Unidad 2: Interpretación de diagramas Entidad/Relación

Índice de contenidos

1 Etapas para el diseño de una base de datos.

Lo primero que se debe **tener muy bien documentado** son **los requerimientos**, se debe **conocer muy bien cuál es el problema a resolver**. Generalmente, los requerimientos bien documentados es todo lo que **se necesita para comenzar a diseñar la *BD***.

El **diseño de una *BD* no es un proceso sencillo**. Habitualmente, la **complejidad de la información y la cantidad de requisitos de los sistemas de información** hacen que sea complicado. Por este motivo, cuando se diseñan *BD* es interesante **aplicar la vieja estrategia de dividir para vencer**.

Por lo tanto, **conviene descomponer el proceso del diseño en varias etapas**; en cada una se obtiene un resultado intermedio que sirve de punto de partida de la etapa siguiente, y en la última etapa se obtiene el resultado deseado. De este modo no hace falta resolver de golpe toda la problemática que plantea el diseño, sino que en cada etapa se afronta un solo tipo de subproblema. Así se divide el problema y, al mismo tiempo, se simplifica el proceso.

Se descompondrá el **diseño de *BD*** en tres etapas o **fases**:

1.1 Diseño conceptual.

En esta etapa **se obtiene una estructura de la información de la futura *BD* independiente de la tecnología que hay que emplear**. No se tiene en cuenta todavía qué **tipo de *BD*** se utilizará –relacional, orientada a objetos, jerárquica, etc.–, en consecuencia, tampoco se tiene en cuenta con qué ***SGBD*** ni con qué lenguaje concreto se implementará la *BD*. Así pues, la etapa del diseño conceptual **permite concentrarse únicamente en la problemática de la estructuración de la información**, sin tener que preocuparse al mismo tiempo de resolver cuestiones tecnológicas.

El **resultado de la etapa del diseño conceptual se expresa mediante algún modelo de datos de alto nivel**. Uno de los **más empleados es el modelo *Entidad/Relación*** (***E/R***).

Para la realización de esquemas que ofrezcan una visión global de los datos, Peter Chen en 1976 y 1977 presenta dos artículos en los que se describe el modelo Entidad/Relación (E/R). Con el paso del tiempo, este modelo ha sufrido modificaciones y mejoras. Actualmente, el modelo Entidad/Relación Extendido (ERE) es el más aceptado, aunque existen variaciones que hacen que este modelo no sea totalmente un estándar.

Durante esta fase, **se plasmarán las entidades** (cada entidad se identificará con un rectángulo y dentro de este se colocará su nombre). A **cada entidad se le colocarán sus respectivos atributos** (elipse) **y se resaltará el atributo principal**, aquel atributo que identificará cada registro de manera única.

Por último se crearán las relaciones (rombos) **que existen entre dichas entidades**.

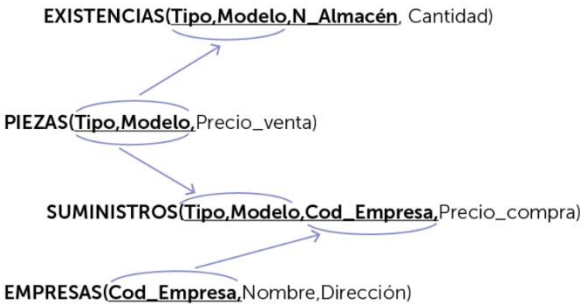
La forma de elaborar un diseño conceptual expresado con el modelo *E/R* se explica en los puntos 2, 3 y 4.

1.2 Diseño lógico.

En esta etapa **se parte del resultado del diseño conceptual**, que **se transformará de forma que se adapte a la tecnología que se debe emplear**. Más concretamente, es preciso que se ajuste al **modelo del *SGBD* con el que se desea implementar la *BD***. Por ejemplo, si se trata de un *SGBD* relacional, esta etapa obtendrá un conjunto de relaciones/tablas con sus atributos, claves primarias y claves foráneas.

Esta etapa parte del hecho de que ya se ha resuelto la problemática de la estructuración de la información en un

ámbito conceptual, y **permite concentrarse en las cuestiones tecnológicas relacionadas con el modelo de *BD***.



Más adelante, en los siguientes puntos de la unidad, se explicará cómo se hace el **diseño lógico de una *BD* relacional**, tomando como punto de partida un diseño conceptual expresado con el modelo *E/R*; es decir, **se verá cómo se puede transformar un modelo *E/R* en un modelo relacional**.

En esta fase, en el caso del modelo de *BD* relacional, además **se debe pensar en cómo normalizar las tablas o relaciones para evitar duplicidad de información y para ahorrar espacio de almacenamiento**. Esto último (ahorrar espacio) ya no es tan importante como hace algunos años, incluso hoy en día se habla de inteligencia de negocios, minería de datos, entre otros términos que exigen eliminar la normalización.

En esta fase, **en el caso de una *BD* relacional, no es necesario preocuparse por el motor de *BD***, ya que el modelo tabular (basado en tablas) es igual en todos los motores de *BD* relacionales.

El diseño lógico de una *BD* relacional se explica en el punto 5 y la normalización en el punto 6.

1.3 Diseño físico.

En esta etapa **se transforma la estructura obtenida en la etapa del diseño lógico, con el objetivo de conseguir una mayor eficiencia; además, se completa con aspectos de implementación física que dependerán del *SGBD***.

Por ejemplo, si se trata de una *BD* relacional, la transformación de la estructura puede consistir en lo siguiente: tener almacenada alguna relación que sea la combinación de varias relaciones que se han obtenido en la etapa del diseño lógico, partir una relación en varias, añadir algún atributo calculable a una relación, etc. Los aspectos de implementación física que hay que completar consisten normalmente en la elección de estructuras físicas de implementación de las relaciones, la selección del tamaño de las memorias intermedias (buffers) o de las páginas, etc.

Tabla: EMPLEADO			
NOMBRE DEL CAMPO	TIPO	ANCHURA	COMENTARIO
Clave_Empleado	Varchar	Entero	Clave del empleado
Nombre	Varchar	30	Nombre del empleado
Domicilio	Varchar	30	Dirección del empleado
Telefono	Varchar	Entero	Teléfono del empleado
Puesto	Varchar	20	Puesto que tiene el empleado
FechaAlta	Fecha	10	Fecha que ingreso a la empresa
FechaNac	Fecha	10	Fecha de nacimiento del empleado

En la etapa del diseño físico –con el objetivo de conseguir un buen rendimiento de la *BD*–, **se deben tener en cuenta las características de los procesos que consultan y actualizan la *BD***, como por ejemplo los caminos de acceso que utilizan y las frecuencias de ejecución. También es necesario **considerar los volúmenes que se espera tener de los diferentes datos que se quieren almacenar**.

En esta unidad no se abordará el diseño físico de la *BD*, se tratará en la unidad 3.

2 Modelo *Entidad/Relación*.

2.1 ¿Qué es el modelo *Entidad/Relación*?

Es una herramienta de referencia para la representación conceptual de problemas del mundo real. Su **objetivo** principal es **facilitar el diseño de *BD* permitiendo la especificación de un esquema que representa la estructura lógica completa de una *BD***. 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* **está basado en una percepción consistente en objetos básicos llamados entidades y relaciones entre estos objetos**, estos y otros conceptos se desarrollan a continuación.

2.2 Entidades.

Una **entidad** puede ser un **objeto físico, un concepto o cualquier elemento que se quiera 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, se puede 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.*

Suponer que se tiene que desarrollar el esquema conceptual para una *BD* 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, se ha de pensar en nombres que tengan especial importancia dentro del lenguaje propio de la organización o sistema que vaya a utilizar dicha *BD***. 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 se establece un símil con la *Programación Orientada a Objetos*, se puede 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.

En el modelo *Entidad/Relación*, **la representación gráfica de las entidades se realiza mediante el nombre de la entidad encerrado dentro de un rectángulo**.

2.2.1 Entidades fuertes y débiles.

Las entidades **pueden ser clasificadas en dos grupos**:

a) 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 *BD* 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 con el nombre de la entidad encerrado dentro de un rectángulo**.

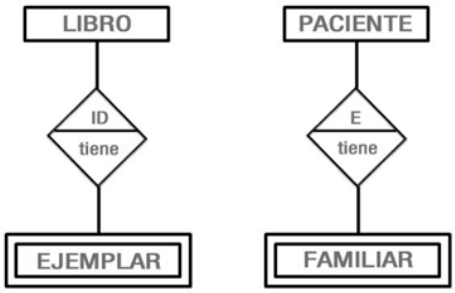
b) Entidades débiles.

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**.

En el modelo *E/R* una entidad débil **se representa con el nombre de la entidad encerrado en un rectángulo doble**.

Las entidades débiles presentan dos **tipos de dependencia**:

- ✓ **Dependencia en existencia**: entre entidades, **si desaparece una instancia de entidad fuerte desaparecerán las instancias de entidad débiles que dependan de la primera** (Ej.: *AUTOR* ← *LIBRO*, *CLIENTE* ← *PEDIDO*). La **representación** de este tipo de dependencia **incluirá una *E* en el interior de la relación débil**.
- ✓ **Dependencia en identificación**: **debe darse una dependencia en existencia y, además, una ocurrencia de la entidad débil no puede identificarse por sí misma, debiendo hacerse mediante la clave de la entidad fuerte asociada**. La **representación** de este tipo de dependencia **incluirá un *ID* en el interior de la relación débil** (con *DIA* se utiliza un doble rombo).



Una entidad débil es una entidad cuya existencia depende de la existencia de otra entidad. Una entidad fuerte es una entidad que no es débil (no depende de la existencia de otra entidad).

Tanto las entidades fuertes como las débiles las vamos a nombrar con sustantivos en singular y mayúsculas (aunque se pueden encontrar diagramas *E/R* en los que se nombran con sustantivos en plural y minúsculas, incluso mezcla de ambas convenciones). Nota: hay algunos conceptos que se desarrollarán en los siguientes puntos (relación, atributo y clave) y que se están utilizando para describir los tipos de dependencias (no debes preocuparte por ahora).

Ejercicio resuelto

Identifica cuál de las siguientes entidades no podría ser considerada como entidad débil (es entidad fuerte):

- ✓ ***PROVEEDOR* (perteneciente a una *BD* de gestión de stocks).**
- ✓ ***PAGO* (perteneciente a una *BD* bancaria).**
- ✓ ***FAMILIAR* (perteneciente a una *BD* hospitalaria).**

Solución: *PROVEEDOR* → Esta entidad puede existir por sí misma sin depender de otras ocurrencias de entidad. Además, posee propiedades o atributos propios que la identifican frente a otras ocurrencias de la misma entidad.

PAGO no podría ser ya que no puede darse sin la existencia de otra entidad como podría ser *PRÉSTAMO*. Si desaparece el préstamo desaparecen también los pagos.

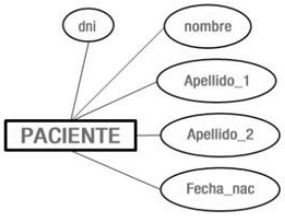
FAMILIAR no podría ser ya que en una *BD* hospitalaria, esta entidad podría depender de otra que sería *PACIENTE*. Si se estuviesen gestionando las visitas, cada familiar estaría asociado a un determinado paciente.

2.3 Atributos.

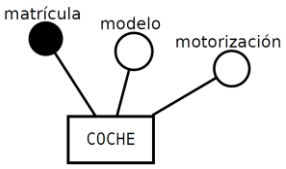
Los atributos **describen características o propiedades que posee cada miembro de un conjunto de entidades**. El mismo atributo establecido para un conjunto de entidades o, lo que es lo mismo, para un tipo de entidad, almacenará información parecida para cada ocurrencia de entidad. Pero, cada ocurrencia 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. y estos dependerán de la información que sea necesaria almacenar.



En el modelo *E/R* los atributos de una entidad **son representados mediante el nombre del atributo rodeado por una elipse**. La **elipse se conecta con la entidad mediante una línea recta**. Cada atributo **debe tener un nombre único** que haga referencia al contenido de dicho atributo. **Los nombres de los atributos se escriben en minúscula**. También se pueden utilizar círculos para representar los atributos, añadiendo junto a ellos el nombre del atributo.



Al **conjunto de valores permitidos para un atributo** se le denomina **dominio**. Todos los posibles valores que puede tomar un atributo deberán estar dentro del dominio. Varios atributos pueden estar definidos dentro del mismo dominio. Por ejemplo, los atributos nombre, apellido primero y apellido segundo de la entidad *PACIENTE*, están definidos dentro del dominio de cadenas de caracteres de una determinada longitud.

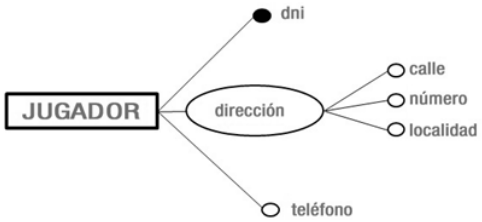
Aunque los dominios suelen ser amplios (números enteros, reales, cadenas de caracteres, etc.), a la hora de llevar a cabo el desarrollo de una *BD*, es mejor establecer unos límites adecuados para que el *SGBD* lleve a cabo las verificaciones oportunas en los datos que se almacenen, garantizando así la integridad de éstos.

2.3.1 Tipos de atributos.

Existen varias características que hacen que los atributos asociados a una entidad o relación sean diferentes, **los clasificaremos según varios criterios**:

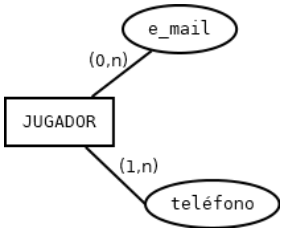
a) Atributos atómicos o compuestos.

- **Atributo simple o atómico**: es un ***atributo que no puede dividirse en otras partes o atributos***, presenta un único elemento. No es posible extraer de este atributo partes más pequeñas que puedan tener significado. Un ejemplo de este tipo de atributos podría ser el atributo *dni* de la entidad *JUGADOR*.
- **Atributo compuesto**: son ***atributos que pueden ser divididos en subpartes***, éstas constituirán otros atributos con significado propio. Por ejemplo, la dirección del jugador podría considerarse como un atributo compuesto por la calle, el número y la localidad (ver imagen).



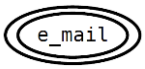
b) Atributos monovaluados o multivaluados.

- **Atributo monovaluado**: es ***aquél que tiene un único valor para cada ocurrencia de entidad***. Un ejemplo de este tipo de atributos es el *dni*.
- **Atributo multivaluado**: es ***aquél que puede tomar diferentes valores para cada ocurrencia de entidad***. Por ejemplo, la dirección de *e-mail* de un empleado podría tomar varios valores para alguien que posea varias cuentas de correo. **En este tipo de atributos hay que tener en cuenta** los siguientes conceptos:
  - ✓ La **cardinalidad de un atributo** indica el ***número mínimo y el número máximo de valores que puede tomar*** para cada ejemplar de la entidad o relación a la que pertenece.
  - ✓ La **cardinalidad mínima** indica la ***cantidad de valores del atributo que debe existir para que la entidad sea válida***. Este número casi siempre es **0** o **1**. Si es 0, el atributo podría no contener ningún valor y si es 1, el atributo debe tener un valor.
  - ✓ La **cardinalidad máxima** indica la ***cantidad máxima de valores del atributo que puede tener la entidad***. Por lo general es **1** o **n**. Si es 1, el atributo no puede tener más que un valor, si es n, el atributo puede tener múltiples valores y no se especifica la cantidad absoluta.



El atributo *e_mail* de la imagen, puede ser opcional y no contener ningún valor, o bien, almacenar varias cuentas de correo electrónico de un jugador. Como se ve, la cardinalidad representada en la imagen es *(0,n)*.

En algunos diagramas *E/R* un atributo multivaluado se puede encontrar representado con una **elipse doble**.



c) **Atributos obligatorios u opcionales.**

- **Atributo obligatorio**: es ***aquél que ha de estar siempre definido para una entidad o relación***. Por ejemplo, para la entidad *JUGADOR* será necesario tener algún atributo que identifique cada ocurrencia de entidad, podría ser su *DNI*. **Una clave o llave es un atributo obligatorio**.

- **Atributo opcional**: es ***aquél que podría ser definido o no para la entidad***. Es decir, puede haber ocurrencias de entidad para las que ese atributo no esté definido o no tenga valor. Para marcarlo se puede indicar la **cardinalidad del atributo como *(0,1)***.

d) **Atributos derivados o almacenados**: el ***valor de este tipo de atributos puede ser obtenido del valor o valores de otros atributos relacionados***. Un ejemplo clásico de atributo derivado es la edad, si se ha almacenado en algún atributo la fecha de nacimiento, la edad es un valor calculable a partir de dicha fecha. Se representa mediante una **elipse con trazo discontinuo**.

2.3.2 Claves.

Está claro que **es necesario identificar correctamente cada ocurrencia de entidad o relación**, de este modo el tratamiento de la información que se almacena podrá realizarse adecuadamente. Esta distinción **podría llevarse a cabo tomando todos los valores de todos los atributos de una entidad o relación**. Pero, en la mayoría de las ocasiones **no es necesario utilizar todos, bastando con un subconjunto de ellos**. Aunque puede ocurrir que ese subconjunto tenga idénticos valores para varias entidades, por lo que cualquier subconjunto no será válido.

Por tanto, los valores de los atributos de una entidad deben ser tales que permitan **identificar unívocamente** a la entidad. En otras palabras, **no se permite que ningún par de entidades tengan exactamente los mismos valores de todos sus atributos**. Teniendo en cuenta esto, prestar atención a los siguientes conceptos:

***Identificador o superclave**: cualquier conjunto de atributos que permite identificar de forma única a cada entidad.*

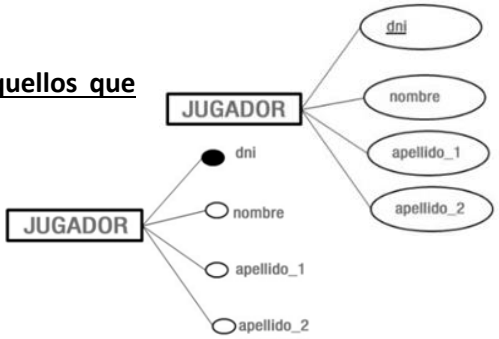
***Clave candidata**: cada una de las superclaves formadas por el mínimo número de atributos posibles.*

***Clave primaria o principal**: es la clave candidata seleccionada por el diseñador de la BD.*

***Claves alternativas**: son el resto de claves candidatas que no han sido escogidas como clave primaria.*

La **representación de las claves primarias** puede realizarse de dos formas:

- ✓ Si se utilizan elipses para representar los atributos, **se subrayarán aquellos que formen la clave primaria**.
- ✓ Si se utilizan círculos para representar los atributos, **se utilizará un círculo negro en aquellos que formen la clave primaria**.



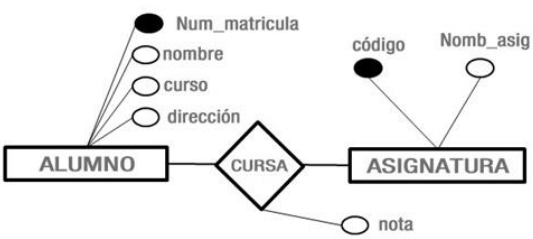
Las **claves alternativas** se pueden representar **subrayándolas con trazo discontinuo** o si se utilizan **círculos dejando la mitad en blanco** ◑.

Ejercicio resuelto

Dada la entidad *TRABAJADOR*, con los atributos *nombre*, *apellido_1*, *apellido_2*, *dni*, *numero_afiliacion_ss*, *fecha_nacimiento* y *código_empresa*. ¿Los atributos *nombre*, *apellido_1* y *apellido_2* podrían formar una clave candidata?

Solución: No, si se tiene en cuenta que puede haber varios trabajadores con el mismo nombre y apellidos. Los atributos *dni* y *numero_afiliacion_ss*, serían dos claves candidatas adecuadas. Si se escoge *dni* como clave primaria, *numero_afiliacion_ss* quedaría como clave alternativa.

2.3.3 Atributos de una relación.



Una relación puede también tener atributos que la describan. Para ilustrar esta situación, observar el siguiente ejemplo: considerar la relación *CURSA* entre las entidades *ALUMNO* y *ASIGNATURA*. Se podría asociar a la relación *CURSA* un atributo nota para especificar la nota que

ha obtenido un alumno en una determinada asignatura.

Otro **ejemplo típico son las relaciones que representan históricos**. Este tipo de relaciones **suele constar de datos como fecha y hora**. Cuando se emite una factura a un cliente o se le facilita un duplicado de la misma, es necesario registrar el momento en el que se ha realizado dicha acción. Para ello, habrá que crear un atributo asociado a la relación entre la entidad *CLIENTE* y *FACTURA* que se encargue de guardar la fecha de emisión.

En el modelo *E/R* **la representación de atributos asociados a relaciones es exactamente igual a la que se utiliza para entidades**. Se puede utilizar **una elipse con el nombre del atributo en su interior, conectada con una línea a la relación, o bien, un círculo blanco conectado con una línea a la relación y junto a él, el nombre del atributo**. En la imagen anterior se puede ver esta segunda representación.

2.4 Relaciones.

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

> ***Relación**: es una asociación entre diferentes entidades. En una relación no pueden aparecer dos veces relacionadas las mismas ocurrencias (o instancias) de entidad.*

La **representación gráfica** en el modelo *Entidad/Relación* corresponde a un **rombo en cuyo interior se encuentra inscrito el nombre de la relación**. El **rombo estará conectado con las entidades a las que relaciona, mediante líneas rectas**.

> *Cuando se deba **dar un nombre a una relación** hay que procurar 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 relación**, se **indique el papel o rol que desempeña cada entidad en la relación**. Como se verá más adelante, los papeles o roles son **especialmente útiles en relaciones reflexivas**. **Normalmente es implícito y no se suele especificar** (son útiles cuando el significado de una relación necesita ser calificado).

Normalmente las relaciones no tienen atributos, cuando surge una relación con atributos significa que debajo hay una entidad que aún no se ha definido. A esa entidad se le llama **entidad asociada** (esta entidad dará origen a una tabla que contendrá esos atributos).

Un **conjunto de relaciones** es un conjunto de relaciones del mismo tipo, por ejemplo, entre *JUGADOR* y *EQUIPO* todas las asociaciones existentes entre los jugadores y los equipos. La **mayoría de los conjuntos de relaciones en un sistema de *BD* son binarias** (dos entidades) **aunque puede haber conjuntos de relaciones que impliquen más de dos conjuntos de entidades**.

Para describir y definir adecuadamente las relaciones existentes entre entidades, es imprescindible conocer los siguientes conceptos:

- ✓ **Grado** de la relación.
- ✓ **Cardinalidad de la relación.**
- ✓ **Cardinalidades de las entidades.**

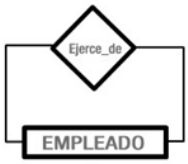
2.4.1 Grado de la relación.

> ***Grado de una relación**: número de entidades que participan en una relación.*

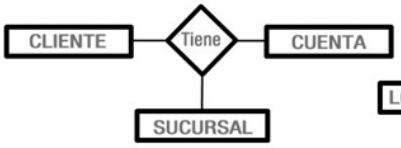
En función del grado se pueden establecer diferentes **tipos de relaciones**:

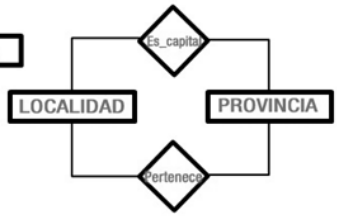
- ✓ **Relación unaria o de grado 1**: es aquella relación en la que **participa una única entidad**. También llamadas **reflexivas o recursivas**.
- ✓ **Relación binaria o de grado 2**: es aquella relación en la que **participan dos entidades**. En general, tanto en una primera aproximación, como en los sucesivos refinamientos, **el esquema conceptual de la *BD* buscará tener sólo este tipo de relaciones**.
- ✓ **Relación ternaria o de grado 3**: es aquella relación en la que **participan tres entidades** al mismo tiempo.
- ✓ **Relación n-aria o de grado n**: es aquella **relación que involucra n entidades**. Este tipo de relaciones no son usuales y deben ser simplificadas hacia relaciones de menor grado.
- ✓ **Relación doble**: ocurre **cuando dos entidades están relacionadas a través de dos relaciones**. Este tipo de relaciones **son complejas de manejar** y se deben evitar.

Relación Binaria



Relación Unaria (Reflexiva)



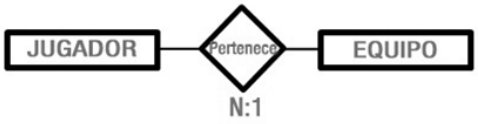


Relación Doble

2.4.2 Cardinalidad de la relación.

***Cardinalidad de una relación**: es el **número máximo de ocurrencias de cada entidad que pueden intervenir en una ocurrencia de relación**. 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**.*

Observar el ejemplo de la imagen, la cardinalidad indicará el número de ocurrencias de la entidad *JUGADOR* que se relacionan con cada ocurrencia de la entidad *EQUIPO* y viceversa. Se podría hacer la siguiente lectura: un jugador pertenece a un equipo y a un equipo pueden pertenecer varios jugadores.

