del nombre la tabla.
- El atributo (o los atributos) que conformen la clave primaria se deben subrayar.
- Las claves externas se representan como flechas que salen de aquellos atributos que la conforman hacia las tablas a las cuales hacen referencia.

Sin embargo, existen herramientas, así como una sintaxis, que nos van a permitir representar de una forma mejor nuestros grafos.

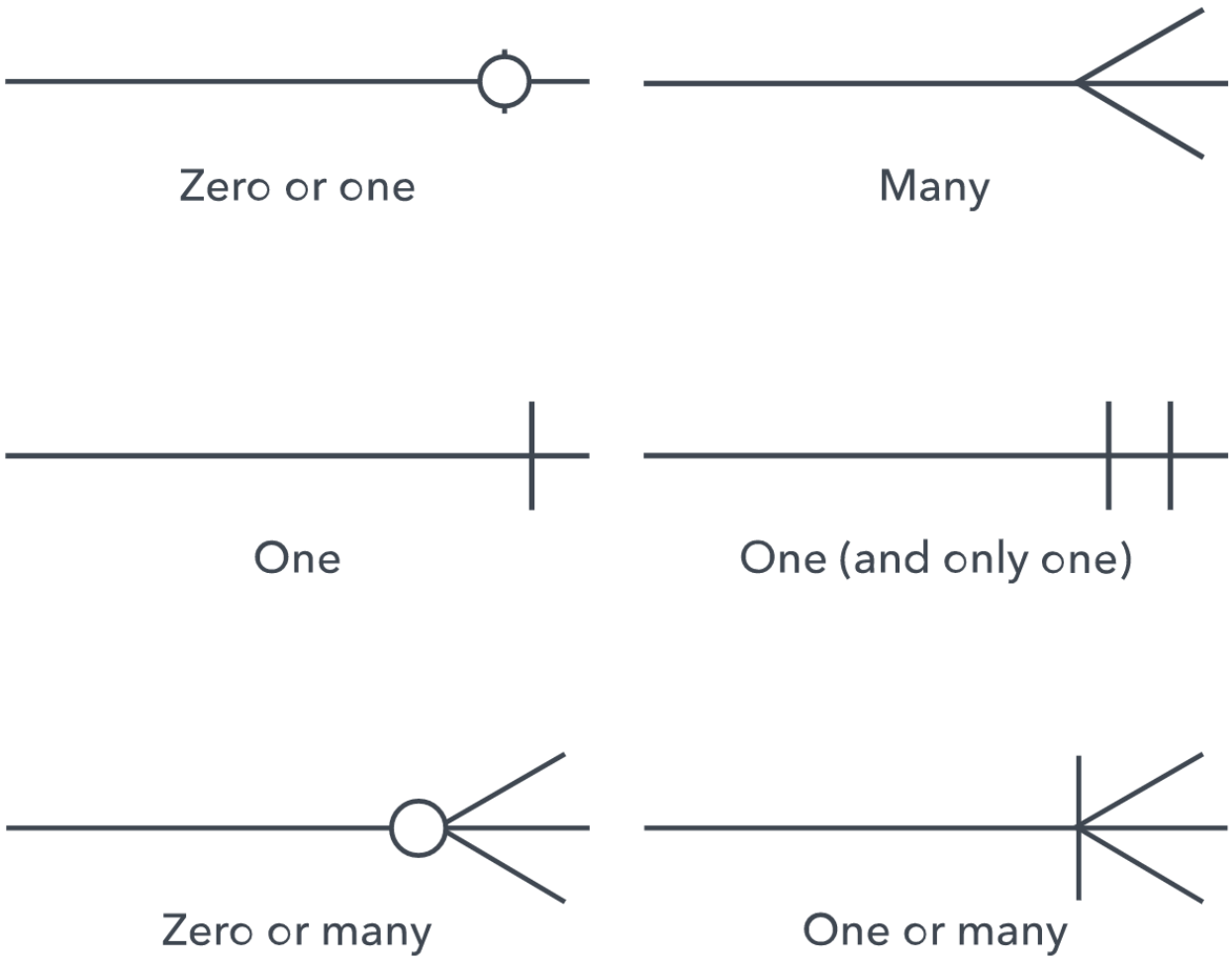
- La herramienta dbdiagram.io nos permite hacerlo de una forma bastante interesante: en lugar de tener un área con los componentes gráficos a dibujar y seguir el principio de *drag&drop*, nuestras tablas, atributos y restricciones se definen a través de un DSL (*Domain Specific Language*). Tiene cuentas freemium y permite tener gratis un número limitado de diagramas.
- Podemos utilizar también como herramienta para representar nuestro grafo relacional un diagrama entidad relación (¿seguro? sí, seguro) con notación de *patas de gallo*. Es una notación especial de e/r que utiliza una simbología especial para indicar la multiplicidad de las asociaciones. Podemos utilizar herramientas gratuítas como draw.io para poder generar este tipo de diagramas.

Un ejemplo de este tipo de diagrama podría ser el siguiente:

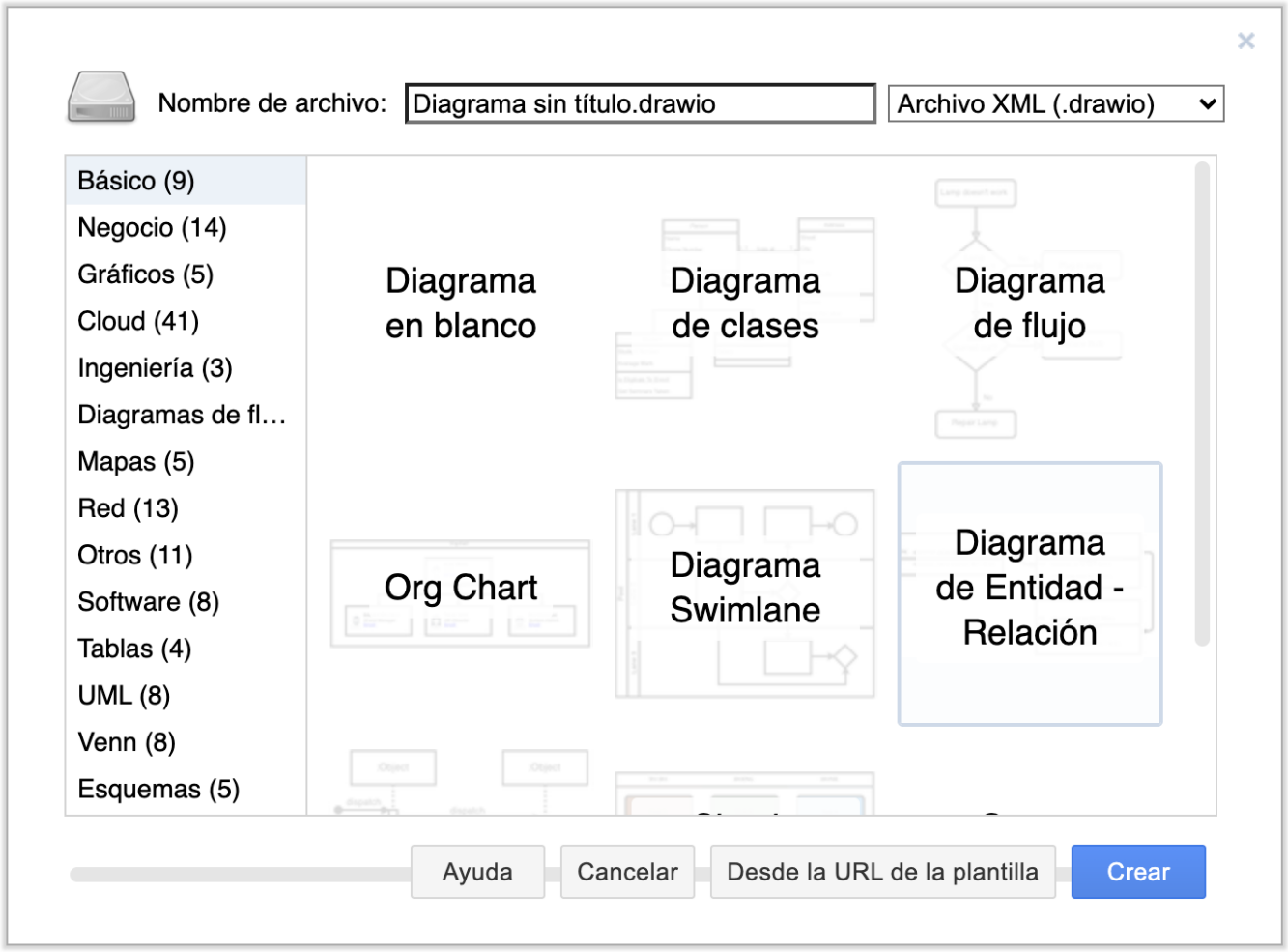


> En este enlace puedes encontrar el instalador de draw.io para tu sistema operativo: https://github.com/jgraph/drawio-desktop/releases