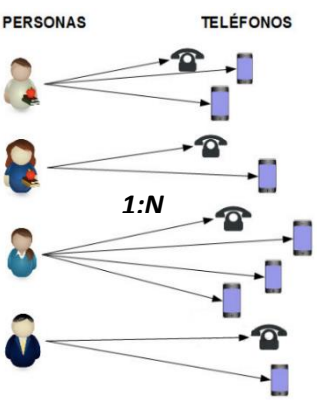


Una posible representación de la **cardinalidad de las relaciones** es la que se ha visto en el ejemplo anterior. Se podrían representar el resto de cardinalidades mediante las etiquetas ***1:1***, ***1:N***, ***N:1***, ***N:M*** que se leerían respectivamente: uno a uno, uno a muchos, muchos a uno y muchos a muchos.

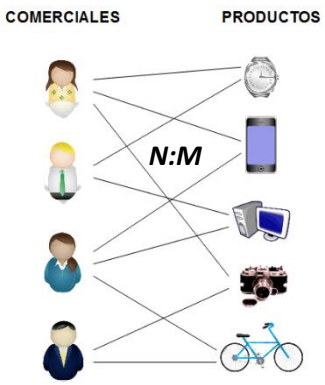
Cardinalidades:

- ✓ **Relaciones uno a uno (*1:1*)**. **Sean las entidades *A* y *B*, una instancia u ocurrencia de la entidad *A* se relaciona únicamente con otra instancia de la entidad *B* y viceversa**. Por ejemplo, para cada ocurrencia de la entidad *ALUMNO* sólo habrá una ocurrencia relacionada de la entidad *EXPEDIENTE* y viceversa. O lo que es lo mismo, un alumno tiene un expediente asociado y un expediente sólo pertenece a un único alumno. Otro ejemplo en el caso de matrimonio se presenta en la imagen.

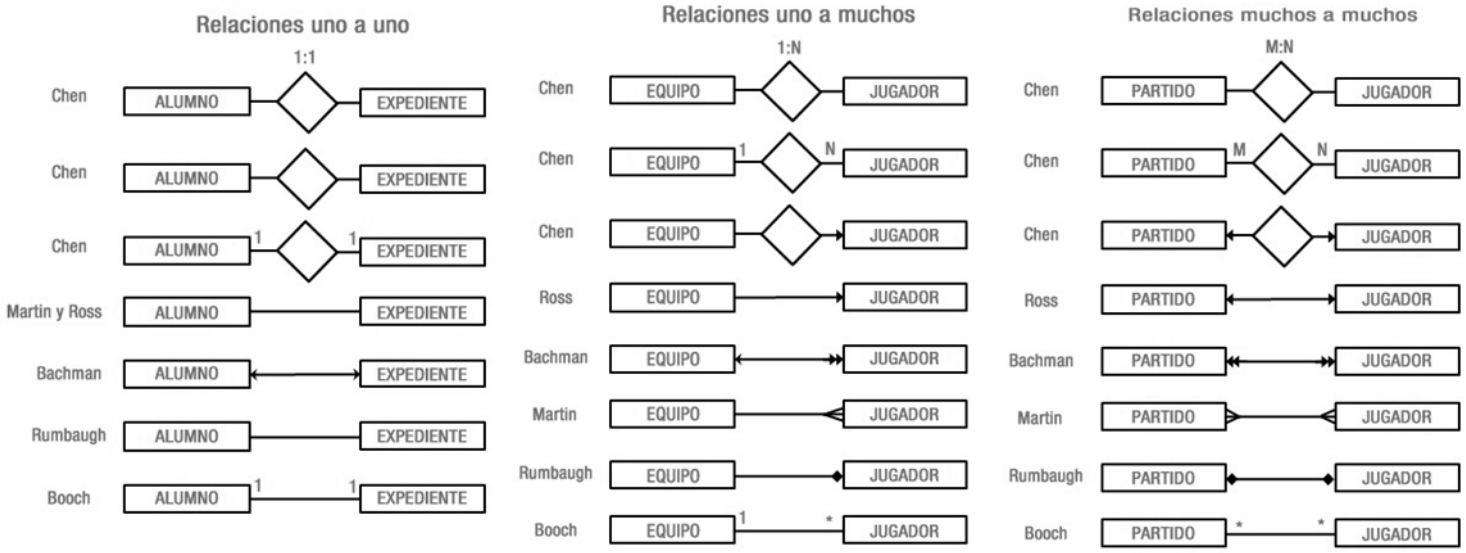
- ✓ **Relaciones uno a muchos (*1:N*)**. **Sean las entidades *A* y *B*, una ocurrencia de la entidad *A* se relaciona con muchas ocurrencias de la entidad *B* y una ocurrencia de la entidad *B* sólo estará relacionada con una única ocurrencia de la entidad *A***. Por ejemplo, para cada ocurrencia de la entidad *DOCENTE* puede haber varias ocurrencias de la entidad *ASIGNATURA* y para varias ocurrencias de la entidad *ASIGNATURA* sólo habrá una ocurrencia relacionada de la entidad *DOCENTE* (si se establece que una asignatura sólo puede ser impartida por un único docente). O lo que es lo mismo, un docente puede impartir varias asignaturas y una asignatura sólo puede ser impartida por un único docente. En la imagen se muestra un ejemplo en el que se tienen *PERSONAS* y *TELÉFONOS*, donde una persona puede tener varios teléfonos y un teléfono solo es de una persona.



- ✓ **Relaciones muchos a uno (*N:1*). Sean las entidades *A* y *B*, una ocurrencia de la entidad *A* está asociada con una única ocurrencia de la entidad *B* y un ejemplar de la entidad *B* está relacionado con muchas ocurrencias de la entidad *A***. Por ejemplo, un jugador pertenece a un único equipo y a un equipo pueden pertenecer muchos jugadores.
- ✓ **Relaciones muchos a muchos (*N:M*). Sean las entidades *A* y *B*, un ejemplar de la entidad *A* está relacionado con muchas ocurrencias de la entidad *B* y viceversa**. Por ejemplo, un alumno puede estar matriculado en varias asignaturas y en una asignatura pueden estar matriculados varios alumnos. En la imagen en el que se tienen *COMERCIALES* y *PRODUCTOS*, donde un comercial puede vender muchos productos y a su vez un mismo producto pueden ser vendidos por diferentes comerciales.

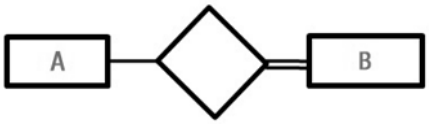


La cardinalidad de las relaciones puede representarse de varias maneras en los esquemas del modelo *Entidad/Relación*. A continuación, se muestra un **resumen de las notaciones**, clasificadas por autores, **más empleadas en la representación de cardinalidad de relaciones**:



2.4.3 Cardinalidad de las entidades.

La **cardinalidad con la que una entidad participa en una relación especifica el número mínimo y el número máximo de correspondencias en las que puede tomar parte cada ejemplar de dicha entidad**. Indica el número de relaciones en las que una entidad puede aparecer.

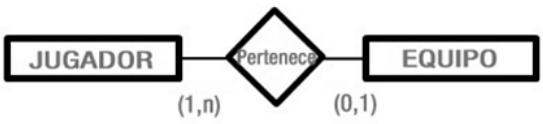


Sean las entidades *A* y *B*, la **participación de la entidad *A* en una relación es <u>obligatoria (total)</u> si la existencia de cada una de sus ocurrencias necesita como mínimo de una ocurrencia de la entidad *B*** (ver imagen). En **caso contrario**, la participación es **opcional (parcial)**.

La **cardinalidad de una entidad se representa con el número mínimo y máximo de correspondencias en las que puede tomar parte cada ejemplar de dicha entidad, entre paréntesis**. Su representación gráfica será, por tanto, una **etiqueta del tipo *(0,1)*, *(1,1)*, *(0,n)* o *(1,n)***. El significado del primer y segundo elemento del paréntesis corresponde a **(cardinalidad mínima, cardinalidad máxima)**:

- ✓ **Cardinalidad mínima**: indica el **número mínimo de asociaciones en las que aparecerá cada ocurrencia de la entidad** (el valor que se anota es de ***0*** o ***1***, aunque tenga una cardinalidad mínima de más de uno, se indica sólo un uno). El valor ***0*** se pondrá **cuando la participación de la entidad sea opcional**.
- ✓ **Cardinalidad máxima**: indica el **número máximo de relaciones en las que puede aparecer cada ocurrencia de la entidad**. Puede ser ***1***, otro **valor concreto mayor que uno** o **muchos** (se representa con ***n***).

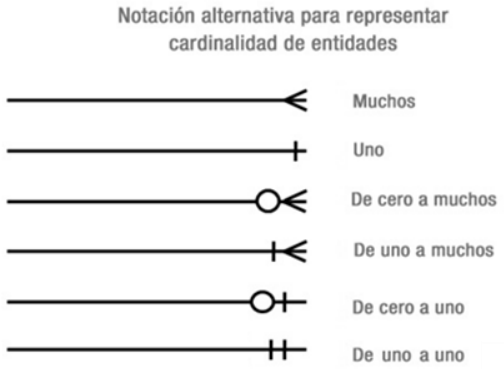
<u>Ejemplo:</u> un *JUGADOR* pertenece como mínimo a ningún *EQUIPO* y como máximo a uno *(0,1)* y, por otra parte, a un *EQUIPO* pertenece como mínimo un *JUGADOR* y como máximo varios *(1,n)*. Como se puede ver, la cardinalidad *(0,1)* de *JUGADOR* se ha colocado junto a la entidad



EQUIPO para representar que un jugador puede no pertenecer a ningún equipo o como máximo a uno. Para la cardinalidad de *EQUIPO* ocurre igual, se coloca su cardinalidad junto a la entidad *JUGADOR* para expresar que en un equipo hay mínimo un jugador y máximo varios.

> *Tener en cuenta que cuando se representa la cardinalidad de una entidad, el paréntesis y sus valores han de colocarse junto a la entidad con la que se relaciona. Es decir, en el lado opuesto a la relación.*

La cardinalidad de entidades también puede representarse en el modelo *Entidad/Relación* con la notación patas de gallo que se representa en la imagen de la derecha. Por tanto, el anterior ejemplo quedaría representado así:



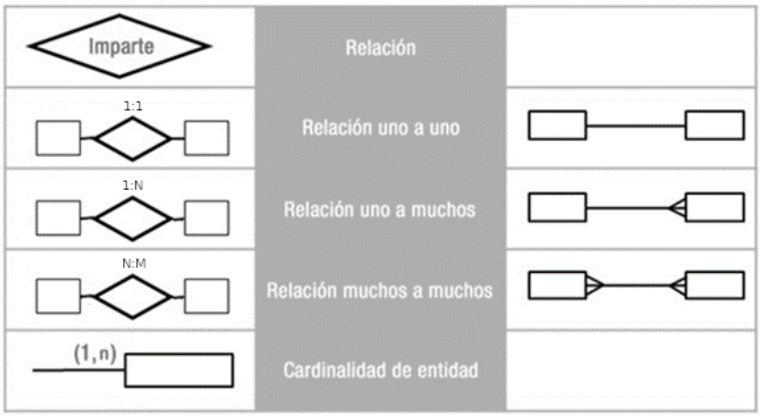
Ejercicio resuelto

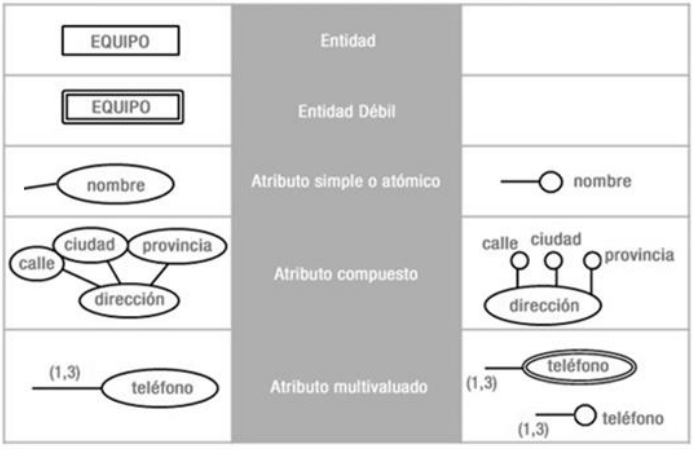
Suponer que se está diseñando una *BD* para un sitio de juegos online. En un punto del proceso de diseño se ha de modelar el siguiente requisito: cada usuario registrado podrá crear las partidas que desee (a las que otros usuarios pueden unirse), pero una partida solo podrá estar creada por un único usuario. Un usuario podrá o no crear partidas. ¿Cuáles serían las etiquetas del tipo (cardinalidad mínima, cardinalidad máxima) que deberían ponerse junto a las entidades *USUARIO* y *PARTIDA* respectivamente, si éstas están asociadas por la relación *crear* (partida)?

Solución: *(1,1)* y *(0,n)* → Con estas cardinalidades se indica que un usuario puede crear varias partidas, o ninguna. Por otra parte, una partida deberá estar creada exclusivamente por un único usuario.

2.5 Simbología del modelo *E/R*.

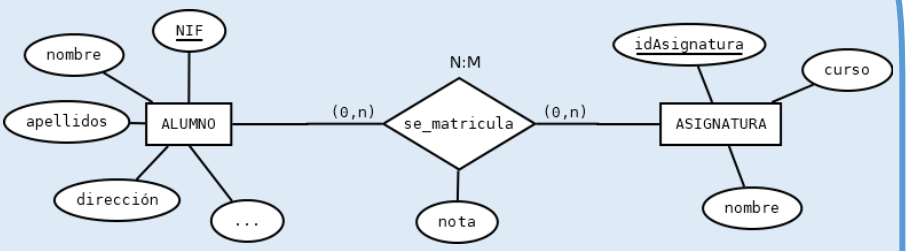
Resumen básico de los **símbolos utilizados en el modelo *Entidad/Relación***. Se puede ver que **existen diferentes maneras de representar los mismos elementos**, las que aquí se resumen permitirán interpretar la gran mayoría de esquemas que se pueden encontrar.





Ejercicio resuelto

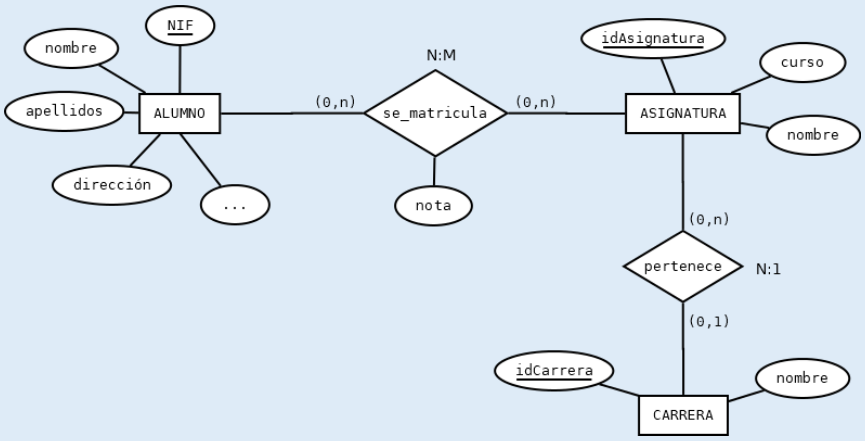
Se desea diseñar un esquema relacional de una *BD* para un centro de enseñanzas que contenga información sobre los alumnos, las asignaturas y las calificaciones que se obtienen en cada una de las mismas. Desarrollar el modelo *E/R*.



Ejercicio resuelto

Se desea diseñar una *BD* para una universidad que contenga información sobre los alumnos, las asignaturas y las carreras que se pueden estudiar. Construir un modelo *E/R* que tenga en cuenta las siguientes restricciones:

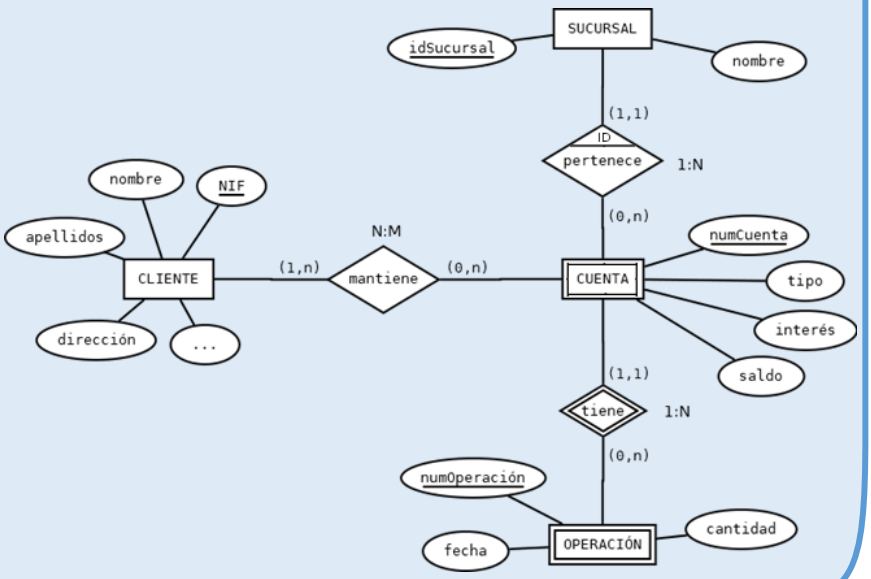
- ✓ **Un alumno puede estar matriculado en muchas asignaturas. Se desea guardar la nota de cada asignatura.**
- ✓ **Una asignatura solo puede pertenecer a una sola carrera.**
- ✓ **Una carrera puede tener muchas asignaturas.**



Ejercicio resuelto

Se desea diseñar una *BD* para una entidad bancaria que contenga información sobre los clientes, las cuentas, las sucursales y las operaciones realizadas en cada una de las cuentas. Construir un modelo *E/R* teniendo en cuenta las siguientes restricciones:

- ✓ **Una operación viene determinada por su número de operación, la fecha y la cantidad.**
- ✓ **Un cliente puede tener muchas cuentas.**
- ✓ **Una cuenta puede pertenecer a varios clientes.**
- ✓ **Una cuenta solamente puede estar en una sucursal.**



3 Modelo *E/R Extendido.*

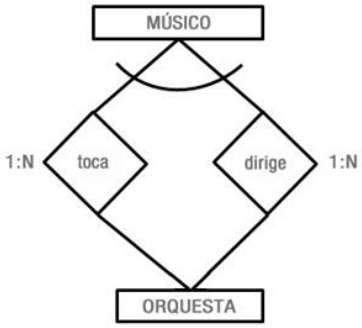
A través del modelo *Entidad/Relación* se pueden modelar la gran mayoría de los requisitos que debe cumplir una *BD*, pero existen algunos que ofrecen especial dificultad a la hora de representarlos a través de la simbología tradicional del modelo *E/R*. Para solucionar este problema, **en el modelo *E/R Extendido* se han incorporado nuevas extensiones que permiten mejorar la capacidad para representar circunstancias especiales**. Estas extensiones intentan eliminar elementos de difícil o incompleta representación a través de la simbología existente.

A continuación, se detallan estas **nuevas características** que convierten al modelo *E/R* tradicional en el modelo *E/R Extendido*, como son: tipos de **restricciones sobre las relaciones, especialización/generalización y agregación**.

3.1 Restricciones en las relaciones.

La primera extensión que el modelo *E/R Extendido* incluye, se centra en la representación de una serie de restricciones sobre las relaciones y sus ejemplares:

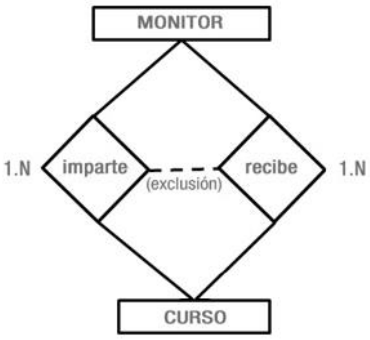
a) Restricción de exclusividad.



Cuando existe una entidad que participa en dos o más relaciones y cada ocurrencia de dicha entidad sólo puede pertenecer a una de las relaciones únicamente, decimos que existe una restricción de exclusividad. Si la ocurrencia de entidad pertenece a una de las relaciones, no podrá formar parte de la otra. **O se produce una relación o se produce otra, pero nunca ambas, ni a la vez ni por separado**.

La **representación gráfica** en el modelo *Entidad/Relación Extendido* de una restricción de exclusividad se realiza **mediante un arco que engloba a todas aquellas relaciones que son exclusivas**.

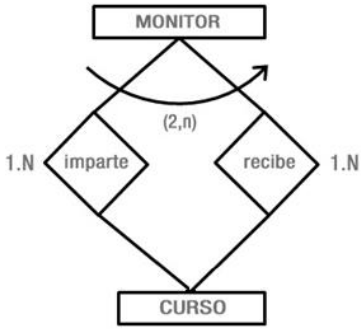
b) Restricción de exclusión.



Este tipo de restricción se produce **cuando las ocurrencias de las entidades sólo pueden asociarse utilizando una única relación**. **O se produce una relación o se produce otra, pero nunca ambas de forma simultánea**. La entidad si podrá actuar en todas las relaciones, pero siempre por separado.

Ejemplo: suponer que un monitor puede impartir diferentes cursos de perfeccionamiento para monitores, y que éste puede a su vez recibirlos. Pero si un monitor imparte un determinado curso, no podrá estar recibiéndolo simultáneamente y viceversa. Se establecerá, por tanto, una restricción de exclusión que **se representa** mediante una **línea discontinua entre las dos relaciones**, tal y como se muestra en el ejemplo.

c) Restricción de inclusividad.

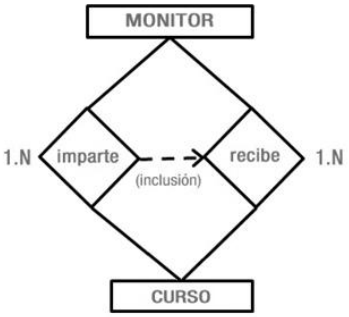


Este tipo de restricciones se aplican cuando es necesario modelar **situaciones en las que para que dos ocurrencias de entidad se asocien a través de una relación, tengan que haberlo estado antes a través de otra relación**.

Ejemplo: suponer que para que un monitor pueda impartir cursos de cocina sea necesario que reciba previamente dos cursos: nutrición y primeros auxilios. Como se puede ver, es posible que los cursos que el monitor deba recibir no tengan que ser los mismos que luego pueda impartir. Aplicando una restricción de inclusividad entre las relaciones imparte y recibe, se estará indicando que cualquier ocurrencia de la entidad *MONITOR* que participa en una de las relaciones (imparte) tiene que participar obligatoriamente en la otra (recibe).

Se representará mediante un **arco acabado en flecha**, que **partirá desde la relación que ha de cumplirse la última hacia la otra relación**. Se indicará **junto al arco la cardinalidad mínima y máxima** de dicha restricción de inclusividad. En el ejemplo, (2,n) indica que un monitor ha de recibir 2 cursos antes de poder impartir varios.

d) Restricción de inclusión.



En algunas ocasiones aplicar una restricción de inclusividad no representa totalmente la realidad a modelar, entonces se hace necesario aplicar una restricción de inclusión que **es aún más fuerte**.

Ejemplo: si se ha de modelar que un monitor pueda impartir un curso, si previamente lo ha recibido, entonces se tendrá que aplicar una restricción de inclusión. Con ella toda ocurrencia de la entidad *MONITOR* que esté asociada a una ocurrencia determinada de la entidad *CURSO*, a través de la relación imparte,

ha de estar unida a la misma ocurrencia de la entidad *CURSO* a través de la relación recibe.

Ejercicio resuelto

Suponer que se ha de modelar mediante el modelo *Entidad/Relación Extendido* el siguiente requerimiento de una *BD*: "Para que un hombre se divorcie de una mujer, primero ha de haber estado casado con ella".

Las entidades participantes son *MUJER* y *HOMBRE*, que estarán asociadas a través de dos relaciones: *se casa* y *se divorcia*. No se tendrá en cuenta la cardinalidad de ambas relaciones.

¿Qué tipo de restricción sobre las relaciones se ha de establecer en el esquema para representar correctamente este requisito?

Solución: Restricción de inclusión → Este tipo de restricción establece la obligatoriedad de haber un casamiento para que pueda haber un divorcio y, además, las entidades que se relacionan a través de la relación *se casa*, deben ser las mismas que las participantes en *se divorcia*.