Este tipo de diagramas se llama de **patas de gallo** por la forma de representar la cardinalidad. Si utiliza la siguiente simbología:



Para crear un diagrama de este tipo tenemos que escoger *Nuevo diagrama > Básico > Entidad Relación*:



6. Paso del modelo entidad/relación a modelo/relacional

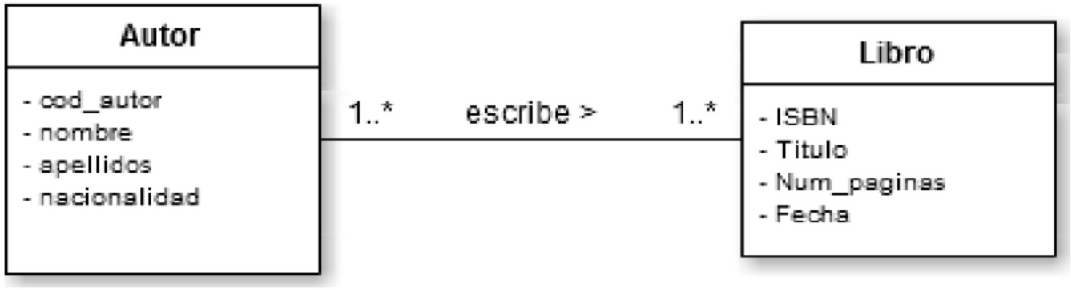
A lo largo de este apartado vamos a aprender un método casi sistemático para convertir un modelo entidad/relación (representado a partir de un diagrama de clases UML) en un grafo relacional. Por tanto, vamos a recibir como **entrada un diagrama entidad/relación** (representado con notación UML), y vamos a producir como **salida un grafo relacional** (probablemente como un diagrama con notación de patas de gallo)

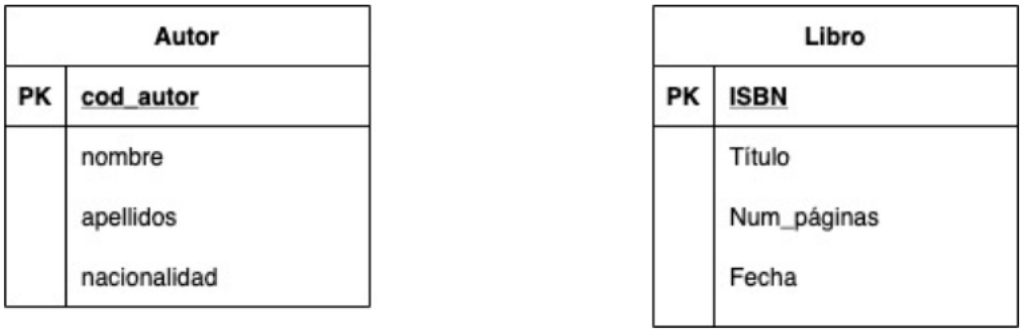
6.1 Traslado de entidades (clases)

Toda entidad (es decir, toda clase) del diagrama entidad relación (de clases) da como resultado una nueva tabla en el grafo relacional.

El nombre de la tabla del esquema relacional será el nombre de la clase, y los atributos de la tabla serán las propiedades de la clase.

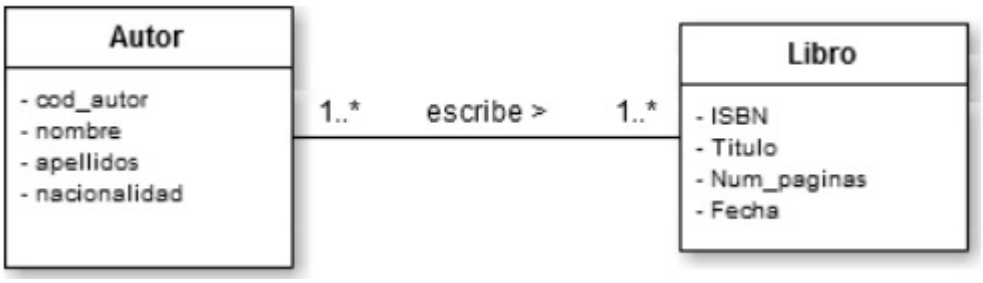
> La herencia será una casuística especial que tendremos en cuenta más adelante.





6.2 Traslado de asociaciones

En un diagrama entidad/relación, una interrelación o asociación expresa que existe una *relación de influencia* entre, la menos, dos entidades.



En el ejemplo de la imagen anterior, la asociación **escribe** denota que un **Autor** escribe **Libros**, y ese hecho debe almacenarse en nuestra base de datos.

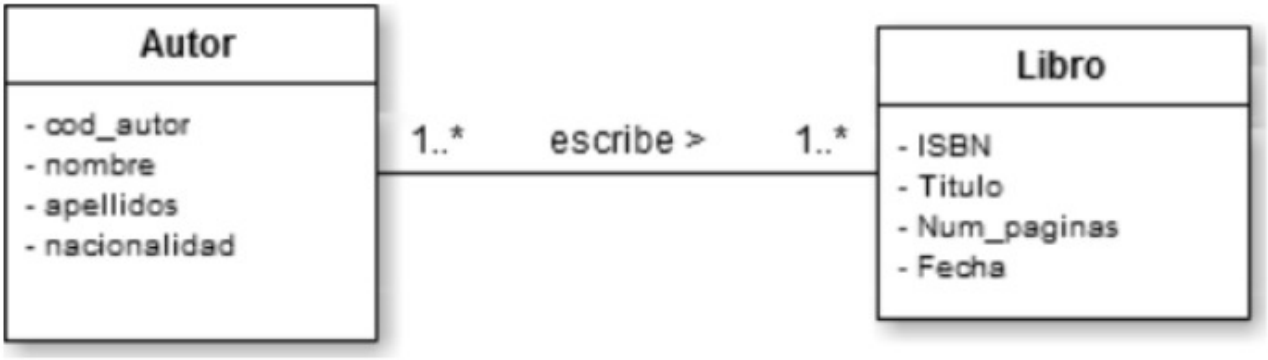
Además, ya hemos visto que las asociaciones tienen *cardinalidad*, la cual expresa el número de objetos de unt ipo que se pueden relacionar con el número de objetos de otro tipo.

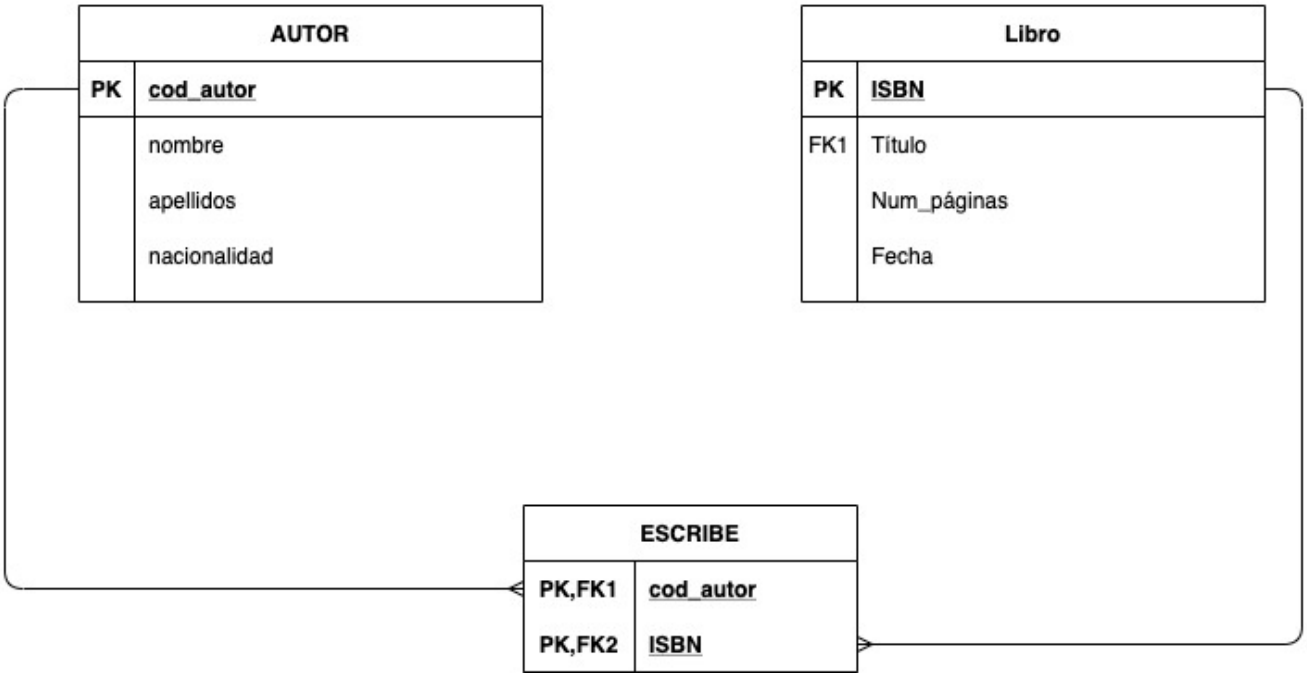
La forma de trasladar una interrelación o asociación del modelo entidad/relación al modelo relacional dependerá de la cardinalidad de la misma.

6.2.1 Traslado de asociaciones muchos a muchos

Dos conjuntos de entidades, A y B, tienen entre ellas una interrelación de cardinalidad muchos a muchos, cuando muchas entidades del tipo A se pueden relacionar con una entidad del tipo B, y muchas entidades del tipo B se pueden relacionar con una entidad del tipo A.

Toda asociación N – M en un diagrama de clases, da como resultado una tabla en el esquema relacional resultante. El nombre de la tabla será el nombre que tuviera la asociación (o en su caso, el nombre de las entidades que asocie, separado por un guión bajo), y los atributos serán aquellos que formen las claves primarias de ambas entidades.



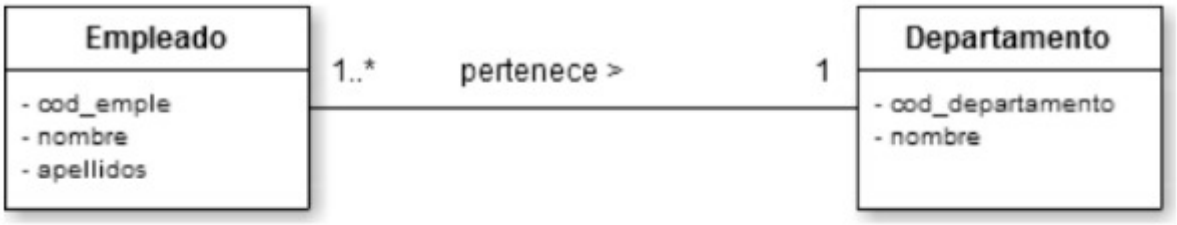


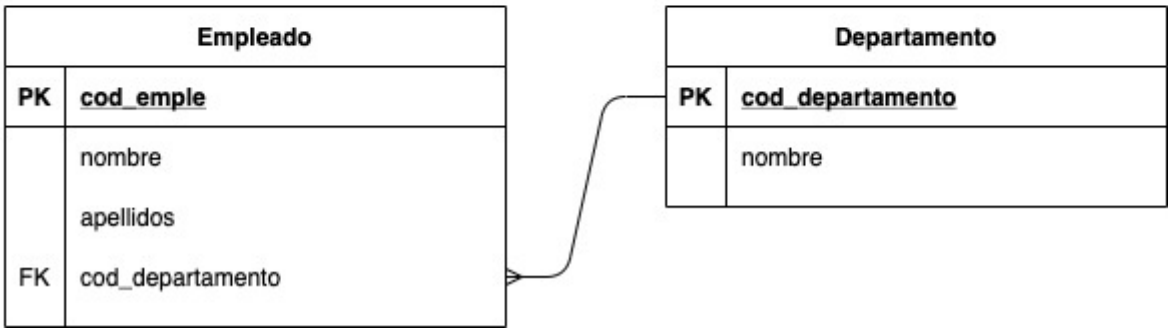
> ¡RECUERDA! Para cumplir con la cardinalidad impuesta por la asociación en el esquema relacional, la clave primaria de la tabla resultante serán todos los atributos que formen las claves primarias de ambas tablas.

6.2.2 Traslado de asociaciones una a muchos

Dos tipos de entidades A y B tienen entre ellas una asociación con cardinalidad uno a muchos cuando una entidad del tipo A se puede relacionar con muchas entidades del tipo B, pero una entidad del tipo B solo puede relacionarse con una del tipo A.

Como norma general, una asociación 1-N no produce una nueva tabla. Se produce, en cambio, una propagación de la clave primaria de la tabla con cardinalidad 1 hacia la tabla con cardinalidad N.

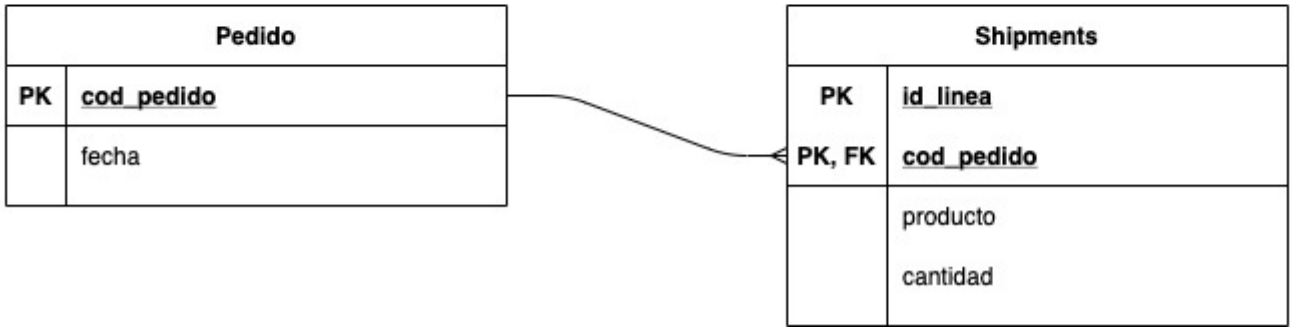




6.2.3 Traslado de asociaciones de composición

Las asociaciones de composición suelen aparecer casi siempre con multiplicidad 1-N, por lo que son un subcaso de la asociación anterior.

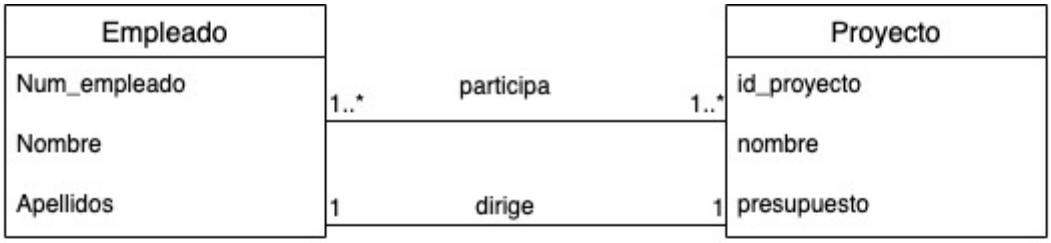
Cuando la asociación entre los tipos de entidades sea de composición (es decir, fuerte/débil), los atributos que se propagan forman parte, además, de la PK de la tabla de destino.