3.2 Generalización y especialización.

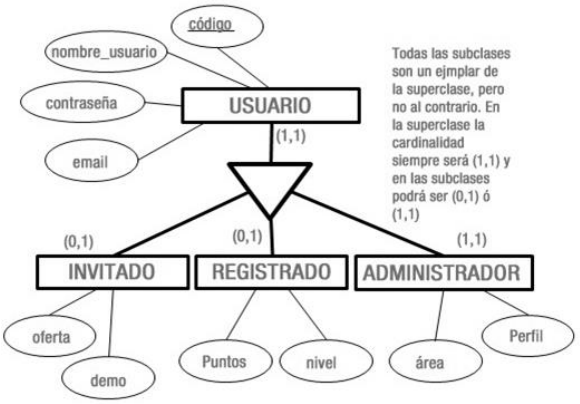
La segunda extensión incorporada en el modelo *Entidad/Relación Extendido* se centra en nuevos tipos de relaciones que van a permitir modelar la realidad de una manera más fiel. Estos nuevos tipos de relación reciben el nombre de **jerarquías** y **se basan en los conceptos de generalización, especialización y herencia**.

Cuando se está diseñando una *BD* **puede que nos encontremos con conjuntos de entidades que posean características comunes, lo que permitiría crear un tipo de entidad de nivel más alto que englobase dichas características**. Y a su vez, **puede que se necesite dividir un conjunto de entidades en diferentes subgrupos de entidades por tener éstas, características diferenciadoras**. Este proceso de refinamiento ascendente/descendente, permite **expresar mediante la generalización la existencia de tipos de entidades de nivel superior que engloban a conjuntos de entidades de nivel inferior**. A los **conjuntos de entidades de nivel superior** también se les denomina **superclase o supertipo**. A los **conjuntos de entidades de nivel inferior** se les denomina **subclase o subtipo**.

Por tanto, existirá la posibilidad de realizar una **especialización de una superclase en subclases**, y análogamente, **establecer una generalización de las subclases en superclases**. La generalización es la reunión en una superclase o supertipo de entidad de una serie de subclases o subtipos de entidades, que poseen características comunes. Las subclases tendrán otras características que las diferenciarán entre ellas.

¿Cómo detectar una generalización? **Se puede identificar una generalización cuando se encuentren una serie de atributos comunes a un conjunto de entidades, y otros atributos que sean específicos. Los atributos comunes conforman la superclase o supertipo y los atributos específicos la subclase o subtipo.**

Las jerarquías se caracterizan por un concepto que se ha de tener en cuenta, **la herencia**. A través de la herencia **los atributos de una superclase de entidad son heredados por las subclases**. Si una superclase interviene en una relación, las subclases también lo harán.



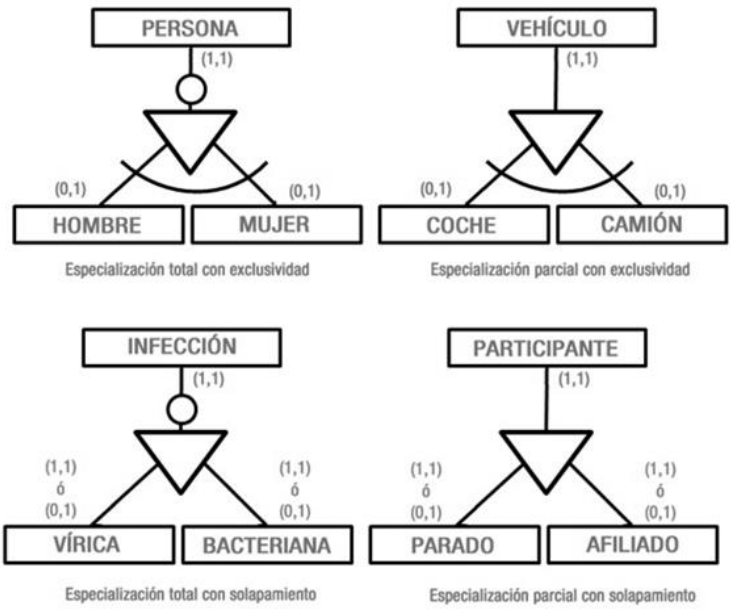
¿Cómo se representa una generalización o especialización? Existen varias notaciones, pero se ha de convenir que la relación que se establece entre una superclase de entidad y todos sus subtipos se expresa a través de las palabras *ES UN*, o en notación inglesa *IS A*, que correspondería con *ES UN TIPO DE*. Partiendo de este punto, **una jerarquía se representa mediante un triángulo invertido, sobre él quedará la entidad superclase y conectadas a él a través de líneas rectas, las subclases**.

En el ejemplo de la imagen, las subclases *INVITADO*, *REGISTRADO* y *ADMINISTRADOR* constituyen subclases de la superclase *USUARIO*. Cada una de ellas aporta sus propias características y heredan las pertenecientes a su superclase.

Una generalización/especialización podrá tener las siguientes **restricciones semánticas**:

- ✓ **Totalidad**: **si todo ejemplar de la superclase pertenece a alguna de las subclases (○)**.
- ✓ **Parcialidad**: **si no todos los ejemplares de la superclase pertenecen a alguna de las subclases**.
- ✓ **Solapamiento**: si **un mismo ejemplar de la superclase puede pertenecer a más de una subclase**.
- ✓ **Exclusividad**: si **un mismo ejemplar de la superclase pertenece sólo a una subclase (⌒)**.

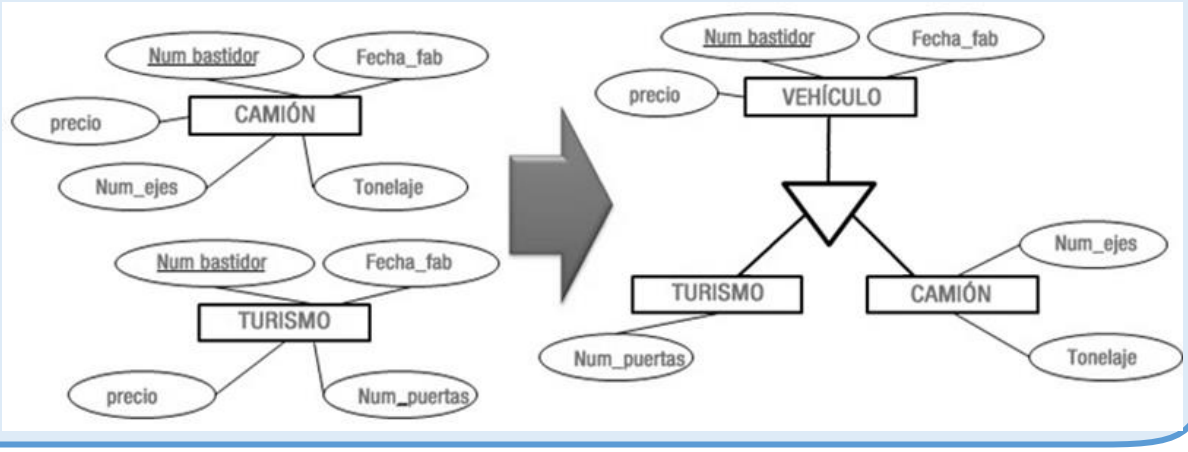
Las **diferentes restricciones semánticas descritas tienen su representación gráfica**, a través del gráfico que se muestra se podrá entender mejor su funcionamiento.



Ejercicio resuelto

Suponer la existencia de dos entidades *TURISMO* y *CAMIÓN*. Los atributos de la entidad *TURISMO* son: *Num_bastidor*, *Fecha_fab*, *precio* y *Num_puertas*. Los atributos de la entidad *CAMIÓN* son: *Num_bastidor*, *Fecha_fab*, *precio*, *Num_ejes* y *Tonelaje*.

Si se analizan ambas entidades existen algunos atributos comunes y otros que no. Por tanto, se podrá establecer una jerarquía. Para ello, se reunirán los atributos comunes y se asociarán a una nueva entidad superclase denominada *VEHÍCULO*. Las subclases *TURISMO* y *CAMIÓN*, con sus atributos específicos, quedarán asociadas a la superclase *VEHÍCULO* mediante una jerarquía parcial con solapamiento. En el siguiente gráfico se puede apreciar la transformación.



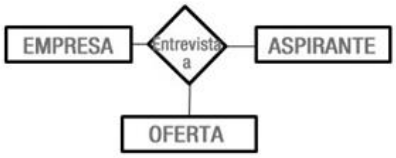
3.3 Agregación.

En el modelo *Entidad/Relación* no es posible representar relaciones entre relaciones. La agregación **es una abstracción a través de la cual las relaciones se tratan como entidades de nivel más alto, siendo utilizada para expresar relaciones entre relaciones o entre entidades y relaciones**.

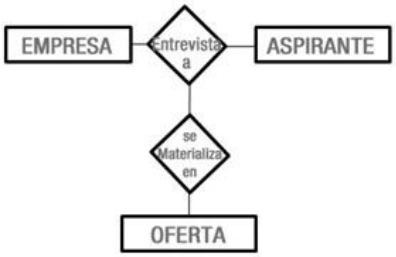
Suponer que se ha de modelar la siguiente situación: una empresa de selección de personal realiza entrevistas a diferentes aspirantes. Puede ser que, de algunas de esas entrevistas a aspirantes, se derive una oferta de empleo, o no. En el siguiente gráfico se representan tres soluciones, las dos primeras erróneas y una tercera correcta, utilizando una agregación.

Como se puede observar, **la representación gráfica de una agregación se caracteriza por englobar con un**

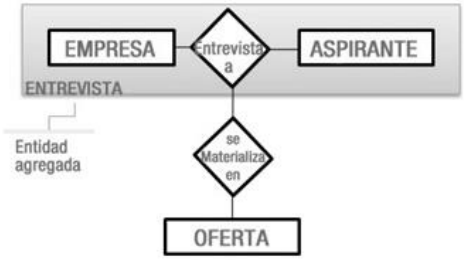
rectángulo las entidades y relación a abstraer. De este modo, **se crea una nueva entidad agregada que puede participar en otras relaciones con otras entidades**. En este tipo de relación especial de agregación, **la cardinalidad máxima y mínima de la entidad agregada siempre será (1,1) no indicándose por ello en el esquema**.



Solución 1: Errónea, ya que estaríamos representando que, por cada entrevista realizada por una empresa a un aspirante, se egenera una oferta de empleo

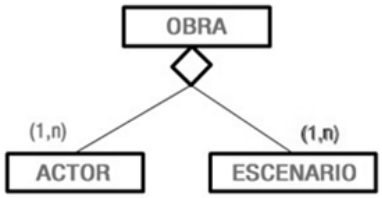


Solución 2: Errónea, porque en el modelo E/R no pueden establecerse relaciones entre varias relaciones

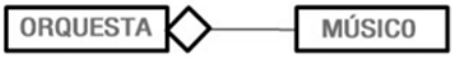


Solución 3: En el modelo E/R Extendido, puede crearse una entidad agregada llamada ENTREVISTA, compuesta por la relación *"Entrevista a"* que existe entre EMPRESA y ASPIRANTE. Entre esta nueva entidad y OFERTA si puede establecerse una relación *"se materializa en"*

Existen **dos clases de agregaciones**:



- ✓ **Compuesto/componente**: **un todo se obtiene por la unión de diversas partes, que pueden ser objetos distintos y que desempeñan papeles distintos en la agregación**. Teniendo esto en cuenta, esta abstracción permite representar que un todo o agregado se obtiene por la unión de diversas partes o componentes que pueden ser tipos de entidades distintas y que juegan diferentes roles en la agregación.



- ✓ **Miembro/Colección**: **un todo se obtiene por la unión de diversas partes del mismo tipo y que desempeñan el mismo papel en la agregación**. Teniendo esto en cuenta, esta abstracción permite representar un todo o agregado como una colección de miembros, todos de un mismo tipo de entidad y todos jugando el mismo rol. Esta agregación puede incluir una restricción de orden de los miembros dentro de la colección (indicando el atributo de ordenación). Es decir, permite establecer un orden entre las partes.

4 Elaboración de diagramas *Entidad/Relación*.

Llegados a este punto, surgirán varias dudas ¿cómo se crea un diagrama *E/R*? ¿por dónde se empieza? ¿y qué se puede hacer con todo lo visto? Son cuestiones totalmente normales cuando se comienza, se van a ver una serie de orientaciones para poder aplicar todos los conceptos aprendidos en la elaboración de diagramas *Entidad/Relación*.

Se sabe que, en la fase de diseño conceptual de la *BD*, en la que nos encontramos, **se ha de generar el diagrama *E/R* que representará de manera más sencilla el problema real a modelar, independientemente del *SGBD***. Este esquema será como un plano que facilite la comprensión y solución del problema. Este diagrama **estará compuesto por la representación gráfica, a través de la simbología vista, de los requisitos o condiciones que se derivan del problema a modelar**.

Saltar este paso en el proceso de creación e implementación de una *BD*, supondría pérdida de información. Por lo que esta fase, requerirá de la creación de uno o varios esquemas previos más cercanos al mundo real, antes del paso a tablas del modelo relacional.

Como en la programación, **la práctica es fundamental**. Los diagramas **no siempre se crean del mismo modo y, en ocasiones, hay que retocarlos e incluso rehacerlos**. A través de la resolución de diferentes problemas y la elaboración de múltiples diagramas, se obtendrá la destreza necesaria para generar esquemas que garanticen una posterior y correcta conversión del modelo *Entidad/Relación* al *modelo Relacional*.

4.1 Identificación de entidades y relaciones.

Lo primero que **se ha de tener** para poder generar un diagrama *E/R* adecuado es **el conjunto de requerimientos, requisitos o condiciones que la *BD* ha de cumplir**. Es lo que se denomina el **documento de especificación de requerimientos**. En otras palabras, el enunciado del problema a modelar. Cuanto más completa y detallada sea la información de la que se disponga, mucho mejor.

¿Cómo se empieza? Las **etapas para la creación del diagrama *E/R*** son:

a) **Identificación de entidades**: es un proceso bastante intuitivo. Para localizar aquellos elementos que serán las entidades del esquema, **analizar la especificación de requerimientos en busca de nombres o sustantivos**. Si estos nombres se refieren a objetos importantes dentro del problema probablemente serán entidades. Se tendrá en cuenta que **nombres referidos a características, cualidades o propiedades no se convertirán en entidades**.

Otra forma de identificar entidades es localizando objetos o elementos que existen por sí mismos. Por ejemplo: *VEHICULO*, *PIEZA*, etc. En otras ocasiones, la localización de varias características o propiedades puede dejar ver la existencia de una entidad.

¿Esto puede ser una entidad o no? Es una pregunta que se repite mucho cuando se está en esta etapa. Algunos autores indican que **para poder considerarse como entidad se deben cumplir tres reglas**:

- ✓ **Existencia propia**.
- ✓ **Cada ejemplar de un tipo de entidad debe poder ser diferenciado del resto** de ejemplares.
- ✓ **Todos los ejemplares de un tipo de entidad deben tener las mismas propiedades**.

El número de **entidades** obtenidas debe ser manejable y según se vayan identificando **se les otorgarán nombres, preferiblemente en mayúsculas y singular, representativos** de su significado o función. De esta manera el diagrama será cada vez más legible.

b) **Identificación de relaciones**: **localizadas las entidades, se deben establecer qué relaciones existen entre ellas**. Para ello, analizar de nuevo el documento de especificación de requerimientos en busca de **verbos o expresiones verbales que conecten unas entidades con otras**. En la gran mayoría de ocasiones se encontrará que las relaciones se establecen entre dos entidades (relaciones binarias), pero se prestará especial atención a las relaciones entre más entidades y a las relaciones recursivas o relaciones **unarias**.

Cada una de las **relaciones** establecidas deberá **tener asignado un nombre, preferiblemente en minúsculas, representativo** de su significado o acción.

En ocasiones, el identificador de una relación está compuesto por varias palabras, como, por ejemplo: es supervisado, trabaja para, etc. Es recomendable utilizar guiones bajos para unir las palabras que forman el identificador.

Dependiendo de la notación elegida, el **siguiente paso será la representación de la cardinalidad** (mínima y máxima) **de las entidades participantes en cada relación y del tipo de correspondencia de la relación** (1:1, 1:N o N:M). Si se ha encontrado alguna **relación recursiva**, reflexiva o unaria, se ha de **representar en el esquema los roles desempeñados** por la entidad en dicha relación.

4.2 Identificación de atributos, claves y jerarquías.

Sólo con la localización de entidades y relaciones no está todo hecho. Se ha de completar el proceso realizando las siguientes tareas:

a) **Identificación de atributos**: volver sobre el documento de especificación de requerimientos para **buscar nombres relativos a características, propiedades, identificadores o cualidades de entidades o relaciones**. Resulta más sencillo si nos preguntamos ¿Qué información es necesario tener en cuenta de una u otra entidad o relación? **Quizás no todos los atributos estén reflejados directamente en el documento de especificación de requerimientos, aplicando el sentido común el diseñador podrá establecerlos en algunos casos y en otros, será necesario consultar e indagar en el problema**.

Tener en cuenta si los atributos localizados son simples o compuestos, derivados o calculados y si algún atributo o conjunto de ellos se repite en varias entidades. Si se da este último caso, **plantear la posibilidad de establecer una jerarquía de especialización**, o bien, dejar las entidades tal y como han sido identificadas.

Cada atributo deberá tener asignado un nombre, preferiblemente en minúsculas, representativo de su contenido o función. Además, siempre es recomendable recopilar la siguiente información de cada atributo:

- ✓ Nombre y descripción.
- ✓ Atributos simples que lo componen, si es atributo compuesto.
- ✓ Método de cálculo, si es atributo derivado o calculado.

En el **caso de encontrar atributos asociados a relaciones con cardinalidad uno a muchos, se valorará asignar ese atributo o atributos a la entidad con mayor cardinalidad** participante en la relación.

b) **Identificación de claves**: del conjunto de atributos de una entidad **se establecerán una o varias claves candidatas, escogiéndose una de ellas como clave o llave primaria de la entidad**. Esta clave estará **formada por uno o varios atributos que identificarán de manera unívoca cada ocurrencia de entidad**. El proceso de **identificación de claves permitirá determinar la fortaleza (al menos una clave candidata) o debilidad (ninguna clave candidata) de las entidades encontradas**.

Se representará la existencia de esta clave primaria mediante la notación elegida para la elaboración del diagrama *E/R*. Del mismo modo, **se deberán representar adecuadamente las entidades fuertes o débiles**.

c) **Determinación de jerarquías**: como se ha comentado anteriormente, es probable que existan **entidades con características comunes que puedan ser generalizadas** en una entidad de nivel superior o superclase (jerarquía de generalización). Pero también, puede ser necesario expresar en el esquema las particularidades de diferentes ejemplares de un tipo de entidad, por lo que se crearán subclases o subtipos de una superclase o supertipo (jerarquía de especialización). Para ello, habrá que analizar con detenimiento el documento de especificación de requerimientos.

Si se identifica algún tipo de jerarquía, se deberá representar adecuadamente según el tipo de notación elegida, **determinando si la jerarquía es total/parcial o exclusiva/con solapamiento**.

4.3 Metodologías.

Hasta aquí, se tienen identificados los elementos necesarios para construir el diagrama, pero ¿Existe alguna metodología para llevarlo a cabo? Sí, y además se podrán utilizar varias. Se partirá de una versión preliminar del esquema conceptual o diagrama *E/R* que, tras sucesivos refinamientos, será modificado para obtener el diagrama *E/R* definitivo. Las **metodologías o estrategias disponibles para la elaboración del esquema conceptual** son las siguientes:

- ✓ **Metodología Descendente (*Top-Down*)**: se trata de **partir de un esquema general e ir descomponiendo éste en niveles, cada uno de ellos con mayor número de detalles**. Se parte de objetos muy abstractos, que se refinan paso a paso hasta llegar al esquema final.
- ✓ **Metodología Ascendente (*Bottom-Up*)**: inicialmente, **se parte del nivel más bajo, los atributos. Se irán agrupando en entidades, para después ir creando las relaciones entre éstas y las posibles jerarquías** hasta obtener un diagrama completo. Se parte de objetos atómicos que no pueden ser descompuestos y a continuación se obtienen abstracciones u objetos de mayor nivel de abstracción que forman el esquema.
- ✓ **Metodología Dentro-fuera (*Inside-Out*)**: inicialmente **se comienza a desarrollar el esquema en una parte del papel y a medida que se analiza la especificación de requerimientos, se va completando con entidades y relaciones** hasta ocupar todo el documento.
- ✓ **Metodología Mixta**: es empleada en problemas complejos. **Se dividen los requerimientos en subconjuntos que serán analizados independientemente**. Se crea un esquema que servirá como estructura en la que irán interconectando los conceptos importantes con el resultado del análisis de los subconjuntos creados. Esta metodología utiliza las técnicas ascendente y descendente. Se aplicará la técnica descendente para dividir los requerimientos y en cada subconjunto de ellos, se aplicará la técnica ascendente.

¿Cuál de estas metodologías utilizar? Cualquiera de ellas puede ser válida, todo dependerá de lo fácil y útil que nos resulte aplicarlas. Probablemente y, casi sin ser consciente de ello, **uno mismo creará su propia metodología combinando las existentes**. Pero, como se decía hace algunos puntos, **la práctica es fundamental**. Realizando gran

cantidad de esquemas, analizándolos y llevando a cabo modificaciones en ellos es como se irá refinando la técnica de elaboración de diagramas *E/R*. **Llegará un momento en que sólo con leer el documento de especificación de requerimientos se sea capaz de ir construyendo en la mente cómo será su representación sobre el papel**.

4.4 Redundancia en diagramas *Entidad/Relación*.

> ***Redundancia**: en BD hace referencia al almacenamiento de los mismos datos varias veces en diferentes lugares.*

La **redundancia de datos puede provocar problemas** como:

- ✓ **Aumento de la carga de trabajo**: al estar almacenado un dato en varios lugares, las operaciones de grabación o actualización de datos necesitan realizarse en varias ocasiones.
- ✓ **Gasto extra de espacio de almacenamiento**: al estar repetidos, los datos ocupan mayor cantidad de espacio en el medio de almacenamiento. Cuanto mayor sea la *BD*, más patente se hará este problema.
- ✓ **Inconsistencia**: se produce cuando los datos que están repetidos, no contienen los mismos valores. Es decir, se ha actualizado su valor en un lugar y en otro no, por lo que no se sabría qué dato es válido y cual erróneo.

Para que una *BD* funcione óptimamente, hay que empezar realizando un buen diseño de ella. Es imprescindible que los diagramas *E/R* controlen la redundancia y, para ello, se debe analizar el esquema y valorar qué elementos pueden estar incorporando redundancia a la solución.

¿Dónde buscar indicios de redundancia en los esquemas? Existen **lugares y elementos que podrían presentar redundancia, por ejemplo**:

- ✓ **Atributos redundantes cuyo contenido se calcula en función de otros**. Un atributo derivado puede ser origen de redundancia.
- ✓ **Varias entidades unidas circularmente a través de varias relaciones**, es lo que se conoce como **un ciclo**. En caso de existir un ciclo, se deben tener en cuenta las siguientes condiciones, antes de poder eliminar dicha relación redundante:
  - Que el significado de las relaciones que componen el ciclo sea el mismo.
  - Que, si se elimina la relación redundante, el significado del resto de relaciones es el mismo.
  - Que, si la relación eliminada tenía atributos asociados, éstos puedan ser asignados a alguna entidad participante en el esquema, sin que se pierda su significado.

Pero hay que tener en cuenta que **no siempre que exista un ciclo se estará ante una redundancia**. Es necesario analizar detenidamente dicho ciclo para determinar si realmente existe o no redundancia.

Para finalizar, una apreciación. **No toda redundancia es perjudicial**. Existen ciertas circunstancias y condiciones en las que es conveniente (sobre todo a efectos de rendimiento) introducir cierta **redundancia controlada** en una *BD*. Ej.: si el método de cálculo del valor de un determinado atributo derivado es complejo y ralentiza el funcionamiento de la *BD*, quizá sea conveniente definir dicho atributo desde el principio y no considerarlo como un atributo redundante. La incorporación o no de redundancia controlada dependerá de la elección que haga el diseñador.

4.5 Propiedades deseables de un diagrama *Entidad/Relación*.

Cuando se construye un diagrama *Entidad/Relación* existen una serie de propiedades o características que éste debería cumplir. **Quizá no se materialicen todas, pero se ha de intentar cubrir la gran mayoría de ellas**. De este modo, se consigue que los diagramas o esquemas conceptuales tengan mayor calidad.

Estas **características o propiedades deseables** son:

- ✓ **Completitud**: si **es posible verificar que cada uno de los requerimientos está representado en dicho diagrama** y viceversa, cada representación del diagrama tiene su equivalente en los requerimientos.
- ✓ **Corrección**: **si se emplean de manera adecuada todos los elementos del modelo *Entidad/Relación***. La corrección de un diagrama puede analizarse desde dos vertientes:
  - **Corrección sintáctica**: cuando no se produzcan representaciones erróneas en el diagrama.

- **Corrección semántica**: cuando las representaciones signifiquen exactamente lo que está estipulado en los requerimientos. Posibles errores semánticos serían: la utilización de un atributo en lugar de una entidad, el uso de una entidad en lugar de una relación, utilizar el mismo identificador para dos entidades o dos relaciones, indicar erróneamente alguna cardinalidad u omitirla, etc.

- ✓ **Minimalidad**: **si se puede verificar que, al eliminar algún concepto presente en el diagrama, se pierde información**. Si un diagrama es redundante, no será mínimo.
- ✓ **Sencillez**: **si representa los requerimientos de manera fácil de comprender**, sin artificios complejos.
- ✓ **Legibilidad**: **si puede interpretarse fácilmente**. La legibilidad de un diagrama dependerá en gran medida del modo en que se disponen los diferentes elementos e interconexiones. Esta propiedad tiene mucho que ver con aspectos estéticos del diagrama.
- ✓ **Escalabilidad**: **si es capaz de incorporar posibles cambios derivados de nuevos requerimientos**.

Ejercicio resuelto

Si en un diagrama *E/R* se asocia un atributo a una entidad, pero este atributo debe asociarse realmente a una relación en la que interviene dicha entidad, se estaría incumpliendo la propiedad de:

Solución: Corrección semántica → La corrección semántica se centra en analizar si cada representación del diagrama significa exactamente lo mismo que lo estipulado por el documento de especificación de requerimientos.

Ejercicio resuelto

Realizar el diagrama de estructura de datos en el modelo E/R. Suponer que en un centro educativo se imparten muchos cursos. Cada curso está formado por un grupo de alumnos, de los cuales uno de ellos es el delegado del grupo. Los alumnos cursan asignaturas, y una asignatura puede o no ser cursada por los alumnos.

Lo primero que hay que hacer es identificar las entidades, luego las relaciones y las cardinalidades y, por último, los atributos de las entidades y de las relaciones (si los hubiera).

1. Identificación de entidades.

Una entidad es un objeto del mundo real, algo que tiene interés para la empresa. Se hace un análisis del enunciado, de donde se sacan los candidatos a entidades: *CENTRO*, *CURSO*, *ALUMNO*, *ASIGNATURA* y *DELEGADO*. Si se analiza esta última se ve que los delegados son alumnos, por lo tanto, se tienen recogidos en *ALUMNO*. Esta posible entidad se eliminará. También se eliminará la posible entidad *CENTRO* pues se trata de un único centro, si se tratara de una gestión de centros tendría sentido incluirla.

2. Identificar las relaciones.

Construir una matriz de entidades en la que las filas y las columnas son los nombres de entidades y cada celda puede contener o no la relación, las relaciones aparecen explícitamente en el enunciado. En este ejemplo, las relaciones no tienen atributos. Del enunciado se saca lo siguiente:

- ✓ Un curso está formado por muchos alumnos. La relación entre estas dos entidades la llamaremos *pertenece*, pues a un curso pertenecen muchos alumnos, relación 1:N. Consideraremos que es obligatorio que existan alumnos en un curso. Para calcular los máximos y mínimos hacemos la pregunta: a un curso, ¿cuántos alumnos pertenecen, como mínimo y como máximo? Y se ponen los valores en la entidad *ALUMNO*, en este caso (1,n). Para el sentido contrario, hacemos lo mismo: un alumno, ¿a cuántos cursos va a pertenecer? Como mínimo a 1, y como máximo a 1, en este caso pondremos (1,1) en la entidad *CURSO*.
- ✓ De los alumnos que pertenecen a un grupo, uno de ellos es delegado. Hay una relación de grado 1 entre la entidad *ALUMNO* que la podemos llamar *es_delegado*. La relación es 1:N, un alumno es delegado de muchos alumnos. Para calcular los valores máximos y mínimos preguntamos: ¿un alumno de cuántos alumnos es delegado? Como mínimo es 0, pues puede que no sea delegado, y como máximo es N, pues si es delegado lo será de muchos; pondremos en el extremo (0,n). Y en el otro extremo pondremos (1,1), pues obligatoriamente el delegado es un alumno.
- ✓ Entre *ALUMNO* y *ASIGNATURA* surge una relación N:M, pues un alumno cursa muchas asignaturas y una asignatura es cursada por muchos alumnos. La relación se llamará *cursa*. Consideramos que puede haber

asignaturas sin alumnos. Las cardinalidades serán (1,n) entre alumno-asignatura, pues un alumno, como mínimo, cursa una asignatura, y, como máximo muchas. La cardinalidad entre asignatura-alumno será (0,n), pues una asignatura puede ser cursada por 0 alumnos o por muchos.

En la tabla siguiente se muestra la matriz de entidades y relaciones entre ellas:

	CURSO	*ALUMNO*	*ASIGNATURA*
CURSO	----------	pertenece (1:N)	----------
ALUMNO	X	es_delegado (1:N)	cursa (N:M)
ASIGNATURA	----------	X	----------

Las celdas que aparecen con una X indican que las relaciones están ya identificadas. Las que aparecen con guiones indican que no existe relación.

3. Identificar los atributos.

Como en el enunciado no explicita ningún tipo de característica de las entidades nos imaginamos los atributos, que pueden ser los siguientes:

- ✓ *CURSO*: *idCurso* (clave primaria), *descripción*, *nivel*, *turno* y *etapa*.
- ✓ *ALUMNO*: *idMatrícula* (clave primaria), *nombre*, *dirección*, *población*, *teléfono* y *numHermanos*.
- ✓ *ASIGNATURA*: *idAsignatura* (clave primaria), *denominación* y *tipo*.

Diagrama de estructura de datos resultante en el modelo *E/R* (notación de *Chen*), creado con *DIA*:

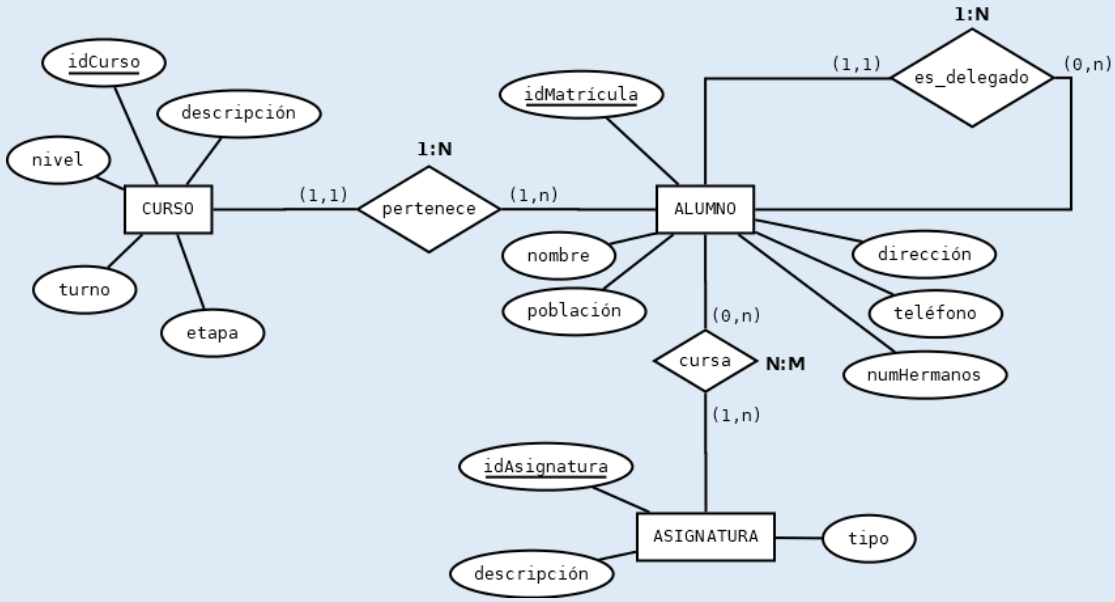
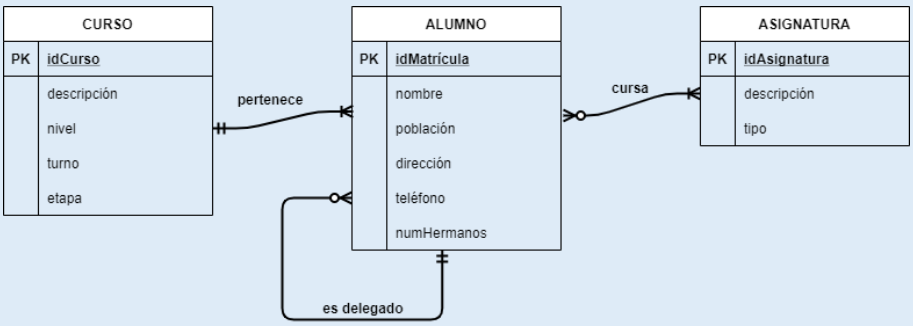


Diagrama de estructura de datos resultante en el modelo *E/R* (notación de *Martin* y *patas de gallo*), creado con el editor online http://draw.io/ :

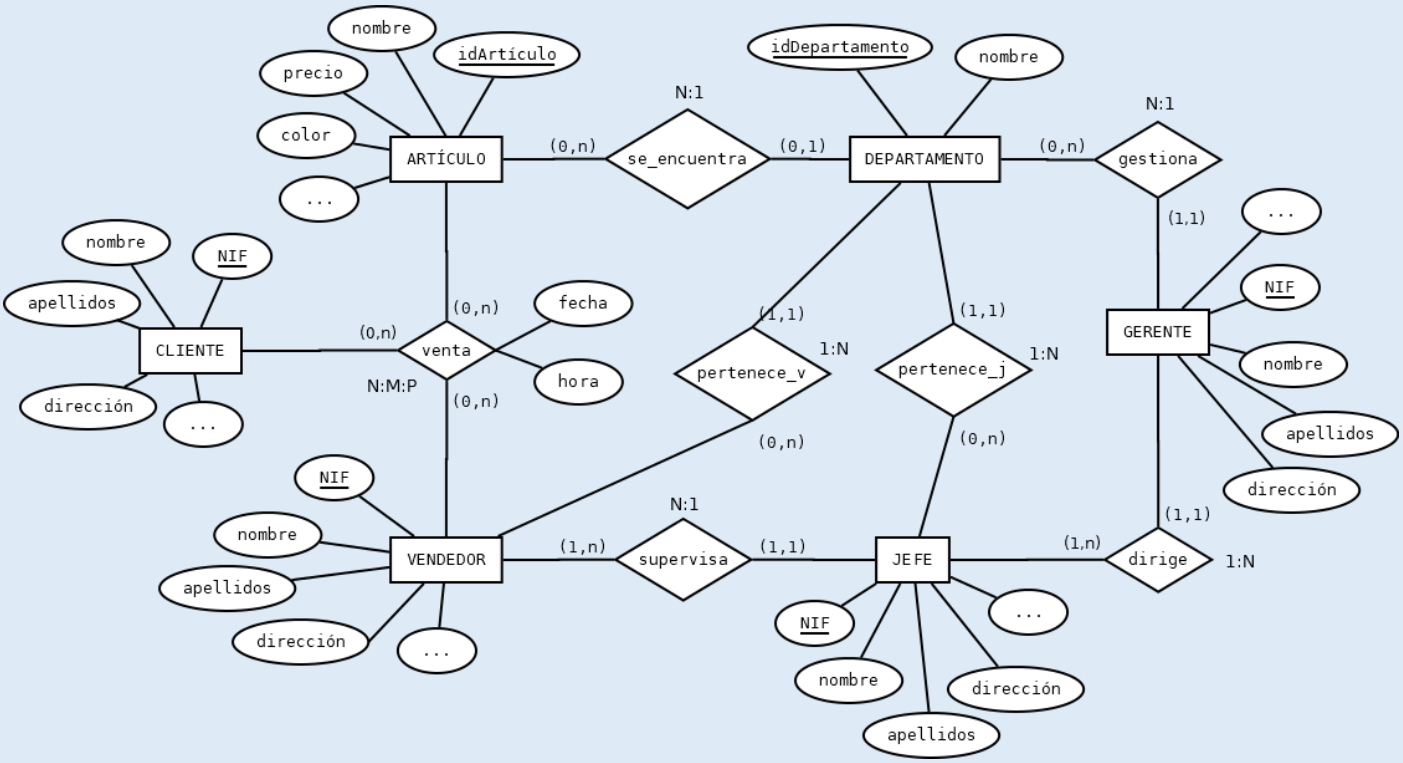


Ejercicio resuelto

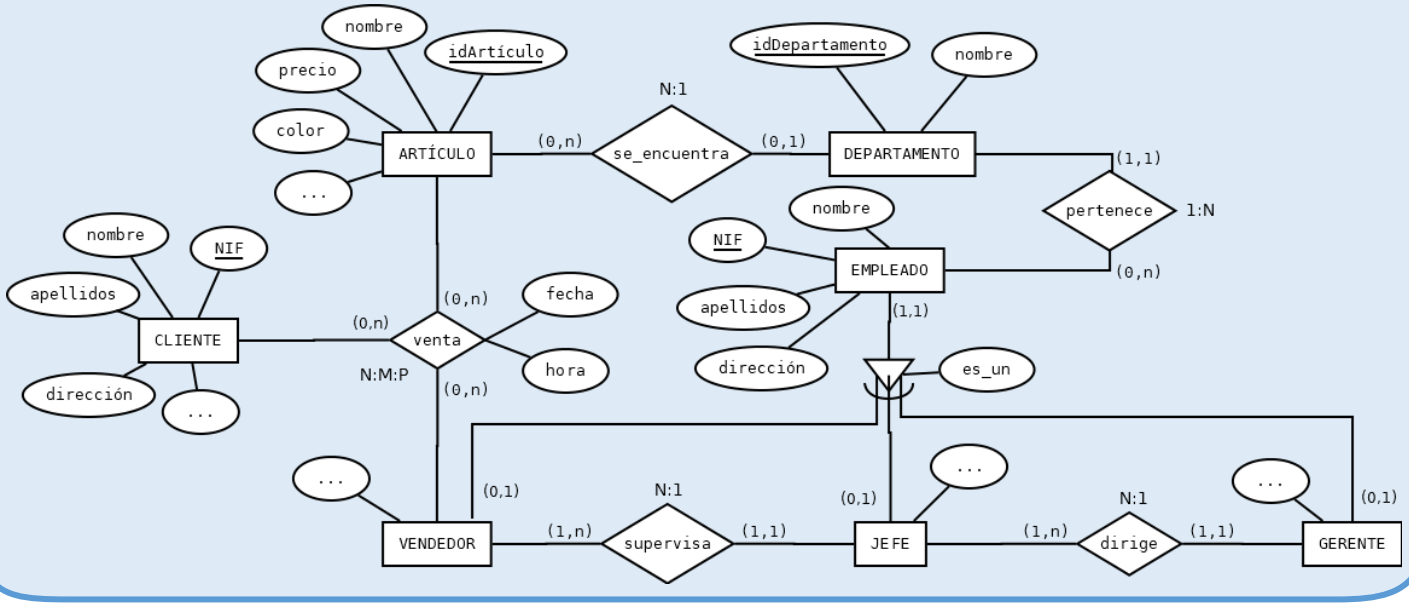
Se desea diseñar una *BD* para un centro comercial organizado por departamentos que contenga información sobre los clientes que han comprado algo, los trabajadores, el género que se oferta y las ventas realizadas. Construir el modelo *E/R* teniendo en cuenta las siguientes restricciones:

- ✓ **Existen tres tipos de trabajadores: gerentes, jefes y vendedores.**

- ✓ **Cada departamento está gestionado por un gerente.**
- ✓ **Un determinado producto sólo se encuentra en un departamento.**
- ✓ **Los jefes y vendedores sólo pueden pertenecer a un único departamento.**
- ✓ **Un gerente tiene a su cargo a un cierto número de jefes y estos a su vez a un cierto número de vendedores.**
- ✓ **Una venta la realiza un vendedor a un cliente y debe quedar constancia del artículo vendido. Sólo un artículo por apunte de venta.**



Otra posible solución haciendo uso del modelo *E/R Extendido*:



TAREA - DIAGRAMAS ENTIDAD/RELACIÓN

Realizar el diagrama de estructuras de datos en el modelo *E/R*, correspondiente de los siguientes enunciados:

1. **Suponer el bibliobús que proporciona un servicio de préstamo de libros a los socios de un pueblo. Los libros están clasificados por temas. Un tema puede contener varios libros. Un libro es prestado a muchos socios, y un socio puede coger varios libros. En el préstamo de libros es importante saber la fecha de préstamo y la fecha**

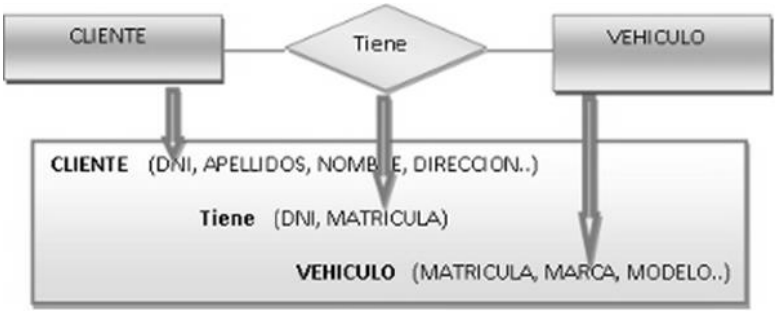
de devolución. De los libros interesa saber el título, el autor y el número de ejemplares.

2. **En una estación de autobuses se cuenta con unos autobuses que recorren una serie de lugares y que son conducidos por varios conductores. Se quiere representar los lugares que son recorridos por cada autobús, conducidos por cada conductor y la fecha en la que se visita el lugar.**
   Define las entidades, los posibles atributos, identifica los atributos clave y los datos importantes para la relación entre las entidades. Indica también la cardinalidad.
3. **Una compañía de distribución de productos para el hogar dispone de proveedores que le suministran artículos. Un artículo sólo puede proveerlo un proveedor.**
   La empresa tiene tres tipos de empleados: oficinistas, transportistas y vendedores. Estos últimos venden los artículos. Un artículo es vendido por varios vendedores, y un vendedor puede vender varios artículos en distintas zonas de venta. De las ventas nos interesa saber la fecha de venta y las unidades vendidas.
4. **La *Consejería de Educación* gestiona varios tipos de centros: públicos, privados y concertados. Los privados tienen un atributo específico que es la cuota y los concertados la agrupación y la comisión. También asigna plazas a los profesores de la comunidad para impartir clase en esos centros. Un profesor puede impartir clase en varios centros.**
   Define las entidades, los posibles atributos, identifica los atributos clave y los datos importantes para la relación entre las entidades. Indica también la cardinalidad.
5. **A un taller de automóviles llegan clientes a comprar coches. De los coches nos interesa saber la marca, el modelo, el color y el número de bastidor.**
   Los coches pueden ser nuevos y de segunda mano. De los nuevos nos interesa saber las unidades que hay en el taller. De los viejos el año de fabricación, el número de averías y la matrícula.
   Los mecánicos se encargan de poner a punto los coches usados del taller. Un mecánico pone a punto a varios coches usados.
   Un cliente puede comprar varios coches; un coche puede ser comprado por varios clientes. De la compra nos interesa la fecha y el precio.
   Define las entidades, los atributos, las relaciones, sus atributos si los hubiera y las cardinalidades.
6. **Una agencia de viajes está formada por varias oficinas que se ocupan de atender a los posibles viajeros. Cada oficina oferta un gran número de viajes. Los viajes trabajan con una serie de destinos y una serie de procedencias. Cada viaje tiene un único destino y una única procedencia. Sin embargo, un destino puede ser objetivo de varios viajes y una procedencia ser punto de partida de varios viajes. Cada viaje tiene muchos viajeros.**
7. **Una entidad bancaria está formada por varias sucursales y cada sucursal tiene un gran número de cuentas que son propiedad de los clientes. Los datos saldo, debe y haber deben aparecer en cada una de las cuentas. Las cuentas son de dos tipos: cuenta de ahorro con el atributo específico tipo de interés y cuenta corriente, con el atributo específico cantidad de descubierto. Las cuentas o son corrientes o son de ahorro.**
   Un cliente puede tener varias cuentas. Una cuenta es sólo propiedad de un cliente.
   Las cuentas realizan una serie de movimientos, en los que además de otros datos deben aparecer la cantidad implicada y la fecha. Existe una serie de tipos de movimientos reconocidos por el banco. Un movimiento pertenece a un tipo. Sin embargo, de un tipo de movimiento puede haber varios movimientos.
8. **Realiza al menos tres de los ejercicios propuestos en el anexo I de *Ejercicios Modelo Entidad/Relación*.**
9. **Realiza al menos tres de los ejercicios propuestos en el anexo I de *Ejercicios Modelo Entidad/Relación Extendido*.**

5 Modelo relacional.

La **fase de diseño conceptual se materializa en el diagrama *E/R***, y permite una gran **independencia de las cuestiones relativas a la implementación física** de la *BD*. El **tipo de *SGBD*, las herramientas software, las aplicaciones, lenguajes de programación o hardware disponible no afectarán**, al menos hasta el momento, a los resultados de esta fase.

El **esquema conceptual habrá sido revisado, modificado y probado para verificar que se cumplen** adecuadamente todos y cada uno de **los requerimientos del problema a modelar**. Este esquema representará el **punto de partida para** la siguiente fase, el **diseño lógico de la *BD***.



El diseño lógico **consistirá en la construcción de un esquema de la información relativa al problema, basado en un modelo de *BD* concreto**. El esquema conceptual se transformará en un **esquema lógico** que utilizará los elementos y características del modelo de datos en el que esté basado el *SGBD*, para implementar la *BD*. Como se pudo ver anteriormente, estos modelos podrán ser: el modelo en red, el modelo jerárquico y, sobre todo, el modelo relacional y el modelo orientado a objetos.

En este caso, **tomando como referencia el esquema *E/R*, se realizará el paso de éste al modelo de datos relacional**. Esta transformación requerirá de la aplicación de determinadas reglas y condiciones que garanticen la equivalencia entre el esquema conceptual y el esquema lógico.

Como **paso posterior**, **sobre la información del esquema lógico obtenido**, será necesario llevar a cabo un proceso que permitirá diseñar de forma correcta la estructura lógica de los datos. Este proceso recibe el nombre de **normalización**, que se conforma como un **conjunto de técnicas que permiten validar esquemas lógicos basados en el modelo relacional**.

Entonces, ¿qué pasos son los siguientes a dar? Resumiendo, un poco, el diagrama *E/R*, se transformará al modelo relacional, se aplicará la normalización y se obtendrá lo que se conoce en el argot como el paso a tablas del esquema conceptual o, lo que es lo mismo, el esquema lógico. Desde ese momento, basándonos en este esquema, se podrá llevar la *BD* a cualquier *SGBD* basado en el modelo relacional e implementarla físicamente. Esta implementación física será totalmente dependiente de las características del *SGBD* elegido.

5.1 El modelo relacional.

*Edgar Frank **Codd*** es el padre del modelo relacional y lo definió en un artículo publicado en 1970. Este **modelo está basado en dos teorías matemáticas: la teoría de conjuntos y la lógica de predicados de primer orden**. Pero no es necesario aprender estas teorías para poder utilizar el modelo relacional.

Este modelo persigue, al igual que la mayoría de los modelos de datos, los siguientes **objetivos**:

- ✓ **Independencia física de los datos**: el modo de almacenamiento de los datos no debe influir en su manipulación lógica.
- ✓ **Independencia lógica de los datos**: los cambios que se realicen en los objetos de la *BD* no deben repercutir en los programas y usuarios que accedan a ella.
- ✓ **Flexibilidad**: presentar los datos a los usuarios de la forma más adecuada a la aplicación que utilicen.
- ✓ **Uniformidad**: presentación de las estructuras lógicas en forma de tablas, fáciles de comprender y manipular por los usuarios.
- ✓ **Sencillez**: las características anteriores, junto con los lenguajes de usuario hacen que este modelo sea fácil de entender y utilizar por el usuario.

Para conseguir estos objetivos, *Codd* introduce el concepto de **relación (tabla)** como **estructura básica del modelo**. Todos los datos de una *BD* se representan en forma de relaciones cuyo contenido varía con el tiempo. Los **conceptos básicos de este modelo** son:

- ✓ **Interrelación** es la **asociación entre dos tablas**.
- ✓ El **conjunto de columnas representa las propiedades de la tabla** y se denominan **atributos**.
- ✓ El conjunto de las **filas se denomina tuplas** y contienen los **valores** que toman cada uno de los atributos para cada elemento de la relación.

Para *Codd* lo importante es el diseño lógico, un modelo abstracto, por lo que no indica nada acerca de la

implementación de este modelo en los *SGBD*. Sin embargo, los trabajos de *Codd* y otros investigadores dieron lugar a diversas bases de datos comerciales y es el modelo lógico en el que se basan la mayoría de los *SGBD* existentes actualmente en el mercado.

Suele haber confusión entre la terminología que utilizan las *BD* y los sistemas de archivos. En ocasiones se encuentra que se hace referencia a las filas como registros, a las columnas como campos y a las tablas como archivos. La diferencia está en que la tabla en una *BD* es un concepto lógico, mientras que los conceptos archivo, registro y campo son conceptos físicos.