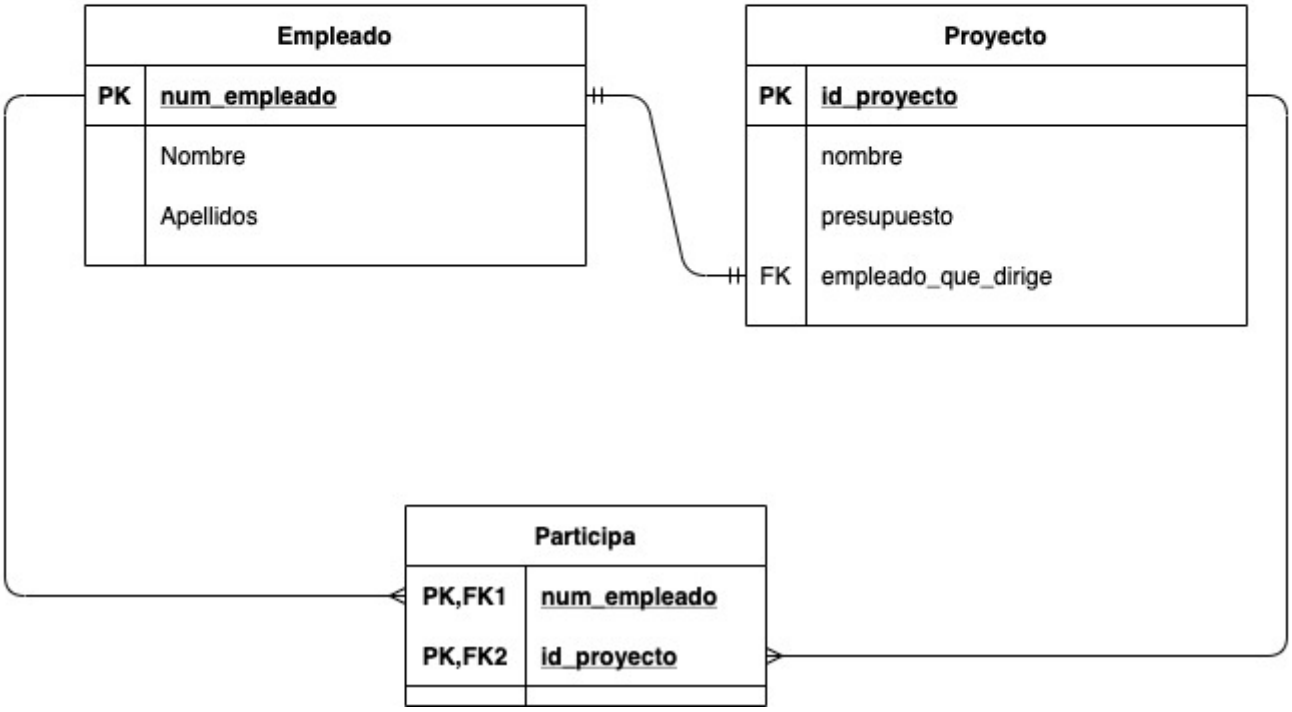


6.2.4 Traslado de asociaciones uno a uno

Estas asociaciones se pueden estudiar como un caso particular de las asociaciones 1-N.

Como normal general, una asociación 1 a 1 producirá la propagación de los atributos de la clave primaria de una tabla a otra. En principio, el orden de dónde hacia dónde propagar no es importante, aunque en ocasiones **la semantica del problema nos puede ayudar a resolver esta elección**. Normalmente, ***este atributo no suele formar parte de la clave primaria de la tabla de destino***.



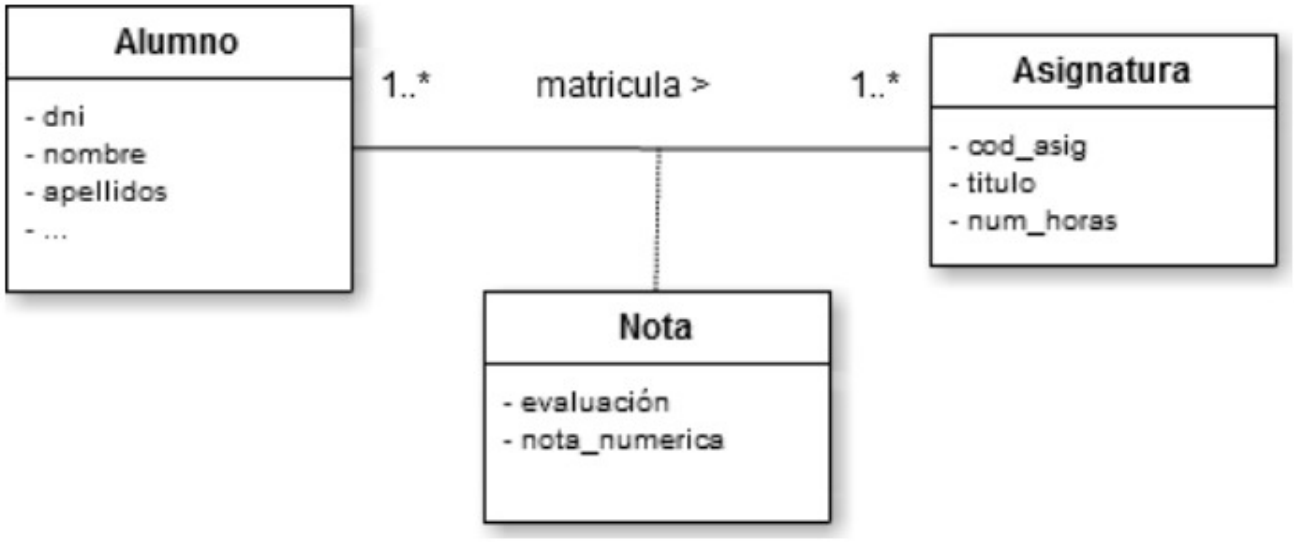


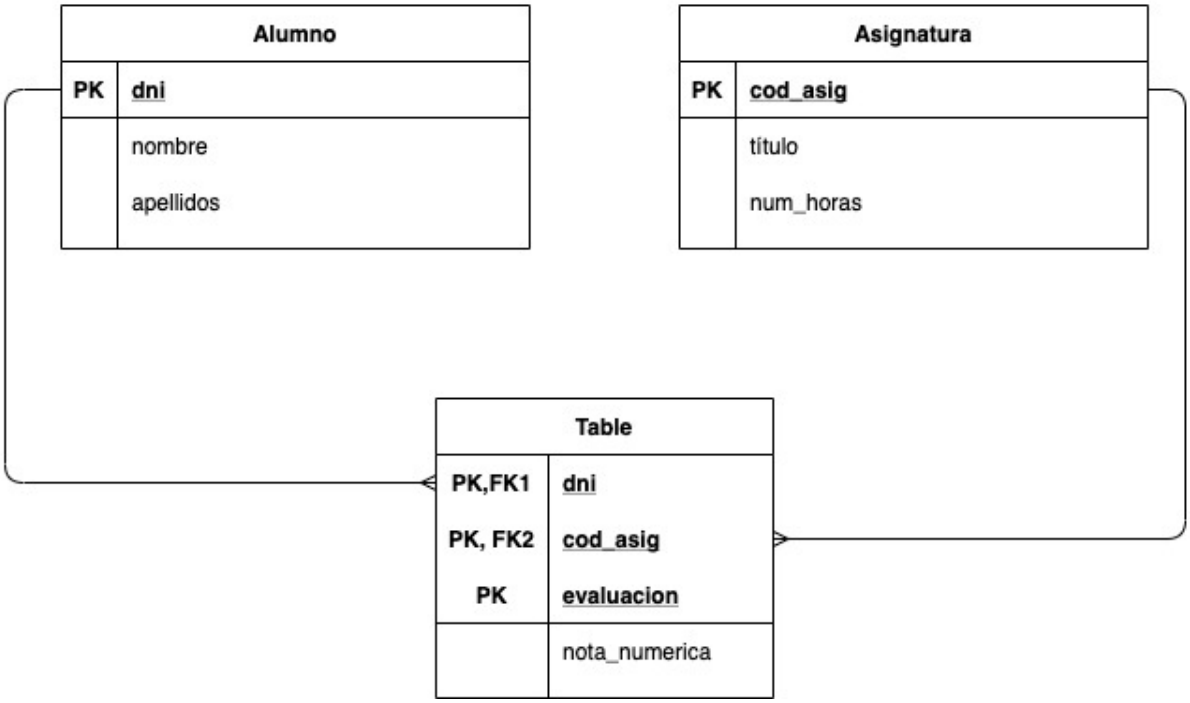
6.2.5 Traslado de una asociación con clase de asociación

Hemos aprendido en entornos de desarrollo que en UML, la forma de representar aquellos atributos que no pertenecen a una clase o a otra, sino a la asociación en sí, se pueden representar mediante una clase de asociación.