En 1985, ***Codd*** publica sus **famosas doce reglas que debe cumplir cualquier *BD* para ser considerada relacional**:

- ✓ **Regla de información**: toda información de una *BD* relacional está **representada mediante valores en tablas**.
- ✓ **Regla de acceso garantizado**: se garantiza que todos **los datos** de una *BD* relacional son lógicamente **accesibles a través de una combinación de nombre de tabla, valor de clave primaria y nombre de columna**.
- ✓ **Tratamiento sistemático de valores nulos**: los valores nulos se soportan en los *SGBD* para representar la **falta de información** de un modo sistemático e independiente de los tipos de datos.
- ✓ **Catálogo en línea dinámico basado en el modelo relacional**: la **descripción de la *BD* se representa en el ámbito lógico de la misma forma que los datos ordinarios**.
- ✓ **Regla de sublenguaje completo de datos**: un sistema relacional puede soportar **varios lenguajes y varios modos de uso** terminal.
- ✓ **Regla de actualización de vista**: todas las vistas, que sean teóricamente actualizables, son también actualizables por el sistema.
- ✓ **Inserción, actualización y supresión de alto nivel**.
- ✓ **Independencia física de datos**.
- ✓ **Independencia lógica de los datos**.
- ✓ **Independencia de la integridad**: las restricciones de integridad específicas de una *BD* relacional deben ser definidas mediante el sublenguaje de datos relacional y almacenarse en el catálogo de la *BD*.
- ✓ **Independencia de la distribución**: un *SGBD* es independiente de la distribución.
- ✓ **Regla de no subversión**: **si un *SGBD* tiene un lenguaje de bajo nivel no se puede utilizar para destruir o evitar las reglas de integridad o las restricciones expresadas en el lenguaje relacional de alto nivel**.

El **modelo relacional propone una representación de la información que**:

- ✓ Origine **esquemas que representen fielmente la información**, los objetos y las relaciones.
- ✓ Sea **fácilmente entendida** por los usuarios.
- ✓ Sea **posible ampliar el esquema de la *BD* sin modificar la estructura lógica** existente **y los programas** de aplicación.
- ✓ Permita la **máxima flexibilidad en la formulación de los interrogantes sobre la información** mantenida en la *BD*.

5.2 Estructura y características.

El modelo relacional se representa gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a campos o atributos de esos registros.

Se han introducido algunos **elementos que intervienen en el modelo relacional**. De forma más detallada son:

- ✓ Una **relación** es una **tabla con columnas y filas**. El usuario percibe la ***BD* como un conjunto de tablas**. Una tabla sirve para almacenar los datos de una entidad.
- ✓ Un **atributo** es el **nombre de una columna de una relación**. Ejemplo: atributo nombre definido sobre el dominio conjunto de 15 caracteres.
- ✓ Una **tupla** es **cada fila de una relación** (o tabla). El **orden de las tuplas no es relevante**.
- ✓ El **grado** de una relación es el **número de atributos que contiene**, es decir el número de columnas de la tabla.
- ✓ La **cardinalidad** de una relación es el **número de tuplas que contiene**.
- ✓ El **valor** es la **intersección entre una fila y una columna**.

Una **tabla** tiene una serie de **características**:

- ✓ Una tabla se percibe como una **estructura bidimensional compuesta de filas y columnas**.

- ✓ Cada tabla o relación **tiene un nombre que la diferencia** de las demás.
- ✓ Cada **fila de la tabla se denomina tupla** y representa la ocurrencia de una entidad.
- ✓ **No admiten filas duplicadas**.
- ✓ El **orden de las filas no es significativo**.
- ✓ Cada **columna representa un atributo** y **tiene un nombre distinto**.
- ✓ El **orden de las columnas es irrelevante**. No están ordenadas.
- ✓ La tabla **es plana**, es decir, **en la intersección fila/columna solo puede haber un valor**, **no se admiten atributos multivaluados**.
- ✓ Cada tabla **debe tener un atributo o conjunto de atributos que identifique a cada fila de forma única**.
- ✓ **Cada columna tiene un mismo tipo de datos** y un intervalo de valores específico: dominio de un atributo.

Ej.: Relación *EMPLEADO*.

ATRIBUTOS

EMPLEADOS

IDEMPLEADO	NIF	APELLIDOS	NOMBRE	FECHA_NACIMIENTO	DIRECCION	CODIGO_POSTAL	TELEFONO	CARGO
EM001	23145678R	Ceballos	Jose	25/06/1971	Pamanes, 172	39521	900100010	Comercial
EM002	12345689D	Arzeno	Claudio	15/03/1968	Ruiz Zorrilla,12	29647	942123456	Peon de almacen
EM003	15935782L	Gonzalez	Miguel	14/02/1970	Paseo Pereda, 21	39425	684236547	Administrativo
EM004	18564789G	De Arco	Juana	31/12/1982	Perez Galdos, s/n	39698	670125478	Comercial
EM005	35235255H	Peña	Agustín	12/06/1975	Leonardo Rubalcaba, 11	39786	942653463	Jefe Taller
EM006	345353571	Díaz	Helena	22/02/1968	Sta Lucía, 34	39676	636252352	Técnico
EM007	34523678G	Mantecón	Francisco	09/09/1970	Gral Alonso Vega, 66	39008	627634631	Técnico
EM008	67678999R	Pereda	Antonio	18/03/1965	Vía Cornelia, 5	39789	942653463	Técnico

TUPLAS

RELACIÓN: EMPLEADOS
GRADO: 9
CARDINALIAD: 8

La visualización desde un punto de vista lógico de una *BD* relacional se corresponde con un conjunto de tablas relacionadas. Para el usuario una tabla es un conjunto de entidades.

5.3 Restricciones del modelo relacional.

En todos los modelos de datos existen restricciones que deben tenerse en cuenta a la hora de diseñar una *BD*. Los datos almacenados en la *BD* deben cumplir con una serie de reglas para garantizar que sean correctos. El modelo relacional, como todo modelo de datos, también presenta ocurrencias no permitidas. Los tipos de restricciones pueden ser:

1. **Restricciones inherentes al modelo**: son restricciones **propias del modelo** e **indican las características propias de una relación**, por tanto, **han de cumplirse obligatoriamente**.
   - ✓ Ausencia de tuplas repetidas.
   - ✓ Irrelevancia del orden de las tuplas.
   - ✓ Irrelevancia del orden de los atributos.
   - ✓ Cada atributo solo puede tomar un único valor del dominio al que pertenece.

2. **Restricciones semánticas o de usuario**: son **impuestas para cada problema concreto, desde la definición de los dominios de un campo a condiciones impuestas a un campo de acuerdo con el valor de otros**. Para establecer estas condiciones se dispone de las siguientes restricciones:
   - ✓ La **restricción de clave primaria** (***PRIMARY KEY***): permite **declarar uno o varios atributos como clave primaria de una relación**. Ej.: la clave principal de la relación *CLIENTE* sería el *codCliente*.
   - ✓ **Restricciones de verificación** (***CHECK***): **comprueban en una actualización si se cumplen las condiciones exigidas en la restricción** o no, **antes de ejecutarla**. Ej.: la edad de un trabajador sea mayor o igual a 16.
   - ✓ **Restricción de unicidad** (***UNIQUE***): permite **definir claves alternativas**. Los **valores de los atributos no pueden repetirse**. Ej.: en la tabla *CLIENTE* la clave principal seleccionada es el *codCliente*, pero el atributo *NIF*, que es clave alternativa en dicha tabla, tampoco admite valores duplicados. Para ello se establece una restricción de tipo *UNIQUE* para este campo.
   - ✓ **Restricción de obligatoriedad** (***NOT NULL***): permite **declarar si uno o varios atributos pueden tomar**

valores nulos. Por definición una **clave principal no puede contener valores nulos** en ninguno de los atributos que la componen (integridad de clave) por tanto se definirán como *NOT NULL*. Ej.: la restricción *NOT NULL* puede ser necesaria también con otro tipo de atributos como el *NIF* citado en el ejemplo anterior, el nombre, etc.

- ✓ Los **disparadores (*TRIGGER*)** son **acciones**, métodos según la terminología orientada a objetos, **que ejecuta un objeto, como respuesta a un evento o acción que se realiza sobre él**, cuando se cumple una determinada condición.
  Ej.: si se tienen dos tablas, una con los datos de los vehículos de los clientes (matrícula y resto de datos que identifican al vehículo) y otra que recoge cada una de las entradas en el taller de esos vehículos (matrícula, fecha y hora), se puede crear un *trigger* que cada vez que se inserte un vehículo en la tabla de vehículos añada una fila en la tabla entradas en taller con los datos anteriores.

3. **Otras restricciones** inherentes son:
   - ✓ **Integridad de clave**: es una restricción que exige que **todos los atributos de la clave primaria** (*PRIMARY KEY*), **han de contener valores no nulos** (*NOT NULL*).
     En el ejemplo de la relación *FACTURA-LÍNEA*, la clave principal de la tabla *LÍNEA* es una clave compuesta de los campos *NumFactura+NumLinea*. Esta restricción implica que ambos atributos deben contener valores que no sean nulos.
   - ✓ **Integridad referencial (*FOREIGN KEY*)**: consiste en que **no puede haber un valor en una clave ajena de una tabla, si antes no existe en la tabla de la que ese campo o conjunto de campos formen la clave primaria**.
     Para explicar este concepto se denominará tabla hija a la tabla que contiene la clave ajena y tabla padre a la tabla que contiene la clave principal.
     Se permiten valores nulos en las claves ajenas; es decir que una clave ajena debe coincidir con un valor de clave primaria de la relación a la que apunta o tener valor nulo.
     Por ejemplo: para relacionar un vehículo con su propietario se establece una clave ajena (*CodCliente*) en la tabla *VEHÍCULO* que coincide con el *CodCliente* correspondiente en la tabla *CLIENTE*. La integridad referencial establece que no podemos introducir ningún código de cliente (clave ajena) en la tabla (*VEHICULO*) si no existe previamente ese código en la tabla *CLIENTE* (clave principal).

5.3.1 Restricciones y operaciones relacionales.

Además de definir las claves ajenas hay que tener en cuenta las operaciones de borrado y actualización que se realizan sobre las filas de la tabla relacionada.

A la hora de hacer operaciones de inserción, actualización o borrado aparecen problemas, ya que hay que tener en cuenta los tres tipos de restricciones establecidas, sobre todo las restricciones de clave primaria e integridad de clave.

Respecto a la **actualización de campos** hay que tener en cuenta:

- ✓ No pueden modificarse los atributos que forman parte de la clave primaria para dar origen a registros con **valores de clave primaria redundantes**.
- ✓ La modificación de atributos que forman la **clave principal** no puede dar lugar a que aparezcan en esos atributos **valores nulos**.
- ✓ Para aceptar los cambios en un atributo, se han de **cumplir las restricciones de clave, referenciales y de usuario** que hayan sido definidas para ese atributo en la tabla a la que pertenezca.
- ✓ Si se modifica el valor de un atributo que forme parte de la **clave primaria**, habrá que **sustituir los valores de los atributos que forman claves ajenas**, en los que aparece el valor antes de modificar el contenido del atributo, por su nuevo valor. Es lo que se denomina actualización en cascada.

Respecto de la de **inserción de campos**:

- ✓ **No pueden añadirse registros** si alguno de los atributos que forman parte de la **clave primaria** posee un **valor nulo**.
- ✓ **No pueden añadirse registros** si el conjunto de valores de los atributos que forman parte de la clave primaria origina una **clave primaria redundante**.

- ✓ Como consecuencia de la actualización en cascada, **para añadir un valor a un atributo que forma parte de la clave ajena, ese valor ha de existir en la clave primaria a la que hace referencia la clave ajena**.
- ✓ Para aceptar los cambios en un atributo **se han de cumplir las restricciones de clave, referenciales y de usuario** que hayan sido definidas para ese atributo en la tabla a la que pertenezca.

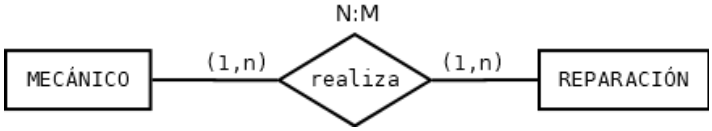
Respecto del **borrado de campos**:

- ✓ Al **borrar una fila que tenga asociados, mediante la clave primaria, otras filas en otras tablas**, se pierde la consistencia de la tabla. Para solucionarlo existen dos posibilidades:
  - **Borrado en cascada**: consiste en borrar todas las filas de las tablas en los que aparezca como clave ajena el conjunto de atributos que forman la clave primaria del registro a borrar.
  - Para borrar un registro cuya clave primaria es clave ajena en otras tablas, primero hay que **borrar los registros con la clave primaria asociada, para después borrar el registro deseado**.
- ✓ Teniendo en cuenta las consideraciones anteriores las **posibilidades** son las siguientes:
  - **Borrado y/o modificación en cascada** (***CASCADE***): el borrado o modificación de una fila en la relación padre (relación con la clave primaria) ocasiona un borrado o modificación de las filas relacionadas en la relación hija (relación que contiene la clave ajena).
  - **Borrado y/o modificación restringida** (***RESTRICT***): en este caso no es posible realizar el borrado o la modificación de las filas de la relación padre si existen las relacionadas en la relación hija.
  - **Borrado y/o modificación con puesta a nulos** (***SET NULL***): esta restricción permite poner la clave ajena en la tabla referenciada a *NULL*, si se produce el borrado o modificación en la tabla primaria o padre.
  - **Borrado y/o modificación con puesta a valor por defecto** (***SET DEFAULT***): el valor que se pone en las claves ajenas de la tabla referenciada es un valor por defecto que se habrá especificado en la creación de la tabla.

5.4 Paso del diagrama *Entidad/Relación* al modelo relacional.

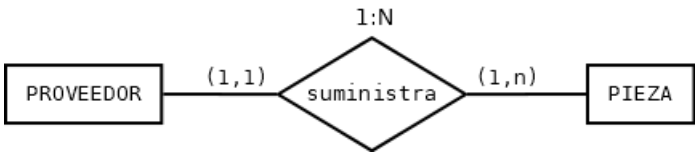
Una vez obtenido el esquema conceptual mediante el modelo *E/R*, hay que definir el modelo lógico de datos. Las **reglas básicas** para transformar un esquema conceptual *E/R* a un esquema relacional son las siguientes:

- ✓ Toda **entidad** se transforma **en una tabla**.
- ✓ Todo **atributo** se transforma **en una columna dentro de la tabla**.
- ✓ El **identificador único de la entidad** se convierte en **clave primaria**.
- ✓ Como las **relaciones** del modelo *E/R* no tienen equivalente en el modelo relacional, ya que sólo existen tablas y operaciones entre ellas, es necesario aplicar las siguientes reglas:
  - En las **relaciones *N:M*** se crea una **nueva tabla que tendrá como clave primaria la concatenación de los atributos clave de las entidades que asocia y con los atributos propios de la relación si los hay**. Esta tabla posee dos claves ajenas, una por cada entidad con la que está relacionada.



MECÁNICO (idMecánico, nombre, …)
REALIZA (idMecánico[fk: MECÁNICO], idReparación[fk: REPARACIÓN])
REPARACIÓN (idReparación, descripción, …)

  - En las **relaciones *1:N*** la **entidad del lado *N* de la relación añade el conjunto de campos necesarios para incorporar a sus atributos la totalidad de la clave primaria de la entidad del lado *1*, creando una clave ajena**, de modo que se puedan relacionar ambas tablas mediante operadores relacionales. El nombre de la relación desaparece.

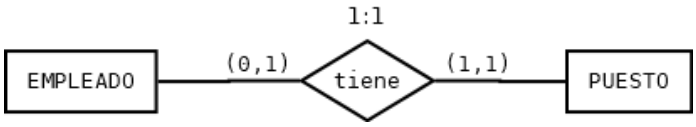


PROVEEDOR (idProveedor, nombre, …)
PIEZA (idPieza, descripción, …, idProveedor[fk: PROVEEDOR])

  - Las **relaciones *1:1*** se transforman en función de las cardinalidades:
    - **Propagando cualquiera de los atributos identificadores y sus atributos asociados creando una única tabla con el conjunto de los atributos de ambas entidades**. La clave primaria sería cualquiera

de las dos. Ambas entidades participan con cardinalidades (1,1).

- **Propagando la clave de cualquiera de las tablas a la otra, según cuál sea la que tenga accesos más frecuentes**. Ambas cardinalidades (1,1). Se crean dos tablas, una para cada entidad.
- **Propagar la clave de la entidad con cardinalidad (1,1) a la entidad que tenga (0,1)**.



EMPLEADO (idEmpleado, nombre, …, idPuesto[fk: EMPLEADO])
PUESTO (idPuesto, descripción, …)

- **Crear una nueva tabla a partir de la relación con las dos claves de ambas**, además de las tablas correspondientes a las entidades asociadas. Cuando ambas tablas tienen cardinalidades (0,1).

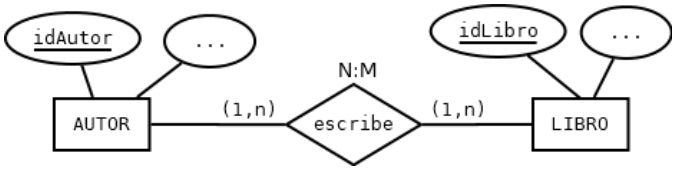
Además de las reglas de transformación anteriores que se aplican con carácter general, existen otros aspectos a tener en cuenta a la hora de obtener un esquema relacional a partir de un modelo *Entidad/Relación*.

5.4.1 Relaciones *N:M*.

TRANSFORMACIÓN DE INTERRELACIÓN *N:M*.

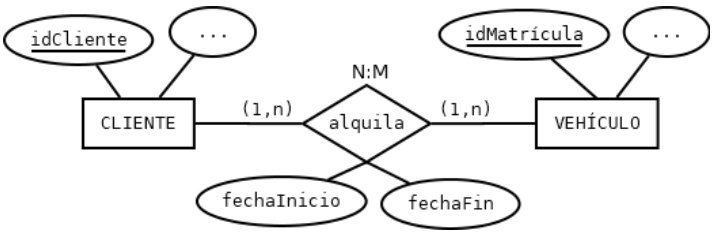
Regla general: **se transforman en una nueva tabla cuya clave se forma, al menos, con la concatenación de las claves de las entidades que participan en la relación**, que son además claves ajenas que referencian a las tablas en las que son claves primarias. El **nombre asignado a la tabla es el que tenía la relación**.

ESCRIBE (idAutor[fk: AUTOR], idLibro[fk: LIBRO])
AUTOR (idAutor, …)
LIBRO (idLibro, …)



TRANSFORMACION DE LA DIMENSION TEMPORAL.

En algunos **casos en los que la relación tenga atributos de tipo fecha, será necesario incluir al menos una fecha como parte del atributo identificador principal para recoger la dimensión temporal del modelo**. En otros casos la fecha puede ser una entidad más o solo un atributo.

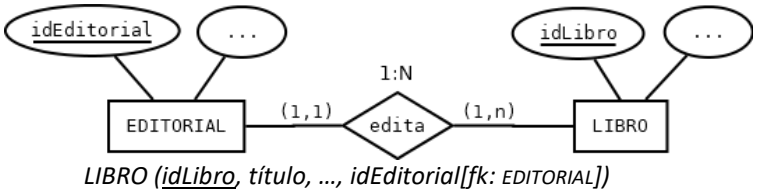


ALQUILA (idCliente[fk: CLIENTE], idMatrícula[fk: VEHÍCULO], fechaInicio, fechaFin)
CLIENTE (idCliente, …)
VEHÍCULO (idMatrícula, …)

5.4.2 Relaciones *1:N*.

TRANSFORMACION DE INTERRELACION *1:N*.

Como norma general **se propaga la clave de la entidad que tiene cardinalidad máxima *1* a la que tiene cardinalidad máxima *N***.

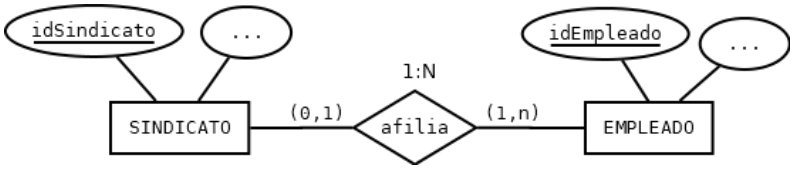


LIBRO (idLibro, título, …, idEditorial[fk: EDITORIAL])
EDITORIAL (idEditorial, nombre, …)

EXCEPCIONES:

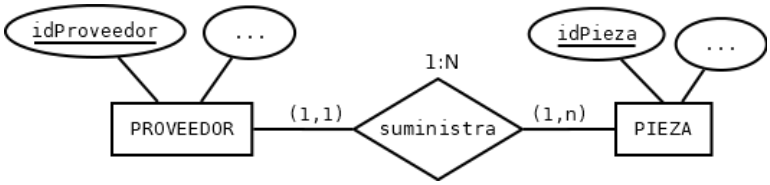
En los siguientes casos interesa más crear una nueva tabla a partir de la relación como en el caso de correspondencias *N:M*:

1) **Cuando el número de ocurrencias de la entidad que propaga la clave es muy pequeño y cabe la posibilidad de que al propagar la clave quedan muchos valores repetidos o nulos** (Ej.: cuando el número de empleados que estén afiliados a un sindicato es muy pequeño). Se tiene cardinalidad (0,1) en uno de los extremos.



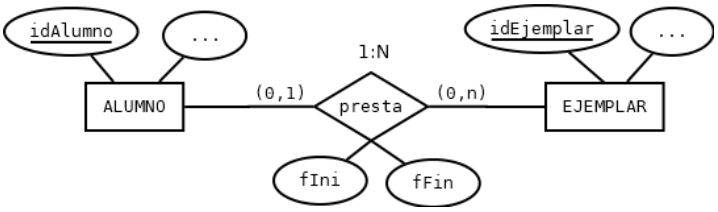
SINDICATO (idSindicato, nombre, …)
AFILIA (idSindicato[fk: SINDICATO], idEmpleado[fk: EMPLEADO])
EMPLEADO (idEmpleado, nombre, …)

2) **Cuando se prevea que en el futuro se puede convertir en una relación *N:M*** (Ej.: si en un futuro una misma pieza puede ser suministrada por distintos proveedores).



PROVEEDOR (idProveedor, nombre, …)
SUMINISTRA (idProveedor[fk: PROVEEDOR], idPieza[fk: PIEZA])
PIEZA (idPieza, descripción, …)

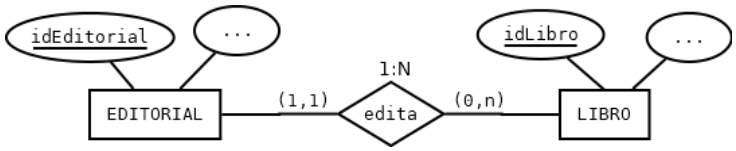
3) **Cuando la relación tenga atributos propios**. En algunos casos se pueden migrar estos atributos junto con la clave, pero en general, se crea una nueva tabla (Ej.: un alumno puede retirar varios ejemplares. No se recoge la dimensión temporal).



ALUMNO (idAlumno, nombre, …)
PRESTA (idAlumno[fk: ALUMNO], idEjemplar[fk: EJEMPLAR], fIni, fFin)
EJEMPLAR (idEjemplar, título, …)

CARDINALIDADES:

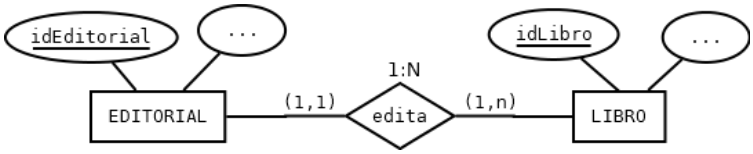
1) **Cuando la entidad que tiene cardinalidad máxima 1, tiene también 1 de cardinalidad mínima, tendremos que tener en cuenta al propagar la clave que en la tabla que recibe la clave, como clave extranjera, no pueda tener valores nulos**.



EDITORIAL (idEditorial, nombre, …)
LIBRO (idLibro, …, idEditorial[fk: EDITORIAL])

Propagación de clave (no nula)

2) **Cuando la entidad que tiene cardinalidad máxima n, tiene de cardinalidad mínima 1, tendremos que controlar por software que, al dar de alta una, la de la otra tabla se introduzca al menos una fila en esta**.

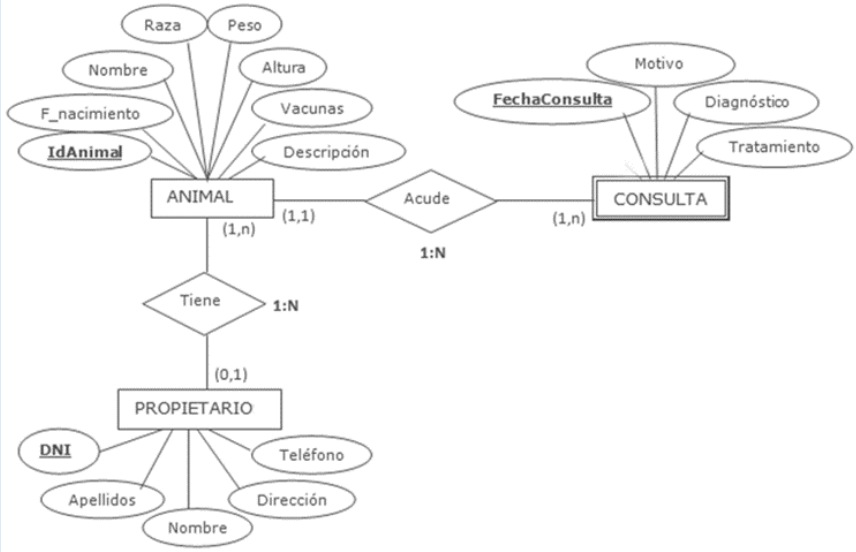


EDITORIAL (idEditorial, nombre, …)
LIBRO (idLibro, …, idEditorial[fk: EDITORIAL])

(no nula)

Ejercicio resuelto

Partiendo del diagrama *E/R* entre las entidades *PROPIETARIO*, *ANIMAL* Y *CONSULTA*. Se pide obtener el modelo relacional correspondiente.



Solución:

ANIMAL *(IdAnimal, F_nacimiento, Nombre, Raza, Peso, Altura, Vacunas, Descripción, DNI[fk: PROPIETARIO])*
CONSULTA *(IdAnimal[fk: ANIMAL], FechaConsulta, Motivo, Diagnóstico, Tratamiento)*
PROPIETARIO *(DNI, Apellidos, Nombre, Dirección, Teléfono)*

5.4.3 Relaciones *1:1*.

No hay una regla fija, puede optarse por la solución basada en:

- ✓ Recoger la mayor cantidad de semántica posible.
- ✓ Tener en cuenta las cardinalidades mínimas.
- ✓ Evitar los valores nulos.
- ✓ Motivos de eficiencia.

1) Cuando las **cardinalidades de ambas entidades son (1,1)** se pueden adoptar distintas soluciones:

   a) No se necesitan 2 tablas, se puede **crear una única tabla en la que se incluyan los atributos de las dos entidades**, cuya **clave principal será cualquiera de los atributos identificadores principales**.

      ESTAR_CASADO (idEsposo, …, idEsposa, …)

   b) **Propagar la clave de cualquiera de ellas a la otra tabla, teniendo en cuenta a cuál de ellas se le efectúan los accesos más frecuentes**.

      ESPOSO (idEsposo, …, idEsposa[fk: ESPOSA])
      ESPOSA (idEsposa, …)

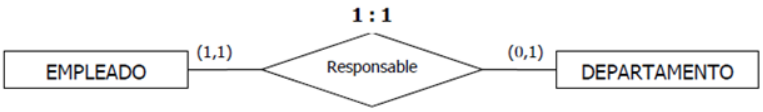
   c) **Propagar las dos claves, introduce redundancias que se controlarán mediante restricciones**.

      ESPOSO (idEsposo, …, idEsposa[fk: ESPOSA])
      ESPOSA (idEsposa, …, idEsposo[fk: ESPOSO])

2) Cuando las **cardinalidades de ambas entidades son (0,1)**, la **relación se transforma en una tabla para evitar los valores nulos**.

   MATRIMONIO (idHombre[fk: HOMBRE], idMujer[fk: MUJER])
   HOMBRE (idHombre, …)
   MUJER (idMujer, …)

3) Si **una tiene cardinalidad (1,1) y la otra (0,1), se propaga la clave de la entidad que tiene cardinalidad (1,1) a la que tiene (0,1)**.



   EMPLEADO (idEmpleado, …)
   DEPARTAMENTO (idDepartamento, …, idEmpleado[fk: EMPLEADO])

Habrá que **controlar** además **que la clave propagada como clave ajena no pueda tomar valores nulos**.

5.4.4 Relaciones reflexivas.

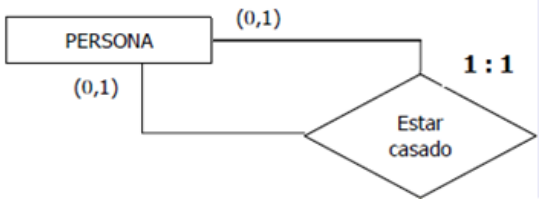
TRANSFORMACION DE UNA INTERRELACION REFLEXIVA.

Se trata de relaciones en las que solo participa una entidad. **Como regla general toda relación reflexiva se convierte en dos tablas: una para la entidad y otra para la relación**.

Se pueden presentar los siguientes casos:

Relación *1:1*.

No se crea una tabla para la relación. La clave de la entidad se repite, con lo que la tabla resultante tendrá ese atributo dos veces, una como clave primaria y otra como clave ajena de ella misma.

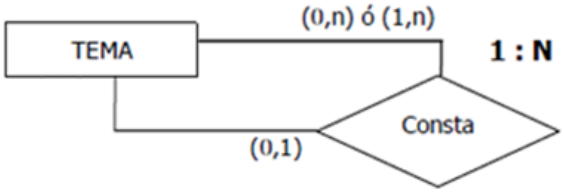


PERSONA (DNI, nombre, …, DNI_Esposo[fk: PERSONA])

Relación *1:N*.

En este caso hay que tener en cuenta la cardinalidad del lado muchos:

a) **Si no es obligatoria se crea una nueva tabla cuya clave será la de la entidad del lado muchos** (*idTema*) **y se propaga la clave a la nueva tabla como clave ajena** (cardinalidad mínima 0 o 1).

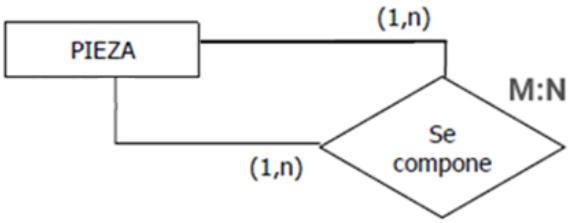


CONSTA (idTema[fk: TEMA], idTemaP[fk: TEMA])
TEMA (idTema, …)

b) **Si es siempre obligatoria no se crea una nueva tabla** (cardinalidad mínima 1).

TEMA (idTema, …, idTemaP[fk: TEMA])

Relación *N:M*.



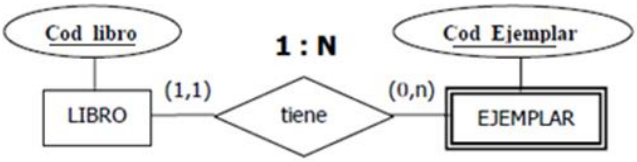
La tabla que resulta de la relación contendrá dos veces la clave primaria de la entidad del lado muchos, más los atributos de la relación, si los hay. La **clave de esta nueva tabla será la combinación de las dos**. Ej.: cada pieza se compone de muchas piezas que, a su vez están compuestas por piezas.

PIEZA (idPieza, descripción, …)
SE_COMPONE (idPiezaC[fk: PIEZA], idPieza[fk: PIEZA]) → *idPiezaC es la pieza que se compone de otras*

5.4.5 Otras.

DEPENDENCIAS (entidades débiles).

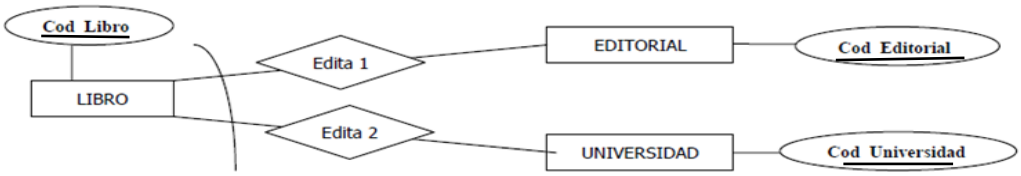
- ✓ **En existencia: se propaga la clave, creando una clave ajena** con las características de
  - No permitir nulos.
  - Modificación en cascada.
  - Borrado en cascada.
- ✓ **En existencia e identificación**: **la propagación de la clave se hace de forma que la clave primaria de la entidad débil estará formada por la concatenación de las dos claves**.



LIBRO (Cod_libro, …)
EJEMPLAR (Cod_libro[fk: LIBRO], Cod_Ejemplar, …)

RELACIONES EXCLUSIVAS.

Se propaga la clave de las entidades que se excluyen, a la otra entidad, como claves ajenas.



Para que se cumpla la exclusividad **es necesario definir restricciones en cada caso**. Por ejemplo, **chequeando que, si una de las claves ajenas tiene datos, la otra sea nula y viceversa**.

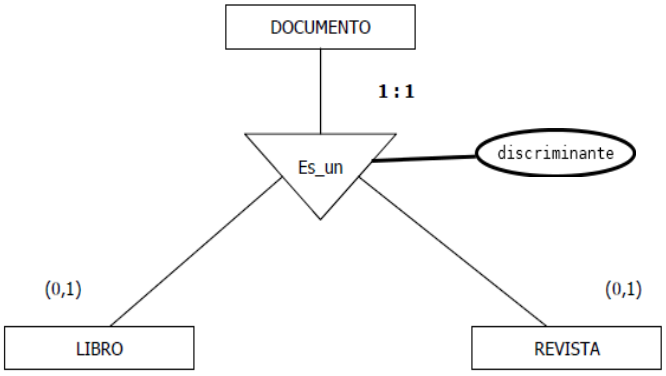
LIBRO (Cod_libro, …, Cod_Universidad[fk: UNIVERSIDAD], Cod_Editorial[fk: EDITORIAL])
UNIVERSIDAD (Cod_Universidad, …)
EDITORIAL (Cod_Editorial, …)

TRANSFORMACIÓN DE JERARQUÍAS.

Pueden darse **3 opciones**:

a) **Crear una sola tabla con todos los atributos de la entidad y de los subtipos, añadiendo como un atributo más el atributo discriminante** (integración de los subtipos en el supertipo). Esto se **aplica cuando**:

- ✓ Los **subtipos se diferencian en muy pocos atributos**.
- ✓ Las relaciones que los asocian al resto de las entidades sean las mismas para los subtipos.



Si la **jerarquía es**:
- ✓ **Total**: el atributo **discriminante no admitirá nulos**.
- ✓ **Parcial**: el atributo **discriminante si admitirá nulos**.

Si **entre los subtipos puede haber**:
- ✓ **Solapamiento**: se forman grupos repetitivos, por tanto, será **necesario crear una nueva tabla que asocie el atributo discriminante con el supertipo**.
- ✓ **Exclusividad**: **no** es necesaria una **tabla nueva**.

DOCUMENTO (<u>código</u>, título, idioma, …, tipo)

b) **<u>Crear una tabla para cada tipo y subtipos que haya</u>**. Esto se **aplica cuando**:
- ✓ Existen **muchos atributos distintos entre los subtipos**.
- ✓ **Se quieren mantener los atributos comunes en una tabla**.

DOCUMENTO (<u>código</u>, título, idioma, …)
LIBRO (<u>códigoLibro</u> [fk: DOCUMENTO], …)
REVISTA (<u>códigoRevista</u> [fk: DOCUMENTO], …)

c) Crear **<u>una tabla por cada subtipo, incluyendo los atributos comunes en cada una</u>** (**eliminación del supertipo**). Esto se **aplica cuando**:
- ✓ Existen **muchos atributos distintos entre los subtipos**.
- ✓ Los **accesos a los datos de los subtipos siempre afectan a los atributos comunes**.

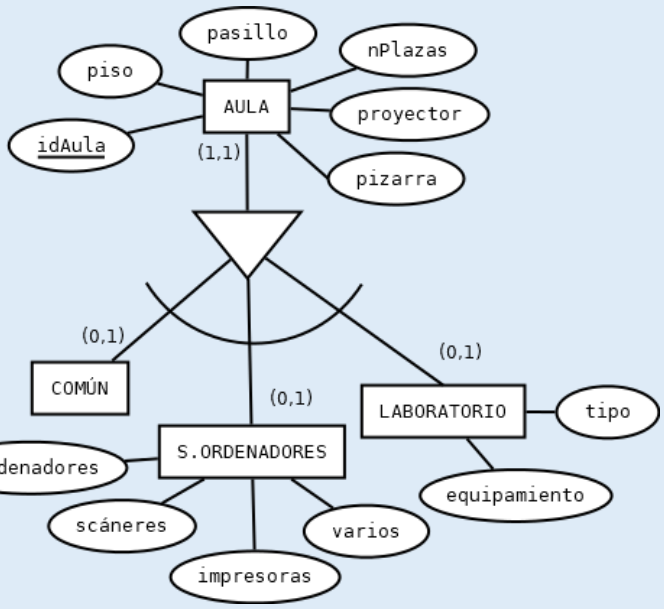
LIBRO (<u>códigoLibro</u>, título, idioma, …)
REVISTA (<u>códigoRevista</u>, título, idioma, …)

<u>Ventajas e inconvenientes:</u>

Opción a) → Es la más rápida por tener que acceder a una sola entidad.
Opción b) → La menos eficiente. La mejor desde un punto de vista semántico.
Opción c) → Más eficiente en consultas que afecten a todos los atributos de un subtipo. Menos eficiente en consultas que afecten a los atributos comunes. Introduce redundancias. Es la que pierde más semántica.

Ejercicio resuelto

Partiendo del modelo *Entidad/Relación* que se muestra en la imagen obtener el modelo relacional correspondiente.



<u>Solución</u>: aplicando opción b)

AULA *(<u>idAula</u>, piso, pasillo, nPlazas, proyector, pizarra)*

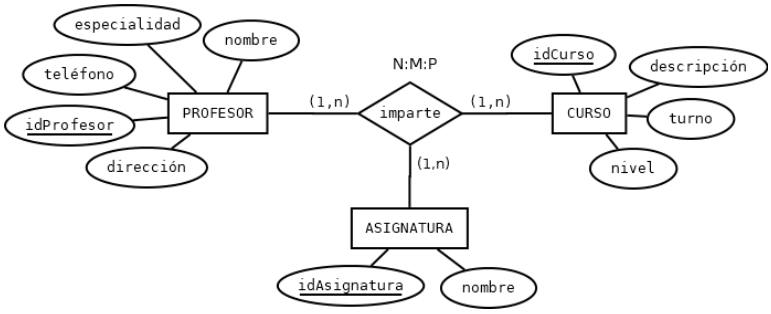
COMÚN *(<u>idAulaComun</u>[fk: AULA])*

S.ORDENADORES *(<u>idAulaOrdena</u>[fk: AULA], nOrdenadores, scáneres, impresoras, varios)*

LABORATORIO *(<u>idAulaLabor</u>[fk: AULA], tipo, equipamiento)*

RELACIONES N-ARIAS.

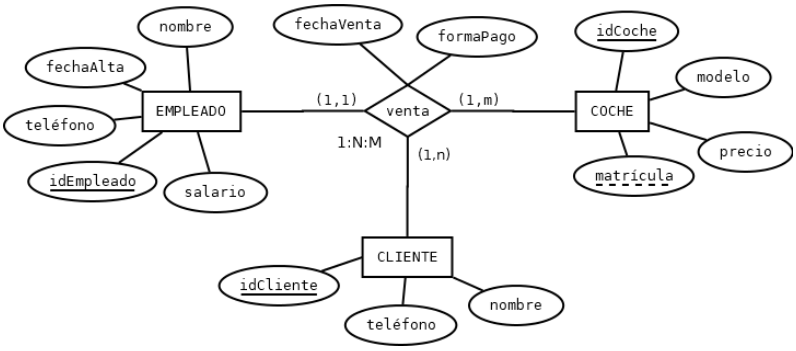
En este tipo de relaciones **se agrupan 3 o más entidades, y para pasar al modelo de datos relacional cada entidad se convierte en una tabla, así como la relación, que va a contener los atributos propios de ella más las claves de todas las entidades**. La **clave de la tabla resultante será la concatenación de las claves de las entidades**. Hay que **tener en cuenta**:



- ✓ Si la **relación es *N:M:P***, es decir, si todas las entidades participan con cardinalidad máxima *N*, la **clave de la tabla resultante es la unión de las claves de las entidades que relaciona**. Esa tabla incluirá los atributos de la relación si los hubiera.

PROFESOR (idProfesor, nombre, dirección, teléfono, especialidad)
CURSO (idCurso, descripción, nivel, turno)
ASIGNATURA (idAsignatura, nombre)
IMPARTE (idProfesor[fk: PROFESOR], idCurso[fk: CURSO], idAsignatura[fk: ASIGNATURA])

- ✓ Si la **relación es *1:N:M***, es decir, una de las entidades participa con **cardinalidad máxima 1, la clave de esta entidad no pasa a formar parte de la clave de la tabla resultante, pero forma parte de la relación como un atributo más**.



CLIENTE (idCliente, nombre, teléfono)
COCHE (idCoche, matrícula, modelo, precio)
EMPLEADO (idEmpleado, nombre, teléfono, salario, fechaAlta)
VENTA (idCoche[fk: COCHE], idCliente[fk: CLIENTE], idEmpleado[fk: EMPLEADO], formaPago, fechaVenta)

5.5 Pérdida de la semántica en la transformación al modelo relacional.

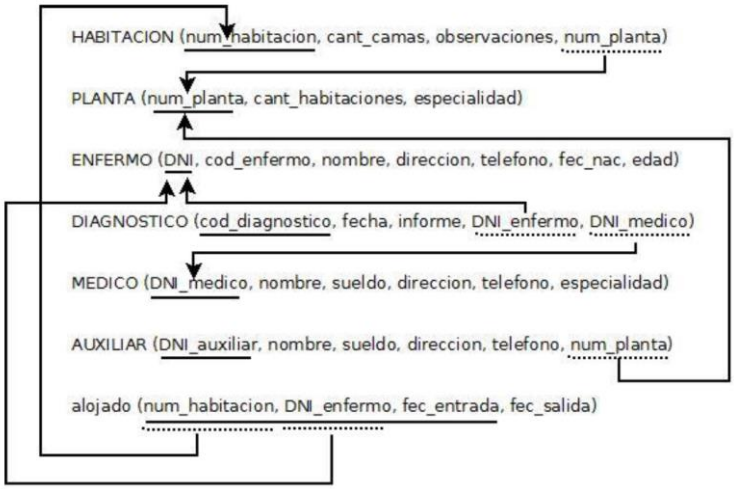
Algunas restricciones son necesarias controlarlas con mecanismos externos al modelo relacional dado que muchos *SGBD* no implementan el modelo relacional completo (hay que **recurrir a otros medios como disparadores, procedimientos almacenados o aplicaciones externas**).

Algunas de las **restricciones** de los esquemas *E/R* **que deben contemplarse** en la transformación al modelo relacional mediante *checks*, aserciones o disparadores son:

- ✓ **Cardinalidades mínimas de *1* en las relaciones *N:M* y *1:N*** (excluyendo aquellas que se controlan con las restricciones *NOT NULL* cuando se realiza una propagación de clave).
- ✓ **Cardinalidades máximas conocidas** en las relaciones binarias *N:M*, *1:N* y relaciones ternarias.
- ✓ **Exclusividad en las generalizaciones**.
- ✓ **Inserciones y borrado en las generalizaciones**.
- ✓ **Restricciones que no figuren en el enunciado inicial pero que se consideran adecuadas** o convenientes.

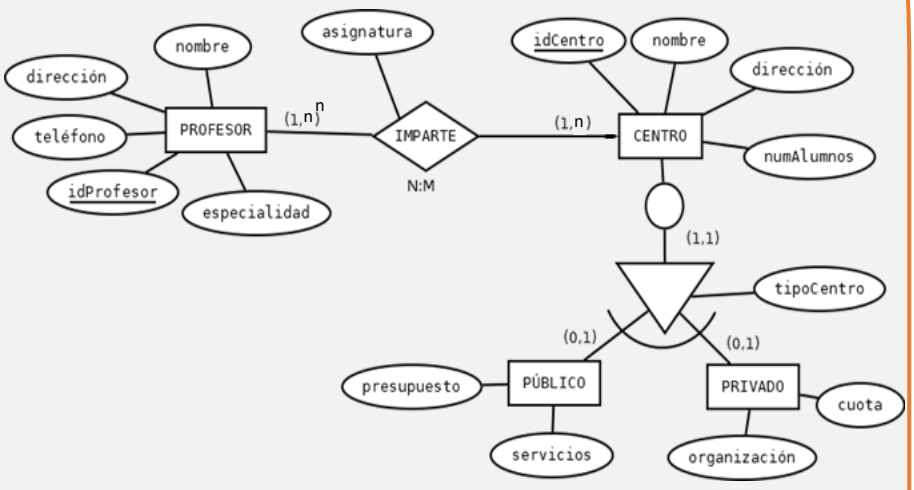
5.6 Grafos relacionales.

Es un **esquema relacional en el que hay líneas que enlazan las claves principales con las claves secundarias (foráneas) para representar mejor las relaciones** (ver imagen, el subrayado mediante puntos indica clave foránea [*fk: TABLA*]).



TAREA

10. **Dado el esquema *E/R* siguiente. Según cada una de las reglas indicadas, ¿Qué esquema *E/R* obtendremos? ¿con qué tablas contará el modelo relacional resultante?**
    a) **Integra los subtipos en el supertipo.**
    b) **Eliminación del supertipo.**



11. **Realiza el diagrama *E/R* que cumpla las especificaciones y pásalo al modelo de datos relacional.**
    Se desea mecanizar la biblioteca de un centro educativo. En la biblioteca existen fichas de autores y libros. Un autor puede escribir varios libros, y un libro puede ser escrito por varios autores. Un libro está formado por ejemplares que son los que se prestan a los usuarios.
    Así un libro tiene muchos ejemplares y un ejemplar pertenece sólo a un libro. De los ejemplares nos interesa saber la localización dentro de la biblioteca. Los ejemplares son prestados a los usuarios, un usuario puede tomar prestados varios ejemplares y un ejemplar puede ser prestado a varios usuarios. Del préstamo nos interesa saber la fecha de préstamo y la de devolución.

12. **Realiza el diagrama *E/R* que cumpla las especificaciones y pásalo al modelo de datos relacional.**
    Se desea informatizar la gestión de los proyectos del departamento de química de una universidad siguiendo las siguientes especificaciones:
    - ***Al departamento llegan una serie de clientes que quieren realizar proyectos. Generalmente los clientes son empresas que realizan contratos con el grupo de investigadores del departamento. Un cliente puede realizar varios proyectos.***
    - ***Un proyecto es de un cliente. Cada proyecto tiene asignada una cuantía de dinero que se utilizará para pagar los gastos del proyecto. De esta cuantía se saca el dinero para realizar los pagos a los colaboradores. También nos interesa saber de los proyectos el nombre, la fecha de comienzo, la de fin, entre otros.***
    - ***De cada proyecto se realizan muchos pagos para pagar a los colaboradores.***
    - ***De los pagos nos interesa saber el concepto, la cantidad, el IVA aplicado y la fecha del pago.***
    - ***Existen varios tipos de pagos (por ejemplo: nómina, representación, material, etc.). Un pago es de un tipo de pago, y a un tipo de pago pueden pertenecer muchos pagos.***
    - ***Existen una serie de colaboradores que son personas o entidades que van a recibir el dinero de los pagos en concepto de una tarea realizada o la compra de material. Un pago sólo puede ser para un colaborador. Este a su vez puede recibir muchos pagos.***
    - ***De los colaboradores nos interesa saber: nombre, NIF, domicilio, teléfono, retención, banco y nº de cuenta.***

13. **Obtén el esquema relacional a partir del modelo *E/R* del ejercicio 6.**

14. **Obtén el esquema relacional a partir del modelo *E/R* del ejercicio 7.**

6 Normalización de modelos relacionales.

A principios de la década de los setenta, concretamente en 1972, *Codd* establece una **técnica para llevar a cabo el diseño de la estructura lógica de los datos representados a través del modelo relacional**, a la que denominó **normalización**. Pero esta técnica no ha de utilizarse para el diseño de la *BD*, sino como un **proceso de refinamiento que debe aplicarse después de lo que se conoce como "paso a tablas", o lo que formalmente se denomina traducción del esquema conceptual al esquema lógico**. Este proceso de refinamiento **conseguirá los siguientes objetivos**:

- ✓ **Suprimir dependencias erróneas entre atributos**.

- ✓ **Disminuir o eliminar redundancias**.
- ✓ **Asegurar la integridad de los datos**.
- ✓ **Optimizar los procesos de inserción, modificación y borrado en la *BD***.

> ***Normalización**: proceso que consiste en imponer a las tablas del modelo Relacional una serie de restricciones a través de un conjunto de transformaciones consecutivas. Este proceso garantizará que las tablas contienen los atributos necesarios y suficientes para describir la realidad de la entidad que representan, permitiendo separar aquellos atributos que por su contenido podrían generar la creación de otra tabla.*

Una **definición sencilla** es que la **normalización** es un **proceso repetitivo que nos permite aplicar una serie de reglas a nuestras relaciones, con la finalidad de evitar redundancias y garantizar la integridad de los datos**.

El proceso de normalización.

El esquema relacional debe modelizar la realidad y para obtenerlo se ha seguido el camino que consiste en, primero, efectuar el diagrama *E/R* para luego efectuar la traducción al modelo relacional (paso a tablas). Si el diagrama *E/R* era correcto, se habrá obtenido un esquema relacional del todo correcto. Este sería el camino aconsejable, pero no siempre es así. **Es posible que un buen diseño produzca tablas defectuosas con redundancia de datos e inconsistencia. Estos problemas, cuando existan, pueden ser eliminados mediante el proceso de normalización**.

El proceso de normalización **se basa en el análisis de las dependencias entre atributos**. Para ello tendrá en cuenta los conceptos de: **dependencia funcional, dependencia funcional completa y dependencia funcional transitiva**.