Por lo general, una clase de asociación en el diagrama de clases se trasladará a una nueva tabla en el esquema relacional (independientemente de la cardinalidad de la asociación). Sus atributos serán las claves primarias de las clases asociadas, así como todas las propiedades de la clase de asociación.

> Existe la posibilidad de que algún atributo de la clase de asociación forme parte de la PK de la tabla resultante (dependerá del problema a resolver)





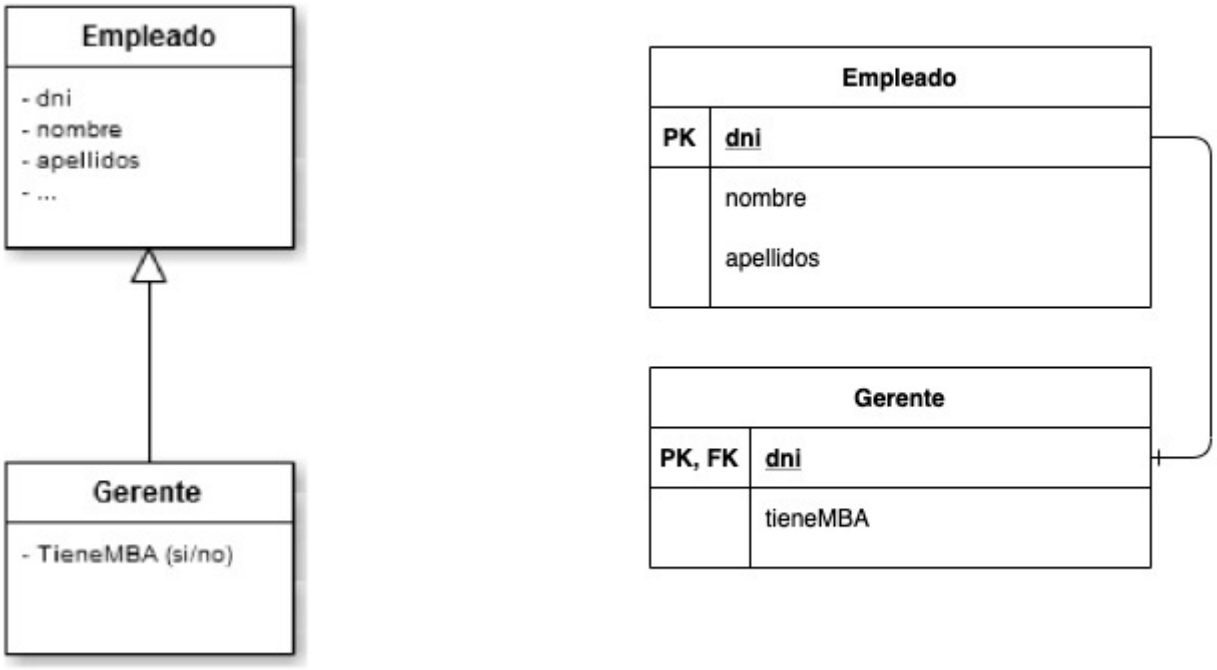
6.3 Traslado de interrelaciones es-un (herencia)

Las relaciones es-un entre entidades (representadas en UML como asociaciones de herencia) tienen un tratamiento especial. De hecho, es tan especial **que dependerá de si la herencia presenta interrelaciones con otras entidades** para poder establecer unas reglas de traslado.

En los siguientes apartados vamos a tratar de encontrar algunas pautas a partir de diferentes situaciones.

Como norma general, **tendremos una tabla por cada una de las entidades de la jerarquía de herencia**. Los atributos de la tabla asociada a la entidad *base* serán los de la propia entidad. Los atributos de la tabla

asociada a la entidad *hija*, serán **la clave primaria de la tabla asociada a la entidad base más los atributos propios**.



> **Hay que tener en cuenta que, con esta solución, para insertar un Gerente hay que insertar una fila en Empleado y otra en Gerente.**

Esta solución es eficiente cuando el número de gerentes sea mucho menor que el número de empleados. Si se diera el caso de que el número previsible de filas en cada tabla es aproximadamente el mismo, posiblemente nos interese otra solución, como trasladarlo todo a una tabla, añadiendo algún atributo booleano que nos permita diferenciar quién sí es gerente y quién no. Esto nos ahorraría realizar el JOIN entre Empleado y Gerente.

6.3.1 Traslado de interrelaciones es-un con un número filas previsible equilibrado

Si se nos presenta el caso de que la previsión de filas para las tablas asociadas a la entidad base es aproximadamente igual, la solución general anterior no sería muy eficiente, ya que tendríamos que estar realizando, constantemente, operaciones JOIN (que penalizan el rendimiento de las consultas) para poder obtener datos.

En el siguiente ejemplo (**¡ojo! si hubiera asociaciones en las entidades base o hijas, posiblemente la solución sería otra**), el número previsible de filas para las tres entidades está equilibrado. En este caso, una solución más eficiente sería:

- Trasladar la entidad base a una nueva tabla
- Trasladar cada entidad hija a una nueva tabla. Sus atributos serían la clave primaria de la tabla base, más los atributos de la clase base, más los atributos propios.
- En las tablas *hijas*, la clave primaria **no sería clave externa** de la tabla base.

Cliente	
PK	dni
	nombre
	apellidos

ClienteMayorista	
PK	dni
	nombre
	apellidos
	razon_social

ClienteVip	
PK	dni
	nombre
	apellidos
	razon_social

Esta solución se suele conocer como *table per subclass* y nos ahorra muchos JOINS. Sin embargo, si en algún momento queremos hacer una consulta sobre todos los tipos a la vez, tendríamos que utilizar el operador UNION.

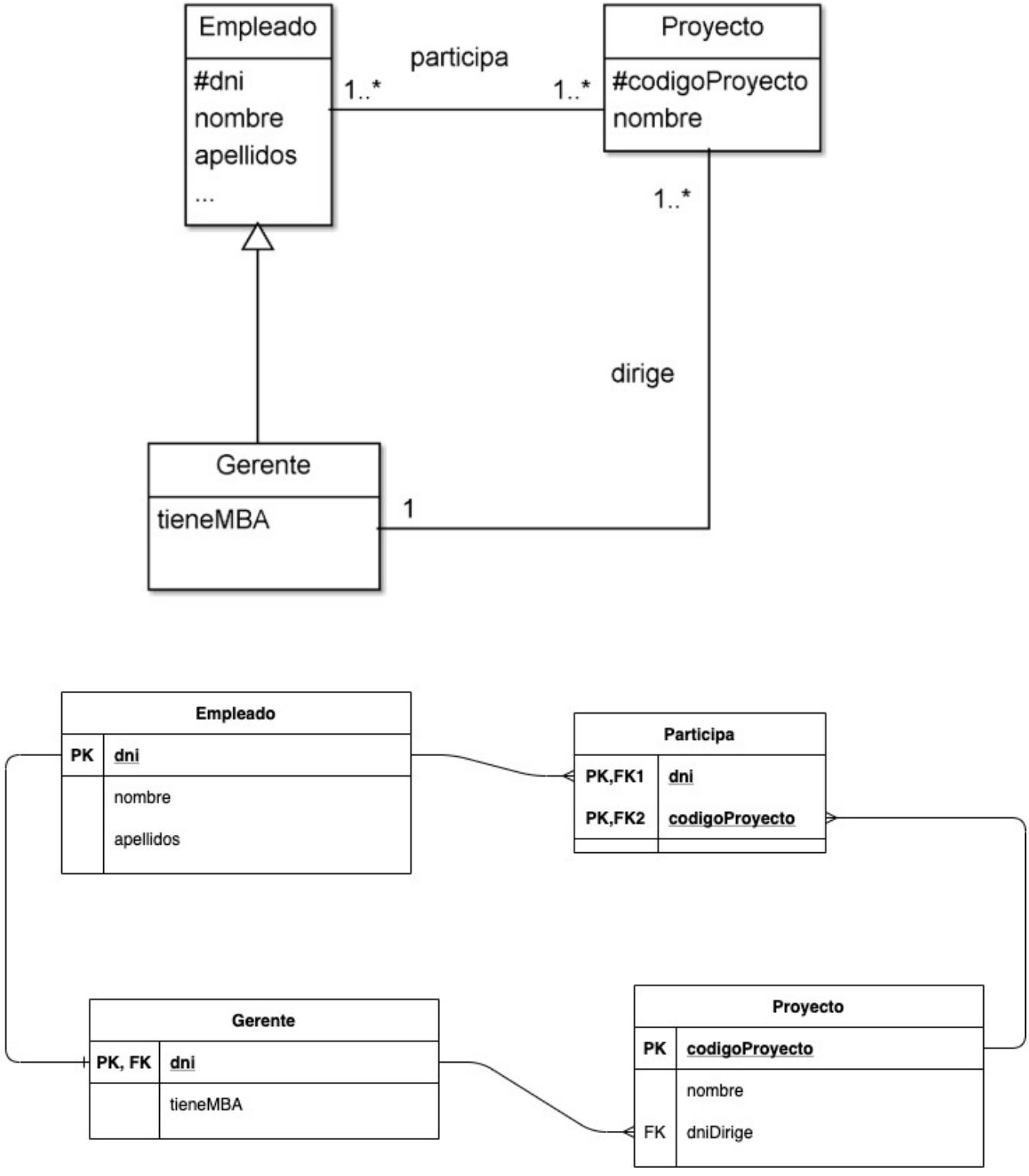
6.3.2 Traslado de interrelaciones es-un con otras asociaciones

En el caso de que las entidades que participan de una herencia (relación es-un) tengan otras interrelaciones (asociaciones) con otras entidades, la forma de trasladar al grafo relacional puede cambiar sustancialmente.

6.3.2.1 Herencia básica con interrelación en la entidad *base*

Supongamos una jerarquía de herencia en la que la entidad base tiene una interrelación con otra entidad. ¿La forma de trasladar sería igual?

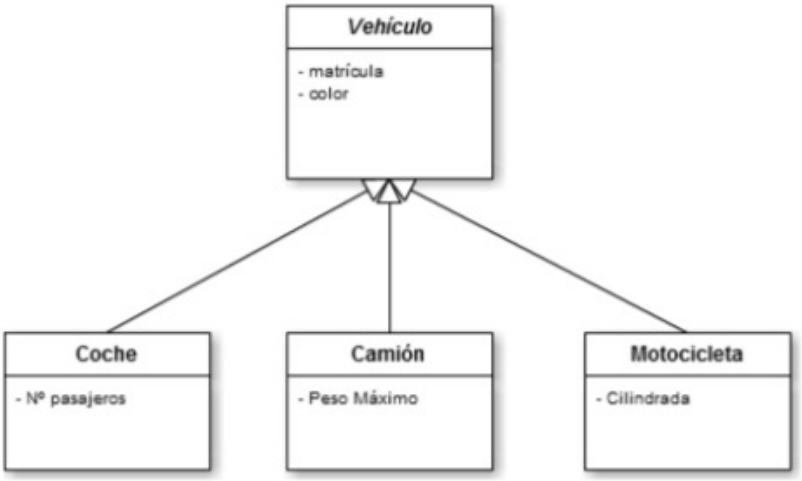
En casos como el de este ejemplo, si seguimos las normas que hemos dictado en apartados anteriores, posiblemente obtengamos un buen diseño.



En este caso, como vemos, si seguimos las normas que hemos trabajado en apartados anteriores obtenemos un buen diseño.

6.3.2.2 Herencia con clase base abstracta

Una clase abstracta se define como una clase de la cuál no vamos a necesitar instancias. Si en nuestro modelo la utilizamos, es porque no tenemos previsión de que vayamos a necesitar guardar información directamente de ello, y su existencia se justifica para aglutinar información (y en orientación a objetos comportamiento) común a todas sus derivadas.

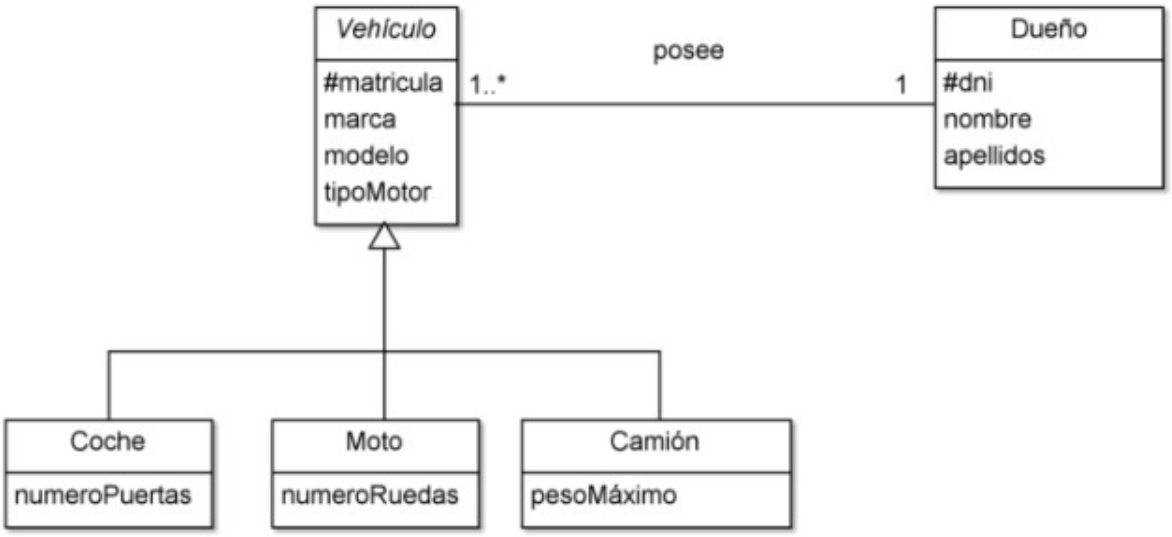


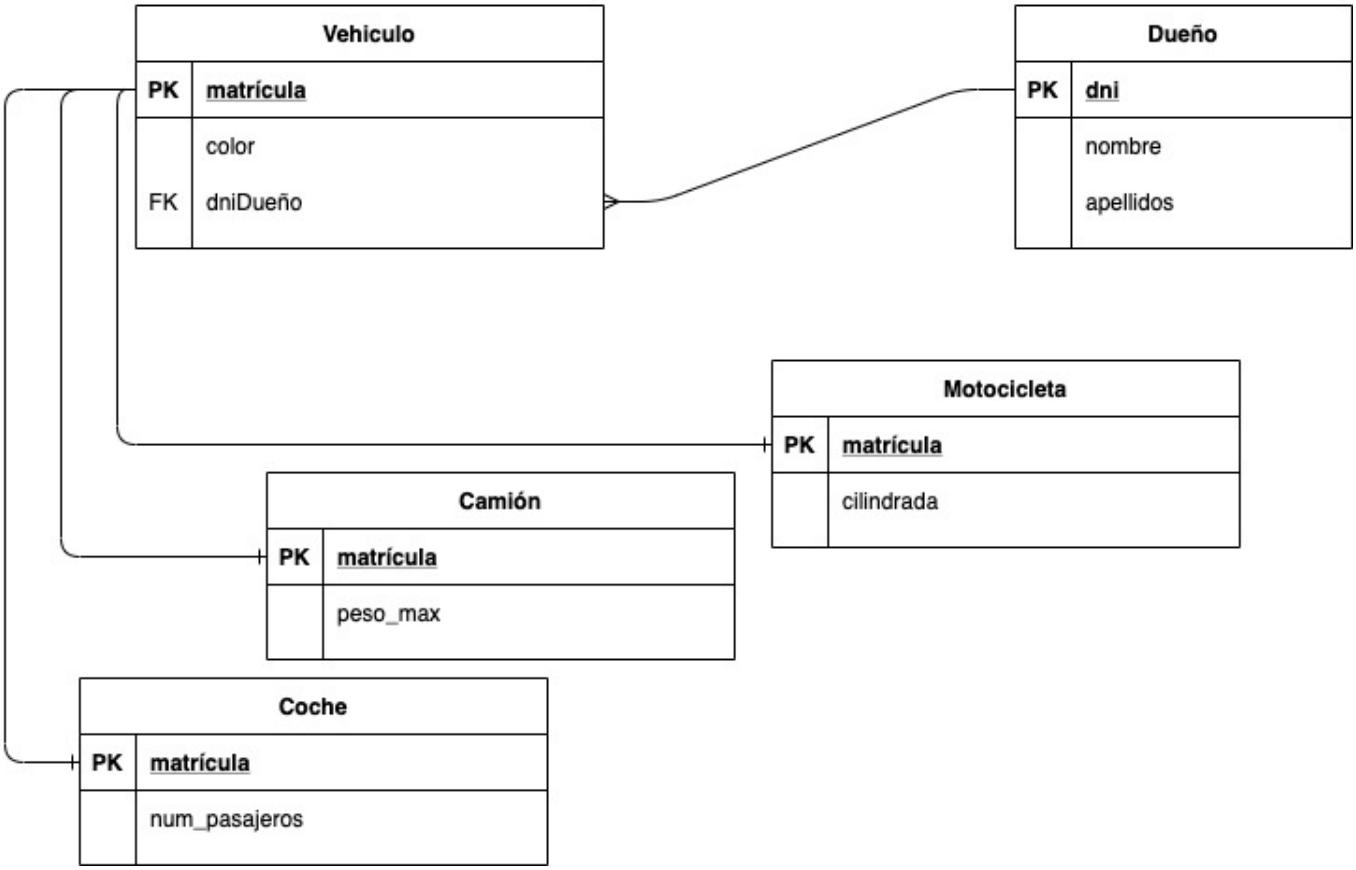
Coche	
PK	matrícula
	color
	num_pasajeros