¿Y cómo se aplica la normalización? Es un proceso que **se realiza en varias etapas secuenciales**. **Cada etapa está asociada a una forma normal (*FN*), que establece unos requisitos a cumplir por la tabla sobre la que se aplica**. Existen varias formas normales: **Primera, Segunda, Tercera, Boyce-Codd, Cuarta, Quinta y Dominio-Clave**. El paso de una *FN* a otra es consecutivo, **si no se satisface una determinada *FN* no puede pasarse a la siguiente**.

Según se va avanzando en la normalización, los requisitos a cumplir serán cada vez más restrictivos, lo que hará que el esquema relacional sea cada vez más robusto.

Como norma general, **para garantizar que no existan problemas en la actualización de datos, es recomendable aplicar el proceso de normalización hasta *Tercera Forma Normal*** (*3FN*), estas tres primeras formas normales fueron postuladas por *Edgar F. Codd*, quien fue el creador del modelo relacional.

6.1 Tipos de dependencias.

Se van a desarrollar aquí los conceptos sobre los que se basa el análisis de dependencias entre atributos, que se lleva a cabo en el proceso de normalización antes indicado, son los siguientes:

- ✓ **Dependencia Funcional (*DF*)**: **dados los atributos *A* y *B*, se dice que *B* depende funcionalmente de *A*, sí, y solo sí, para cada valor de *A* sólo puede existir un valor de *B***. La dependencia funcional siempre se establece **entre atributos de una misma tabla**. El atributo *A* se denomina determinante, ya que *A* determina el valor de *B*. Para representar esta dependencia funcional se utiliza la siguiente **notación: *A* → *B***. Tener en cuenta que *A* y *B* podrían ser un solo atributo o un conjunto de ellos.

- ✓ **Dependencia Funcional Completa (*DFC*)**: dados los atributos *A1*, *A2*, ..., *Ak* y *B*, **se dice que *B* depende funcionalmente de forma completa de *A1*, *A2*, ..., *Ak*, sí y solo si *B* depende funcionalmente del conjunto de atributos *A1*, *A2*, ..., *Ak*, pero no de ninguno de sus posibles subconjuntos**. **Notación: *A1*, *A2*, … → *B*** (es decir, *A1, A2, …* determinan *B)*.

- ✓ **Dependencia Funcional Transitiva (*DFT*)**: dados tres atributos *A*, *B* y *C*, **se dice que existe una dependencia transitiva entre *A* y *C*, si *B* depende funcionalmente de *A*** (es decir, *A* determina *B*) **y *C* depende funcionalmente de *B*** (es decir, *B* determina *C)*. *A*, *B* y *C* podrían ser un solo atributo o un conjunto de ellos. **Notación: *A* → *B* → *C***.

Ejercicio resuelto

Dadas las siguientes tablas:

EMPLEADO (DNI, Nombre, Dirección, Localidad, Cod_Localidad, Nombre_hijo, Edad_hijo)
LIBRO (Título_libro, Num_ejemplar, Autor, Editorial, Precio)

Resolver las siguientes cuestiones:

a) **Indicar qué atributos presentan una *DF* de la clave primaria de la tabla *EMPLEADO*.**
b) **Indicar qué atributos presentan una *DFC* en la tabla *LIBRO*.**
c) **Indicar qué atributos presentan una *DFT* en la tabla *EMPLEADO*.**

Apartado a):

Los atributos *Nombre* y *Dirección* dependen funcionalmente de *DNI*, ya que para un *DNI* específico sólo podrá haber un nombre y una dirección. Pero los atributos *Nombre_hijo* y *Edad_hijo* no presentan esa *DF* de *DNI*, ya que para un *DNI* específico se podrían tener varios valores diferentes en esos atributos (varios hijos). Estas dependencias funcionales se expresan mediante la notación: *DNI → Nombre, Dirección*.

Apartado b):

Los atributos *Editorial* y *Precio* dependen funcionalmente del conjunto de atributos que forman la clave primaria de la tabla, pero no dependen de *Título_libro* o de *Num_ejemplar* por separado, por lo que presentan una *DFC* de la clave. El atributo *Autor* depende funcionalmente sólo y exclusivamente de *Titulo_libro*, por lo que no presenta una *DFC* de los atributos que forman la clave.

Apartado c):

Los atributos *Cod_Localidad* y *Localidad* dependen funcionalmente de *DNI*, pero entre *Cod_Localidad* y *Localidad* existe otra dependencia funcional. Por tanto, se establece que *Localidad* depende funcionalmente de *Cod_Localidad*, y a su vez, *Cod_Localidad* depende funcionalmente de *DNI*. Con lo que se puede ver que existe una dependencia transitiva entre *Localidad* y *DNI*. Si se representa con la notación asociada a las dependencias funcionales, quedaría: *DNI → Cod_Localidad → Localidad*.

6.2 Formas normales.

6.2.1 Primera Forma Normal (*1FN*).

Una tabla está en *Primera Forma Normal* (*1FN*) sí, y sólo sí, todos los atributos de la misma contienen valores atómicos y no tiene grupos repetitivos (elimina grupos repetidos).

Por lo tanto y según lo establecido en el modelo relacional, en **una relación se debe cumplir** que:

- ✓ **No deben existir grupos repetitivos**.
- ✓ No importa el orden.
- ✓ **Debe existir una llave primaria**.
- ✓ **Todos los atributos deben ser atómicos**.
- ✓ No debe tener tuplas repetidas.

La *1FN* especifica que **cada atributo en las tablas debe ser atómico**. Significa que la información que contienen no puede ser descompuesta en elementos más pequeños (todos los atributos, deben ser indivisibles). Si fuera así, **se tendrían que separar, o en otras columnas, o ponerlos en una tabla separada**.

Por ejemplo, mirar la tabla *PERSONA*:

PERSONA

id	nombre	apellidos	telefonos
1	Raquel	Vanegas López	677 899 154
2	Paula	Monestros García	648 152 123
3	Francisco	Martínez Mesa	645 979 132, 615 123 672

Esta tabla no está en *1FN*. El atributo *teléfonos* no es atómico. Si sólo se permiten 2 números de teléfono por persona, lo que se puede hacer es crear dos columnas, *teléfono1* y *teléfono2*. Obviamente, lo mismo no vale para los

apellidos, ya que no interesa acceder a ellos por separado (o podría ser que sí y sería necesario separarlos en dos columnas, *apellido1* y *apellido2*).

Pero, ¿qué hacer si la cantidad de números de teléfono no es limitada? Se tendrá que crear otra tabla que contenga los números de teléfono con una relación uno a varios con la tabla *PERSONA*.

Tabla *PERSONA* en *1FN*

id (PK)	nombre	apellidos
1	Raquel	Vanegas López
2	Paula	Monestros García
3	Francisco	Martínez Mesa

personaId (FK)	telefono
1	677 899 154
2	648 152 123
3	645 979 132
3	615 123 672

Tabla *TELÉFONO* en relación uno a varios con *PERSONA*.

Otro ejemplo: suponer que se tiene en una tabla una columna *Dirección* para almacenar la dirección completa, dato que se compondría del *nombre de la calle*, el *número*, el *piso*, la *puerta*, el *código postal*, la *población* y la *provincia*.

Una tabla con esta estructura plantea problemas a la hora de recuperar información. Imaginar que se necesita conocer todas las entradas correspondientes a una determinada población, o que se quiere buscar a partir del código postal. Al ser la dirección completa una secuencia de caracteres de estructura libre no resultaría nada fácil.

Además, se debe evitar la repetición de los datos de la población y provincia en cada una de las filas. **Siempre que al muestrear la información de una tabla aparezcan datos repetidos, existe la posibilidad de crear una tabla independiente con ellos**.

*La mayoría de las veces no será necesario aplicar este proceso ya que las tablas se encontrarán en 1FN si se ha hecho un buen diseño conceptual y lógico. La aplicación de **la 1FN será la correcta elección de la clave principal**.*

Ejemplo: *¿Cómo hacer que una tabla esté en 1FN?*

Suponer que se tiene la siguiente relación que representa los proyectos que lleva cada empleado y el puesto que tiene actualmente (ver tabla *Empleados*).

Empleados			
Nombre	**Apellido**	**Puesto**	**Proyectos**
José	Pérez	Supervisor	Aplicación móvil iOS, Página web, CMS
Ángela	García	Supervisor	Base de datos, CMS

En dicha tabla de empleados, se tienen los siguientes problemas:

- ✓ No existe una llave primaria.
- ✓ El atributo *Proyectos* no es una columna atómica o indivisible.

Entonces, se pueden hacer los siguientes cambios:

- ✓ Primero agregar una llave primaria a la que se puede llamar *No Empleado*.
- ✓ Dividir el atributo *Proyecto* que no es atómico, en atributos atómicos, con el mismo dominio.

Empleados						
No Empleado	**Nombre**	**Apellido**	**Puesto**	**Proyecto 1**	**Proyecto 2**	**Proyecto 3**
1	José	Pérez	Supervisor	Aplicación móvil iOS	Página web	CMS
2	Ángela	García	Supervisor	Base de datos	CMS	

Al dividir el atributo *Proyecto*, se tienen atributos atómicos, pero, también se crea algo que se llama grupo repetitivo, este tipo de grupos generan algunos problemas, por ejemplo:

- ✓ Imponen un límite, ya que, en este caso, un empleado no podría tener más de tres proyectos.
- ✓ Por otra parte, si se observa, *Ángela* solo tiene dos proyectos, por lo que el atributo *Proyecto 3* se encuentra vacío, lo que provoca un desperdicio de espacio.
- ✓ Otro de los problemas está relacionado con las búsquedas, ya que, por ejemplo, en este caso, si se desean conocer los proyectos que tiene cada empleado, se deberían realizar búsquedas sobre tres atributos diferentes.

Entonces, cómo aún se tienen grupos repetitivos, y según la definición de la *1FN*, las relaciones no deben tener ningún grupo repetitivo, por lo tanto, se debe encontrar una forma de eliminar dicho grupo.

Bien, en este punto en el que nos encontramos, lo más sencillo es generar una fila/tupla por cada concurrencia que exista, esto nos llevaría a la siguiente tabla:

No Empleado	Nombre	Apellido	Puesto	Proyecto
1	José	Pérez	Supervisor	Aplicación móvil iOS
1	José	Pérez	Supervisor	Página web
1	José	Pérez	Supervisor	CMS
2	Ángela	García	Supervisor	Base de datos
2	Ángela	García	Supervisor	CMS

Haciendo esto se ha eliminado al grupo repetitivo, pero:

- ✓ Ahora se tienen llaves primarias repetidas.
- ✓ Y redundancia a simple vista en los datos.

Por lo tanto, se debe seguir operando, y lo que se va a hacer ahora es dividir la tabla en dos relaciones:

1. Una relación con el grupo repetitivo que se llamará *Proyectos asignados*.
2. Y otra relación con el resto de atributos que contenga los datos de la relación original, la cual se nombrará nuevamente como *Empleados*.
3. Ahora, simplemente se debe agregar la llave primaria de *Empleados* en la nueva relación *Proyectos asignados*.

Empleados			
No Empleado	Nombre	Apellido	Puesto
1	José	Pérez	Supervisor
2	Ángela	García	Supervisor

Proyectos asignados	
No Empleado	Proyecto
1	Aplicación móvil iOS
1	Página web
1	CMS
2	Base de datos
2	CMS

Sin embargo, *Proyectos asignados* no está normalizada, así que se debe repetir el proceso hasta que dicha relación este en la *1FN*, para ello se creará una tabla intermedia, pero antes, se va a analizar cómo está actualmente:

- ✓ Observar que la llave *No Empleados* tiene datos duplicados, no obstante, es una llave primaria de *Empleados*, y en esta relación es una llave foránea, y según lo que se ha visto en el modelo relacional, una llave foránea puede tener datos duplicados.
- ✓ También se puede ver que el atributo *Proyecto* tiene redundancia respecto al dato *CMS*.

Proyectos asignados	
No Empleado	Proyecto
1	Aplicación móvil iOS
1	Página web
1	CMS
2	Base de datos
2	CMS

Para eliminar la redundancia en el atributo *Proyecto*, se debe crear otra relación que:

- ✓ Almacena los datos de los proyectos.
- ✓ Y que tenga su propia llave primaria.
- ✓ A esta nueva relación se la nombrará como *Proyectos*.

Proyectos	
No Proyecto	Proyecto
100	Aplicación móvil iOS
101	Página web
102	CMS
103	Base de datos

Ahora simplemente, agregar la llave primaria *No Proyecto* en la relación *Proyectos Asignados*, por buenas prácticas asegurarse de tener las llaves siempre al inicio de la relación.

Proyectos asignados		
No Empleado	No Proyecto	Proyecto
1	100	Aplicación móvil iOS
1	101	Página web
1	102	CMS
2	103	Base de datos
2	102	CMS

Ahora la relación *Proyectos asignados*:

- ✓ Cuenta con dos llaves foráneas:
  - o *No Empleado*.
  - o *No Proyecto*.
- ✓ **Estas dos llaves foráneas componen una llave primaria compuesta**.
- ✓ No existen tuplas duplicadas.
- ✓ No importa el orden.
- ✓ Todos sus atributos son atómicos.
- ✓ Y no hay grupos repetitivos.

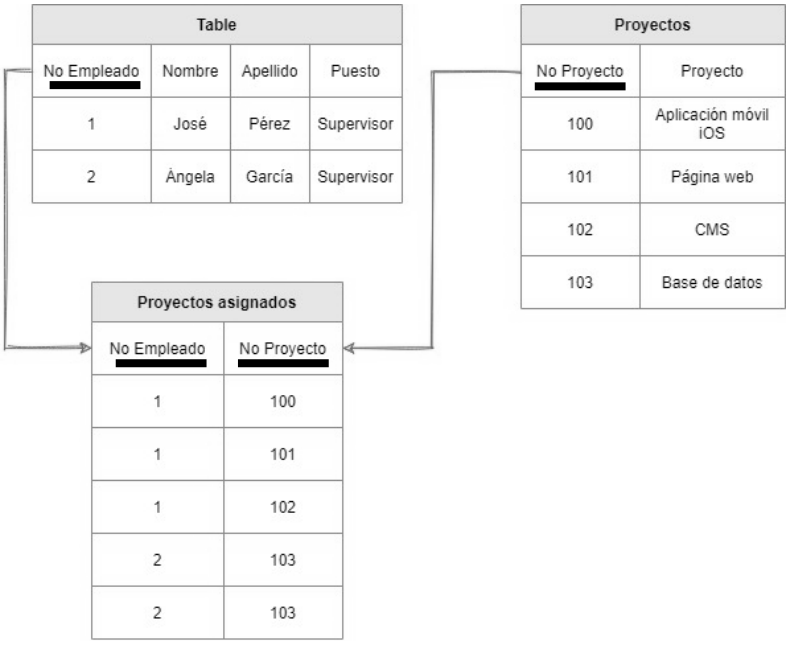
Proyectos asignados	
No Empleado	No Proyecto
1	100
1	101
1	102
2	103
2	102

Sin embargo, el atributo *Proyecto* es un atributo derivable, ya que, se puede obtener de la tabla *Proyectos*, por lo que se puede eliminar y quedarnos solamente con:

- ✓ *No Empleado*.
- ✓ *No Proyecto*.

Por lo tanto, la tabla *Proyectos asignados* es una tabla de relación o tabla intermedia, que, en este caso, relaciona la tabla *Empleados* y *Proyectos* y, además, se encuentra en la primera forma normal (*1FN*).

Finalmente se llegaría al siguiente esquema de relaciones:



Entonces, en la *1FN*:

- ✓ El objetivo es eliminar los grupos repetitivos de forma independiente:
  - Moviendo el grupo a una nueva entidad.
  - Se agrega la llave primaria original, convirtiéndose en llave foránea.
- ✓ Si la nueva entidad no posee una llave primaria natural, se puede agregar una llave suplente o una llave de otra entidad.
- ✓ Si en el proceso de normalización, nos encontramos con otras entidades, se debe repetir el proceso de normalización.

6.2.2 Segunda Forma Normal (*2FN*).

A esta segunda forma también se la conoce como "***Forma de la dependencia funcional completa***", pero ¿qué indica la segunda forma normal (*2FN*)?

Una tabla está en *Segunda Forma Normal* (*2FN*) sí, y sólo sí, está en *1FN* y, además, todos los atributos que no pertenecen a la clave dependen funcionalmente de forma completa de ella (elimina las dependencias parciales de la clave primaria). Es obvio que **una tabla que esté en *1FN* y cuya clave esté compuesta por un único atributo, estará en *2FN***.

Y a que se refiere con depender únicamente de una parte de la clave primaría, pues bien, recordar que en una relación existe una clave primaria, y esta clave primaria puede estar formada por uno o más atributos, entonces:

- ✓ La **clave primara completa** es la conformación de todos los atributos que forman la clave, ya sea uno o más de uno.
- ✓ En una entidad los atributos pueden depender de todos los atributos que conforman la clave primaria, a esto se le conoce como **dependencia funcional completa**.
- ✓ Pero, también existe el caso donde en una entidad, los atributos dependan únicamente de algunos atributos que conforman la clave primaría, a esto se le conoce como **dependencia funcional parcial**.

Por lo tanto, para llevar a una relación a la *2FN* se necesita:

1. Identificar las dependencias parciales.
2. Crear una nueva relación por cada dependencia parcial existente.

Mirar por ejemplo la tabla *HABILIDAD_EMPLEADO*, donde la clave primaria es la combinación de los atributos *Empleado* y *Habilidad*.

HABILIDAD_EMPLEADO

Empleado (PK)	Habilidad (PK)	dirección
Jones	mecánico	131 calle bondad
Jones	carpintero	131 calle bondad
Marvin	mecánico	2 calle sol
Beckerd	limpieza	25 avenida de la borsa
Beckerd	construcción	25 avenida de la borsa

El atributo *dirección* es dependiente de solo una parte de la clave compuesta, por lo cual se tiene redundancia en la columna dirección. Se está repitiendo innecesariamente el valor de este atributo. Peor aún, si en algún momento se actualiza la dirección en una fila de esta tabla, se tendrá un problema de consistencia del valor. Esta tabla no respeta la *2FN*.

La solución consiste en sacar el atributo dirección de la tabla y colocarlo en otra tabla con relación uno a varios.

La nueva tabla *EMPLEADO*

Empleado (PK, FK)	direccion
Jones	131 calle bondad
Marvin	2 calle sol
Beckerd	25 avenida de la bolsa

La tabla *HABILIDAD_EMPLEADO*

empleado (PK)	habilidad (PK)
Jones	mecánico
Jones	carpintero
Marvin	mecánico
Beckerd	limpieza
Beckerd	construcción

Este diseño sí está en *2FN*. Si se quiere actualizar la dirección de un empleado, se hará a través de la tabla *EMPLEADO*, sin riesgo de incoherencia en los datos.

¿Cómo se normaliza a *Segunda Forma Normal*?

a) **Se crea, a partir de la tabla inicial, una nueva tabla con los atributos que dependen funcionalmente de forma completa de la clave**. La **clave de esta tabla será la misma clave primaria de la tabla inicial**. Esta tabla ya estará en *2FN*.

b) **Con los atributos restantes, se crea otra tabla que tendrá por clave el subconjunto de atributos de la clave inicial de los que dependen de forma completa**. Se **comprueba si esta tabla está en *2FN***. Si es así, la tabla inicial ya está normalizada y el proceso termina. Si no está en *2FN*, se toma esta segunda tabla como tabla inicial y se repite el proceso.

Cuando la clave primaria de una tabla se compone de un solo atributo automáticamente está en 2FN

Ejemplo: *¿Cómo hacer que una relación esté en 2FN?*

Analicemos el siguiente caso, el cual es similar al que se usó en la *1FN*: una relación con la asignación de proyectos y el costo.

Proyectos asignados			
No Empleado	No Proyecto	Proyecto	Costo
1	100	Aplicación móvil iOS	$10000
1	101	Página web	$25000
1	102	CMS	$50000
2	103	Base de datos	$100000
2	102	CMS	$50000

Se puede identificar lo siguiente:

- ✓ Se tiene una clave primaria compuesta:
  - *No Empleado*.
  - *No Proyecto*.
- ✓ Los atributos *Proyecto* y *Costo* son atributos dependientes.
- ✓ No obstante, el atributo *Costo* depende de la clave compuesta, es decir, tanto de *No Empleado* como de *No Proyecto*, a esto se le conoce como una **dependencia funcional completa**.
- ✓ Por otra parte, el atributo *Proyecto*, únicamente depende del atributo *No proyecto*, es decir, solo depende de una parte de la clave primaria, a esto se le conoce como **dependencia funcional parcial**.

Bajo estas condiciones y recordando la definición de la *2FN*, se tienen dependencias parciales, lo que indica que *Proyectos asignados* no está en la *2FN*, para poder eliminar las dependencias, se tiene que hacer lo siguiente:

1. Separar las dependencias en una nueva relación.
2. Eliminar las tuplas duplicadas.
3. Dejar en una relación el resto de atributos con su clave primaria.

Entonces, primero se separarán las dependencias parciales y se creará una relación de nombre *Proyectos*. Seguidamente asegurarse de que no existan tuplas duplicadas, que en este caso es eliminar el último registro.

Proyectos	
No Proyecto	Proyecto
100	Aplicación móvil iOS
101	Página web
102	CMS
103	Base de datos

Dejar el resto de atributos junto a la clave primaria.

Con esta división se ha logrado que en las dos relaciones nuevas:

- ✓ Todos los atributos dependan únicamente de su clave primaria.
- ✓ Además, tienen un dominio común *No Proyecto*, con el cual se puede realizar la unión natural entre estas nuevas relaciones.

Proyectos asignados		
No Empleado	No Proyecto	Costo
1	100	$10000
1	101	$25000
1	102	$50000
2	103	$100000
2	102	$50000

Por lo tanto, las relaciones *Proyectos* y *Proyectos asignados* están en su segunda forma normal (*2FN*).

Ejercicio resuelto

La siguiente relación no está en *2FN*. ¿Qué relaciones resultan de pasarla a *2FN*?

CONSULTA (codPaciente, codMédico, fechaConsulta, nombrePaciente, diagnóstico, tratamiento, especialidad)

Solución:

Relaciones resultantes al pasar a *2FN*:

CONSULTA (codPaciente [fk: PACIENTE], codMédico [fk: MÉDICO], fechaConsulta, diagnóstico, tratamiento)
MÉDICO (codMédico, especialidad)
PACIENTE (codPaciente, nombrePaciente)

6.2.3 Tercera Forma Normal (*3FN*).

En esta tercera forma normal se toca el tema de la transitividad y es por ello que se la conoce como la **forma de la dependencia funcional transitiva**.

Una tabla está en *Tercera Forma Normal* (*3FN*) sí, y sólo sí, está en *2FN* y, además, cada atributo que no está en la clave primaria no depende transitivamente de la clave primaria (elimina las dependencias transitivas de la clave primaria).

La tercera forma normal *3FN* también depende de las formas normales anteriores. Se parece a la *2FN*, pero trata de los atributos que no forman parte de la clave.

Quizá la definición de la tercera forma normal (*3FN*) parezca muy compleja, pero lo que al final quiere decir es que, **todos los atributos que no forman parte de la clave primaria, deben depender únicamente de la clave primaria y no de otro atributo que no sea parte una clave**.

Para que se comprenda mejor, se define lo siguiente:

- ✓ Un **atributo primo**, es aquel **atributo que forma parte de una clave primaria**.
- ✓ Y un **atributo no primo**, es **aquel que no forma parte de ninguna clave primaria**.

Entonces, **en la *3FN* ningún atributo no primo puede depender funcionalmente de otro atributo no primo**.

Ejemplo: una *BD* dónde se guardan los tricks realizados por los concursantes en un concurso de skateboard.

Tomar la siguiente tabla *TRICK_PARTICIPANTE*:

TRICK_PARTICIPANTE

id	nombre	trick	puntos
1	Maria	Olle	4
2	Rafael	Heelflip	2
3	Amalia	50-50 rail	5
4	Bert	Heelflip	2
5	Bert	BS 180	3
6	José	FS 360	6
7	José	Ollie	4

Para cumplir con la *3FN*, se debe sacar la columna puntos y meterla en otra tabla, ya que cada trick vale un número determinado de puntos.

id (PK)	nombre	trick
1	Maria	Ollie
2	Rafael	Heelflip
3	Amalia	50-50 rail
4	Bert	Heelflip
5	Bert	BS 180
6	José	FS 360
7	José	Ollie

Se tendrá pues la tabla *TRICK_PARTICIPANTE* en *3FN*

trick (FK)	puntos
Ollie	4
Heelflip	2
50-50 rail	5
BS 180	3
FS 360	6

Y la tabla *VALOR_TRICK* con una clave ajena haciendo referencia a los tricks de la tabla *TRICK_PARTICIPANTE*.

Así se asegura que cada trick tiene la misma puntuación como tiene que ser.