Camión	
PK	matrícula
	color
	peso_max

Motocicleta	
PK	matrícula
	color
	cilindrada

Pero, **¿qué pasa si la entidad representada por la clase base tiene algún tipo de interrelación con otra entidad?** La solución propuesta en el grafo relacional anterior no nos serviría



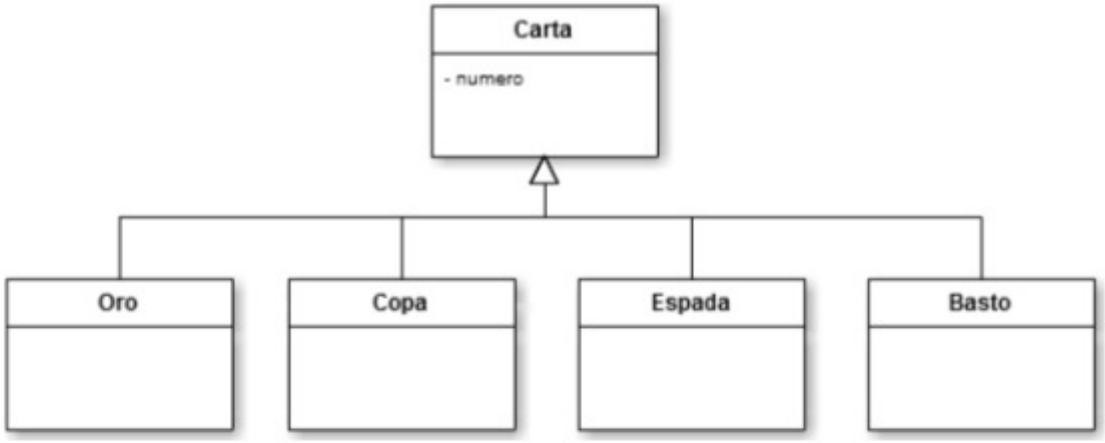


En este caso, **sería necesario trasladar la clase abstracta como una tabla**, para poder gestionar la asociación con la clase Dueño; y por tanto, las clases *hijas* se trasladarían creando una nueva tabla, con la clave primaria de la tabla asociada a la clase base y los atributos propios. Así mismo, la clave primaria de estas sería también clave externa.

6.3.2.3 Herencia con clases hijas sin atributos

Este es un caso típico en el que las clases hijas solamente nos sirven para diferenciar tipos de instancias de la clase base, pero que no aportan ningún atributo nuevo.

En este caso, podríamos plantear algunas diferentes soluciones:



Solución 1

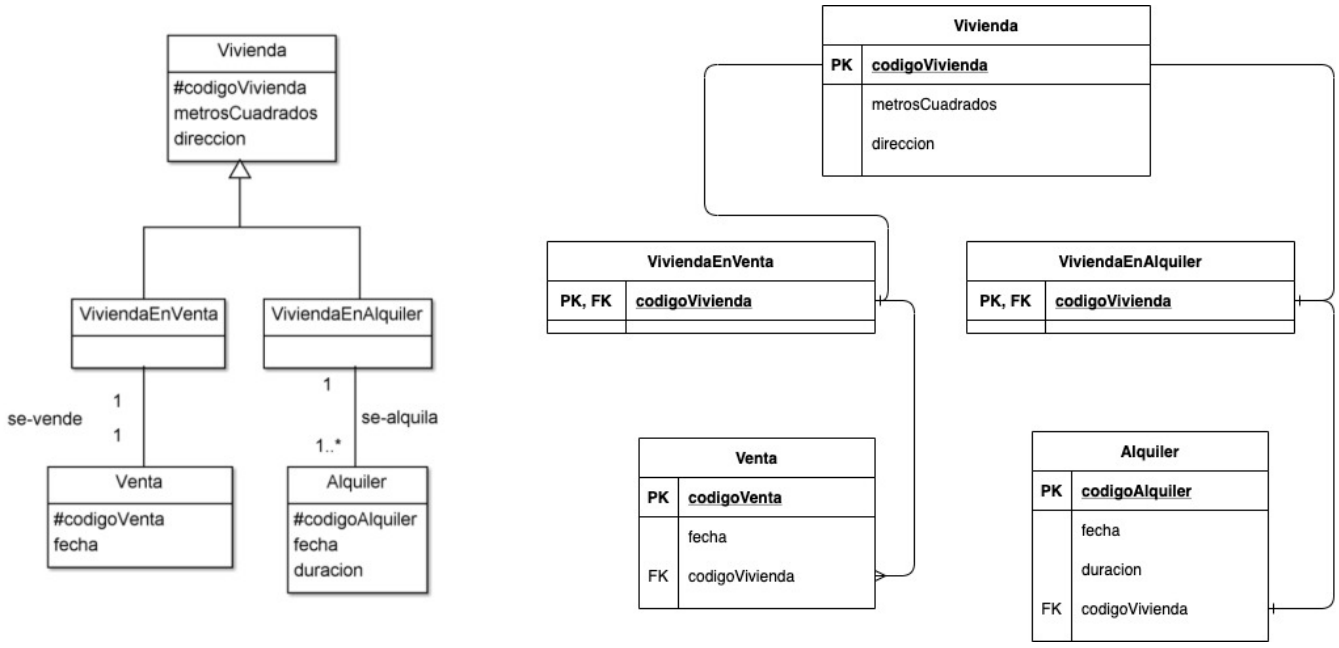
Carta	
PK	numero
	palo VARCHAR CHECK (palo IN ('Oro', 'Copa', 'Espada', 'Basto'))

Solución 2

TipoCarta	
PK	id_tipo
	nombre

Carta	
PK	numero
FK1	id_tipo

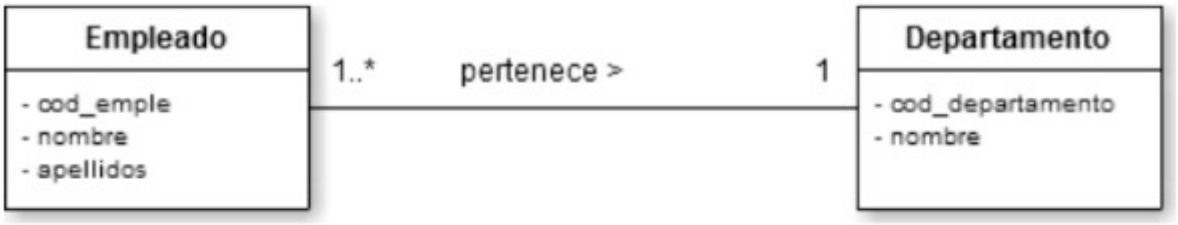
Pero, **¿qué sucedería si las clases hijas, no teniendo atributos, sí que tienen asociaciones?**. En este caso, las soluciones propuestas anteriormente no nos servirían y tendríamos que proponer una alternativa en la que **sí trasladásemos las clases hijas como tablas**.

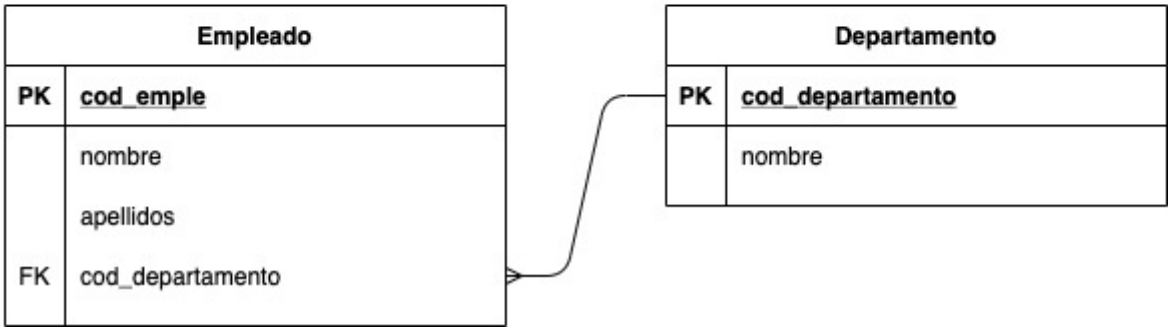


6.4 Nota aclaratoria sobre el tipo de conectores a usar en los diagramas

El uso de una notación pensada para diagramas entidad/relación a la hora de representar un diagrama *relacional* puede provocar algo de controversia, sobre todo en relación al tipo de conectores que escoger en los extremos de las asociaciones. Vamos a tratar de zanjar la cuestión indicando que **siempre representaremos la cardinalidad máxima de las asociaciones**, y siguiendo las siguientes pautas:

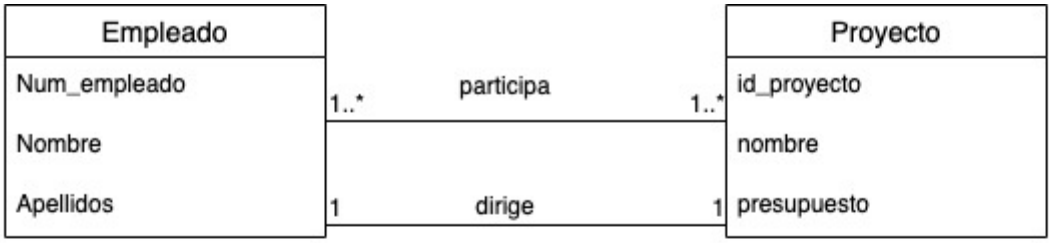
1. **Como norma general**, las claves externas resultantes de trasladar la clave primaria de una tabla a otra tabla, y que representen una interrelación entre entidades se conectará con un símbolo de pata de gallo en el extremo de la clave externa, y sin símbolo en el extremo de la clave primaria. Esto aplica al traslado de asociaciones muchos a muchos, uno a muchos, composición... con *la salvedad de las asociaciones uno a uno*.

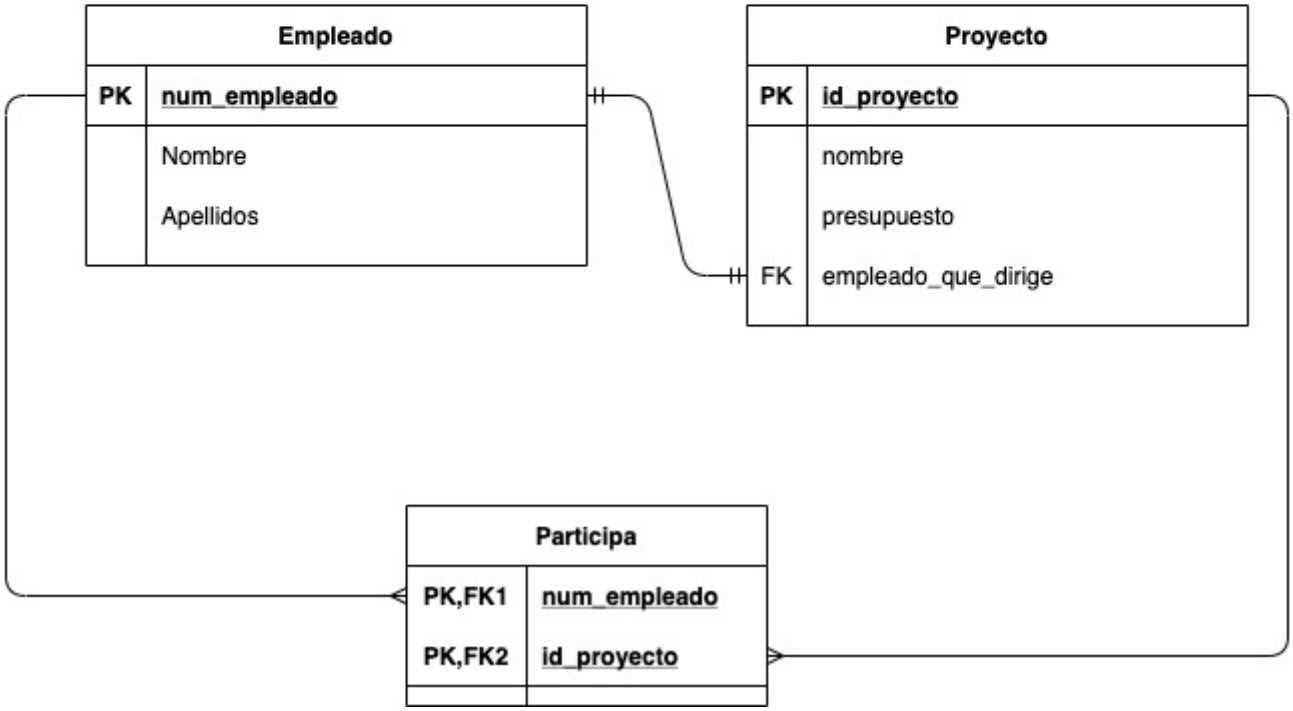




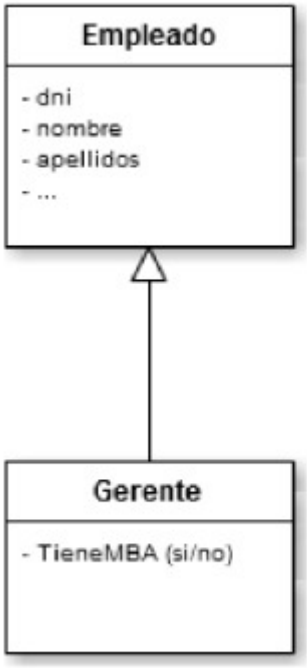
2. Para el caso del traslado de **asociaciones uno a uno**, la clave externa que represente esta asociación se conectará con un símbolo de cruce de línea vertical sobre el extremo de la asociación conectado a

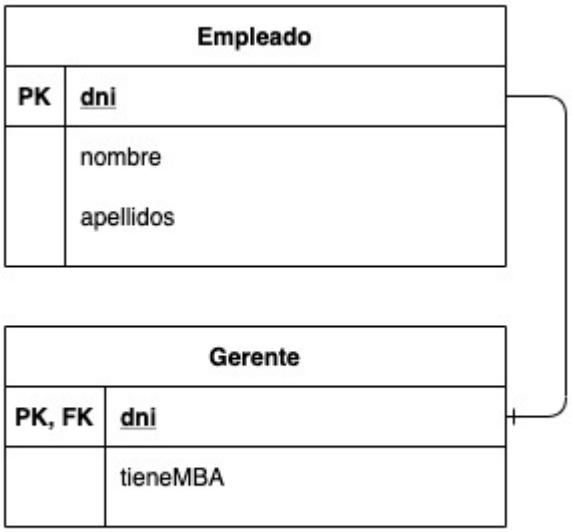
dicha clave externa. En el otro extremo (clave primaria), podremos usar o bien el mismo símbolo, o no usar ninguno.





3. En el caso de una **herencia de clases** (jerarquía de entidades), cuando tengamos como resultado de trasladar una clase hija una tabla con la clave primaria de la clase base, siendo esta clave primaria en la hija, y también clave externa hacia la tabla de la clase base, se conectará con un símbolo de cruce de línea vertical sobre el extremo de la asociación en la tabla asociada a la clase hija, y sin extremo en la tabla de la clase base.





Bibliografía

1. https://www.campusmvp.es/recursos/post/Disenando-una-base-de-datos-en-el-modelo-relacional.aspx
2. https://apuntesfpinformatica.es/bases-de-datos/