¿Cómo se normaliza a *Tercera Forma Normal*?

a) **Se crea, a partir de la tabla inicial, una nueva tabla con los atributos que no poseen dependencias transitivas de la clave primaria**. Esta tabla ya estará en *3FN*.

b) **Con los atributos restantes, se crea otra tabla con los atributos no clave que intervienen en la dependencia transitiva, y se elige uno de ellos como clave primaria, si cumple los requisitos para ello**. Se **comprueba si esta tabla está en *3FN***. Si es así, la tabla inicial ya está normalizada y el proceso termina. Si no está en *3FN*, se toma esta segunda tabla como tabla inicial y se repite el proceso.

Ejemplo: Llevando una relación a *3FN*.

En la siguiente relación *Equipos asignados*, se pueden ver los equipos que se encuentran asignados a los clientes. En la relación *Equipos asignados* se puede ver que:

Equipos asignados					
No Cliente	Nombre	Apellido	IMEI	Nombre equipo	Celular
1	Juan	González	123456789	Equipo 1	5500000000
2	Sandra	Pérez	113456789	Equipo 2	5500000001
3	Santiago	Valdés	113356789	Equipo 3	5500000002
4	Pedro	García	113355789	Equipo 4	5500000003
5	Rolando	García	113355779	Equipo 5	5500000004

- ✓ Se encuentra en *1FN*, ya que:
  - Cumple con lo establecido en el modelo relacional.
  - Cuenta con una clave primaria *No Cliente*.
  - No hay grupos repetitivos.
- ✓ Y también se encuentra en *2FN*, ya que:
  - Ninguno de los atributos depende únicamente de una parte de la clave primaria.

Para realizar el análisis, hay que identificar las dependencias funcionales:

- ✓ *No Cliente* → *Nombre*, *Apellido*, *IMEI*.
- ✓ *IMEI* → *Nombre equipo*, *Celular*.

Donde, *IMEI* no es un atributo primo y, tanto *Nombre equipo* y *Celular* dependen de *IMEI*, sin embargo, *Nombre equipo* y *Celular*, tampoco son atributos primos. Esto se contrapone con la *3FN* y se debe corregir separando las dependencias que se contraponen con *3FN* y eliminando de la relación original los atributos dependientes, quedando de la siguiente manera:

Equipos		
IMEI	Nombre equipo	Celular
123456789	Equipo 1	5500000000
113456789	Equipo 2	5500000001
113356789	Equipo 3	5500000002
113355789	Equipo 4	5500000003
113355779	Equipo 5	5500000004

Equipos asignados			
No Cliente	Nombre	Apellido	IMEI
1	Juan	González	123456789
2	Sandra	Pérez	113456789
3	Santiago	Valdés	113356789
4	Pedro	García	113355789
5	Rolando	García	113355779

Ahora las nuevas relaciones *Equipos asignados* y *Equipos*, no tienen atributos no primos dependientes de otros atributos no primos, por lo que, ambas tablas se encuentran en *3FN*.

Por último, en muchos casos de la vida real, normalizar nuestra BD hasta este punto, es más que suficiente, ya que, se han eliminado la mayoría de los casos de redundancia y anomalías que generan problemas, no obstante, existen otras formas normales que se pueden aplicar en ciertos casos, y que, dependiendo de la aplicación, quizá sea necesario el considerar aplicarlas.

6.2.4 Forma Normal de Boyce-Codd (*FNBC*).

Cuando se normaliza por encima de la *3FN*, se dice que es una **normalización superior**, ya que, como se comentó anteriormente, con las tres formas normales en muchas ocasiones, es más que suficiente, sin embargo, existen algunos casos donde no se pueden eliminar ciertas redundancias únicamente aplicando las tres primeras formas normales, y esos casos son los que se verán en este y los siguientes apartados.

La *forma normal de Boyce y Codd*, nace en 1974 a manos de *Raymond Boyce* y *Edgar Codd*, quienes se dan cuenta que, a pesar de aplicar las tres primeras formas normales, puede haber ciertas **redundancias muy inusuales**, sobre todo **cuando existe más de una llave candidata y dichas llaves tienen atributos en común**.

Una tabla está en *Forma Normal de Boyce-Codd* (*FNBC* o *BCFN*) sí, y sólo sí, está en *3FN* y para cada una de las dependencias funcionales no triviales, el determinante es una superclave (elimina dependencias sobre claves candidatas).

Recordar que una **superclave** es un atributo o conjunto de atributos que identifica de forma única a una relación.

Un determinante será todo atributo simple o compuesto del que depende funcionalmente de forma completa algún otro atributo de la tabla. Aquellas tablas en la que todos sus atributos forman parte de la clave primaria, estarán en *FNBC*. Por tanto, si se encuentra un determinante que no es clave candidata, la tabla no estará en *FNBC*. **Esta redundancia suele ocurrir por una mala elección de la clave**. Para normalizar a *FNBC* se tendrá que descomponer la tabla inicial en dos, siendo cuidadosos para evitar la pérdida de información en dicha descomposición.

Es **muy difícil que una tabla que está en *3FN* no esté en *FNBC***, pero se puede lograr eligiendo mal las claves de las tablas. Por ejemplo, si se tiene una tabla con *idTrabajador*, *idDepartamento*, *idResponsable*, donde el *idResponsable* es la persona responsable del trabajador. La clave sería, si cada trabajador puede trabajar en varios departamentos y tener distintos responsables (*idTrabajador*, *idDepartamento*, *idResponsable*). Pero si resulta que cada responsable lo es de un único departamento, entonces *idResponsable* dependería de *idDepartamento*, lo que convierte a *idResponsable* en determinante (atributo que depende de otro atributo), pero no es clave candidata.

Ejemplo: Llevar la siguiente relación a la *FNBC*.

Estudiante	Materia	Profesor
Juan	Algoritmos	Profesor 1
Juan	Base de datos	Profesor 2
Juan	Lenguajes de programación	Profesor 3
Pedro	Base de datos	Profesor 2
Pedro	Lenguajes de programación	Profesor 4
Luis	Algoritmos	Profesor 1
Luis	Lenguajes de programación	Profesor 3

Analizando la relación, se tiene que:

- ✓ Varios estudiantes pueden matricularse en la misma materia.
- ✓ Y a la vez, más de un profesor puede dar la misma materia.

Con esto, se puede determinar el conjunto de dependencias funcionales:

- ✓ Para poder obtener a un profesor, se necesita conocer al estudiante y la materia.
  - o *Estudiante*, *Materia* → *Profesor*
- ✓ Y mediante un profesor, se puede conocer una materia.
  - o *Profesor* → *Materia*

Como se puede observar, prácticamente es el mismo caso que el ejercicio anterior, por lo tanto, las claves candidatas son:

- ✓ *Estudiante*, *Materia*.
- ✓ *Estudiante*, *Profesor*.

Parecería que no hay ningún problema ¿Cierto? Pero, viendo la relación con datos:

- ✓ ¿Qué pasaría si un estudiante decide dejar una materia? Por ejemplo: *Pedro* deja la materia de *Lenguajes de programación*, si esto llegará a pasar, se eliminaría la fila y no se sabría que el *Profesor 4* imparte la materia *Lenguajes de programación*.
- ✓ Por otra parte, si llega un nuevo profesor y quiere dar una nueva materia, no se podría agregar una nueva fila a la relación hasta que un estudiante se inscriba a esa materia con ese nuevo profesor.

Precisamente es este tipo de anomalías lo que trata de eliminar la *FNBC*, y para ello:

- ✓ Nos centramos en la dependencia funcional *Profesor* → *Materia* y la movemos a una nueva relación:
  - o R1 (*Materia*, *Profesor*)
- ✓ Y en la relación original, dejamos el determinante de esta dependencia funcional más el resto de atributos.
  - o R2 (*Estudiante*, *Profesor*)

Quedando nuestras relaciones de la siguiente manera, recuerda eliminar las tuplas duplicadas:

Materia	Profesor
Algoritmos	Profesor 1
Base de datos	Profesor 2
Lenguajes de programación	Profesor 3
Lenguajes de programación	Profesor 4

Estudiante	Profesor
Juan	Profesor 1
Juan	Profesor 2
Juan	Profesor 3
Pedro	Profesor 2
Pedro	Profesor 4
Luis	Profesor 1
Luis	Profesor 3

Ahora nuestras nuevas relaciones *R1* y *R2*, se encuentran en la *FNBC* y eliminan las anomalías de inserción y eliminación de datos que se vieron previamente y, a través de una unión natural, se pueden obtener los datos de la relación original *R*.

6.2.5 Cuarta Forma Normal (*4FN*).

Esta forma nace gracias a *Margaret S. Wu*, quien, en un estudio, analizo las *BD* de cerca de cuarenta empresas, donde nueve *BD*, presentaban anomalías que dieron paso a esta *4FN*.

La cuarta forma normal tiene como objetivo eliminar todas las dependencias multivaluadas de las relaciones, por lo tato **una relación está en *4FN*, sí, y sólo si, se encuentra en la *3FN* o en la *FNBC* y cada dependencia multivaluada no trivial que exista, el determinante es una superclave**.

Una dependencia multivaluada es una sentencia donde dos o más atributos son independientes unos de otros.

Por lo tanto, si una relación contiene dependencias multivaluadas se contraponen a la *4FN* y se debe eliminar.

Ejemplo: Llevar una relación a *4FN*.

Supongamos que tenemos una relación que indica la cobertura de envío de varias sucursales que venden celulares, donde, cada sucursal cubre ciertas zonas donde envía los tipos de celulares que tiene asignados.

Envío de celulares		
Sucursal	**Tipo equipo**	**Zona**
Sucursal 1	Celular 1	Zona 1
Sucursal 1	Celular 1	Zona 2
Sucursal 1	Celular 2	Zona 1
Sucursal 1	Celular 2	Zona 2
Sucursal 2	Celular 1	Zona 1
Sucursal 2	Celular 2	Zona 1
Sucursal 3	Celular 1	Zona 1
Sucursal 3	Celular 1	Zona 3

La relación *Envío de celulares*:

- ✓ Cumple con lo establecido en el modelo relacional.
- ✓ No tiene grupos repetitivos.
- ✓ No tiene dependencias parciales.
- ✓ No hay ninguna dependencia transitiva.
- ✓ Y el determinante es una super llave.

La relación *Envío de celulares* está en la *3FN* y cumple con la *FNBC*.

No obstante, existe una redundancia muy sutil:

- ✓ *Sucursal 3* aparece dos veces, ya que, debe cubrir tanto la *Zona 1* y *Zona 3*.

Entonces, si a *Sucursal 3* se le asigna un nuevo tipo de equipo a vender, se tendría que agregar tantas tuplas que satisfagan las zonas que debe cubrir, que, en este caso, se tendría que agregar dos tuplas.

Sucursal	Zona
Sucursal 1	Zona 1
Sucursal 1	Zona 2
Sucursal 1	Zona 1
Sucursal 1	Zona 2
Sucursal 2	Zona 1
Sucursal 2	Zona 1
Sucursal 3	Zona 1
Sucursal 3	Zona 3

Sucursal	Tipo equipo
Sucursal 1	Celular 1
Sucursal 1	Celular 1
Sucursal 1	Celular 2
Sucursal 1	Celular 2
Sucursal 2	Celular 1
Sucursal 2	Celular 2
Sucursal 3	Celular 1
Sucursal 3	Celular 1

Este tipo de dependencias es lo que se vio anteriormente como una dependencia multivaluada y, se contrapone a la *4FN*.

Para eliminar esta anomalía, se tiene que hacer algo similar a los casos anteriores, dividir en relaciones más pequeñas separando las dependencias multivaluadas, que, en este caso, *Sucursal* multidetermina a *Tipo equipo*, pero también multidetermina a *Zona*, quedando las relaciones de la siguiente forma.

Ahora, eliminemos las tuplas repetidas:

Sucursal	Tipo equipo
Sucursal 1	Celular 1
Sucursal 1	Celular 2
Sucursal 2	Celular 1
Sucursal 2	Celular 2
Sucursal 3	Celular 1

Sucursal	Zona
Sucursal 1	Zona 1
Sucursal 1	Zona 2
Sucursal 2	Zona 1
Sucursal 3	Zona 1
Sucursal 3	Zona 3

Al hacer este movimiento en ambas relaciones, se elimina la anomalía, y se cumpliría con la *4FN* y, se pueden agregar más zonas a una sucursal o se puede agregar más tipos de equipos a cualquier sucursal.

6.2.6 Quinta Forma Normal (*5FN*).

La quinta forma normal, también conocida como la **forma normal de la proyección - unión (*PJ/NF*)**.

Una relación está en *5FN*, sí, y sólo si, está en *4FN* y toda dependencia de unión natural no trivial, es implicada por una superclave.

Otra forma de verlo: **una relación está en *5FN* sí, y sólo si, está en *4FN* y, no puede ser descompuesta no aditivamente en relaciones más pequeñas**.

Si una relación satisface la dependencia de unión natural, dicha relación no se encuentra en *5FN*, ya que puede ser descompuesta no aditivamente en relaciones más pequeñas.

Ejemplo: ¿Cómo llevar una relación a *5FN*?

Para saber si una relación se encuentra o no en *5FN*, se va a validar si satisface o no la dependencia de unión natural. Dada la siguiente relación *Proveedores de equipos*:

Proveedor	Equipo	Proyecto
P1	E1	Proyecto 1
P2	E2	Proyecto 2
P3	E3	Proyecto 3
P4	E4	Proyecto 1

Para validar si esta relación está en *5FN*, se debe crear n proyecciones, en este caso son solo 3:

Proveedor	Equipo
P1	E1
P2	E2
P3	E3
P4	E4

Proveedor	Proyecto
P1	Proyecto 1
P2	Proyecto 2
P3	Proyecto 3
P4	Proyecto 1

Equipo	Proyecto
E1	Proyecto 1
E2	Proyecto 2
E3	Proyecto 3
E4	Proyecto 1

Ahora hacer la unión natural de las relaciones, recordar que el orden con el que se hace la unión natural no importa, se debe llegar al mismo resultado:

Proveedor	Equipo	Proyecto
P1	E1	Proyecto 1
P2	E2	Proyecto 2
P3	E3	Proyecto 3
P4	E4	Proyecto 1

Después de llevar a cabo la unión natural se ve que se ha vuelto a obtener la relación original, esto quiere decir que se ha cumplido y por lo tanto, nuestra relación original no está en la *5FN* ya que satisface la dependencia de unión natural, no obstante las proyecciones *R1*, *R2* y *R3* si están en *5FN*.

Ejercicio: AGENCIAS DE MARKETING.

Hacer lo mismo que en el ejercicio anterior, obtengamos todas las proyecciones posibles y después hacer su unión natural.

Agencia	Empresa
Agencia 1	Empresa 1
Agencia 1	Empresa 2
Agencia 2	Empresa 1

Agencia	Servicio
Agencia 1	Servicio 1
Agencia 1	Servicio 2
Agencia 1	Servicio 3
Agencia 2	Servicio 3

Empresa	Servicio
Empresa 1	Servicio 1
Empresa 1	Servicio 2
Empresa 1	Servicio 3
Empresa 2	Servicio 3

Hacer la unión natural:

Agencia	Empresa	Servicio
Agencia 1	Empresa 1	Servicio 1
Agencia 1	Empresa 1	Servicio 2
Agencia 1	Empresa 2	Servicio 3
Agencia 2	Empresa 1	Servicio 3

Después de llevar a cabo la unión natural se ve que no se ha podido recuperar la relación original, esto quiere decir que no se ha cumplido y por lo tanto, la relación original está en la *5FN*, ya que no satisface la dependencia de unión natural y no es necesario realizar ningún tipo de división.

Agencia	Empresa	Servicio
Agencia 1	Empresa 1	Servicio 1
Agencia 1	Empresa 1	Servicio 2
Agencia 1	Empresa 1	Servicio 3
Agencia 1	**Empresa 2**	**Servicio 1**
Agencia 1	**Empresa 2**	**Servicio 2**
Agencia 1	Empresa 2	Servicio 3
Agencia 2	Empresa 1	Servicio 3

6.2.7 Forma Normal de Dominio/Clave (*DKFN*).

La forma normal de dominio/clave (*DKNF*) es una forma normal **usada en normalización de *BD* que requiere que la *BD* contenga restricciones de dominios y de claves**.

Una **restricción del dominio especifica los valores permitidos para un atributo** dado, mientras que una **restricción clave especifica los atributos que identifican únicamente una fila en una tabla dada**.

Esta es el santo grial de la *BD* y es alcanzado cuando cada restricción en la relación es una consecuencia lógica de la definición de claves y dominios, y, haciendo cumplir las restricciones y condiciones de la clave y del dominio, causa que sean satisfechas todas las restricciones. Así, esto evita todas las anomalías no-temporales.

Es mucho más fácil construir una *BD* en forma normal de dominio/clave que convertir pequeñas *BD* que puedan contener numerosas anomalías. Sin embargo, construir con éxito una *BD* en forma normal de dominio/clave sigue siendo una tarea difícil, incluso para programadores experimentados de *BD*. Así, mientras que la forma normal de dominio/clave elimina los problemas encontrados en la mayoría de las *BD*, tiende para ser la forma normal más costosa de alcanzar. Sin embargo, el no poder alcanzar la forma normal de dominio/clave puede llevar costos ocultos a largo plazo, debido a anomalías que aparecen con el tiempo en las *BD* que solamente se adhieren a formas normales más bajas.

Una violación de *DKNF* ocurre en la siguiente tabla:

Persona rica

DNI Persona rica	Tipo de persona rica	Valor neto en dólares
123	Millonario excéntrico	124,543,621
456	Multimillonario malvado	6,553,228,893
789	Multimillonario excéntrico	8,829,462,998
012	Millonario malvado	495,565,211

Asumir que el dominio para *DNI Persona rica* consiste en los *DNI's* de toda la gente rica en una muestra predefinida de gente rica; el dominio para el *Tipo de persona rica* consiste de los valores 'Millonario excéntrico', 'Multimillonario excéntrico', 'Millonario malvado', y 'Multimillonario malvado'; y el dominio para el *Valor neto en dólares* consiste de todos los números enteros mayor que o igual a 1.000.000.

Hay una restricción que liga el *Tipo de persona rica* al *Valor neto en dólares*, incluso aunque no se pueda deducir uno del otro. La restricción dicta que un *Millonario excéntrico* o *Millonario malvado* tendrá un valor neto de 1.000.000 a 999.999.999 inclusive, mientras que un *Multimillonario excéntrico* o un *Multimillonario malvado* tendrá un valor neto de 1.000.000.000 o más. Esta restricción no es ni una restricción de dominio ni una restricción de clave; por lo tanto no se puede confiar en las restricciones de dominio y las de clave para garantizar que una combinación anómala de *Tipo de persona rica* / *Valor neto en dólares* no tenga cabida en la *BD*.

La violación de la *DKNF* podría ser eliminada alterando dominio *Tipo de persona rica* para hacer que sea consistente con solo dos valores, 'Malvado' y 'Excéntrico' (el estatus de persona rica como un millonario o un multimillonario es implícito en su *Valor neto en dólares*, así que no se pierde ninguna información útil).

Persona rica

DNI Persona rica	Tipo de persona rica	Valor neto en dólares
123	Excéntrico	124,543,621
456	Malvado	6,553,228,893
789	Excéntrico	8,829,462,998
012	Malvado	495,565,211

Estado de riqueza

Estado	Mínimo	Máximo
Millonario	1,000,000	999,999,999
Multimillonario	1,000,000,000	999,999,999,999

DKNF es frecuentemente difícil de alcanzar en la práctica.

Ejercicio resuelto

Sea la siguiente tabla. Se pide normalizarla hasta *FNBC*.

COMPRA (cod_compra, cod_prod, nomb_prod, fecha, cantidad, precio, fecha_rec, cod_prov, nomb_prov, tfno)

- **Comprobar *1FN*:**

La tabla *COMPRA* está en *1FN* ya que todos sus atributos son atómicos y todos los atributos no clave dependen funcionalmente de la clave.

- **Comprobar *2FN*:**

¿Todo atributo depende de todo el conjunto de atributos que forman la clave primaria, o sólo de parte? Como se ve, existen atributos que dependen sólo de una parte de la clave, por lo que esta tabla no está en *2FN*.

Dependencias:
cod_prod → *nomb_prod*, y *cod_prod* es parte de la clave primaria.

Al no estar en *2FN*, se ha de descomponer la tabla *COMPRA* en:

COMPRA (cod_compra, cod_prod, fecha, cantidad, precio, fecha_rec, cod_prov, nomb_prov, tfno).
PRODUCTO (cod_prod, nomb_prod).

Una vez hecha esta descomposición, ambas tablas están en *2FN*. Todos los atributos no clave dependen de toda la clave primaria.

- **Comprobar *3FN*:**

PRODUCTO está en *3FN*, ya que por el número de atributos que tiene no puede tener dependencias transitivas. ¿*COMPRA* está en *3FN*? Hay que preguntarse si existen dependencias transitivas entre atributos no clave.

Dependencias:
cod_prov → *nomb_prov* (siendo *cod_prov* el código del proveedor y *nomb_prov* el nombre del proveedor)
cod_prov → *tfno*

COMPRA no está en *3FN* porque existen dependencias transitivas entre atributos no clave, por tanto, se ha de descomponer:

COMPRA (cod_compra, cod_prod, fecha, cantidad, precio, fecha_rec, cod_prov)
PROVEEDOR (cod_prov, nomb_prov, tfno)

- **Comprobar *FNBC*:**

PRODUCTO está en *FNBC*, ya que está en *3FN* y todo determinante es clave candidata.
COMPRA está en *FNBC*, ya que está en *3FN* y todo determinante es clave candidata.
PROVEEDOR está en *FNBC*, ya que está en *3FN* y todo determinante es clave candidata.

La tabla inicial *COMPRA* queda normalizada hasta *FNBC* del siguiente modo:

PRODUCTO (cod_prod, nomb_prod)
COMPRA (cod_compra, cod_prod, fecha, cantidad, precio, fecha_rec, cod_prov)
PROVEEDOR (cod_prov, nomb_prov, tfno)

6.3 Desnormalización.

Cada vez que se avanza en el nivel de normalización se necesitan más uniones entre las entidades y eso ralentiza la respuesta del sistema. Como la respuesta rápida a las demandas del usuario debe tenerse en cuenta a la hora del diseño de una *BD*, a veces es necesario desnormalizar alguna parte de la misma.

El proceso de denormalización **consiste en crear redundancia a propósito en casos concretos, donde perdemos eficiencia en almacenamiento para ganar rapidez en las consultas**.

> ***Desnormalizar*** *es transformar una BD en un nivel de normalización inferior. No obstante, es necesario tener en cuenta que a cambio de un desempeño más rápido deberemos soportar una mayor redundancia de datos.*

Los diseñadores de una *BD* deben conciliar tres requerimientos a menudo incompatibles entre sí:

- ✓ Un diseño apropiado.
- ✓ Velocidad de procesamiento.
- ✓ Almacenamiento de la información, controlando las redundancias.

En ocasiones es necesario aceptar una cierta redundancia para que la *BD* cumpla mejor con el objetivo de mostrar la información adecuadamente.

TAREA

15. Dada la siguiente tabla *EXPERIENCIA* en la que se guarda información sobre la antigüedad de los empleados por departamento en una empresa, normalizarla hasta *3FN*.

idEmpleado	Nombre	idDepartamento	nombreDepartamento	añosDepartamento
1	Juan	6	Contabilidad	6
2	Pedro	3	Sistemas	3
2	Pedro	6	Contabilidad	5
3	Sonia	2	I+D	1
4	Verónica	3	Sistemas	10
4	Verónica	6	Contabilidad	2

16. Dada la siguiente tabla en la que se guarda información sobre las viviendas que tiene una persona, normalizar hasta *3FN*. Suponer que en un mismo código postal no se puede tener más de una vivienda por persona.

DNI	Nombre	Apellidos	Dirección	codPostal	Población	Provincia
413245-B	Juan	Ramos	Las Cañas, 59	19005	Guadalajara	Guadalajara
413245-B	Juan	Ramos	Pitón, 12	45589	Caleruela	Toledo
23456-J	Pedro	Pérez	Vitoria, 3	28804	Alcalá de Henares	Madrid
23456-J	Pedro	Pérez	El Altozano	10392	Berrocalejo	Cáceres
34561-B	María	Rodríguez	Sanz Vázques, 2	19004	Guadalajara	Guadalajara
222346-J	Juan	Cabello	El ensanche, 3	28802	Alcalá de Henares	Madrid
222346-J	Juan	Cabello	Los abedules, 10	10300	Navalmoral de la Mata	Cáceres

17. Dada la siguiente relación *EMPLEADO* en la que se guardan los datos de los empleados de una fábrica, normalizar hasta *3FN*. Suponer que un empleado puede trabajar en diferentes departamentos y puestos a lo largo de su vida laboral en la empresa.

DNI	numSegSocial	Nombre	Apellidos	Departamento	Puesto	Salario	Antigüedad
413245-B	28-1234566	Juan	Ramos	Compras	Gerente	2.300	5
23456-J	28-2345686	Pedro	Pérez	Nóminas	Auxiliar	1.200	3
123123-C	19-458766	María	Gil	Almacén	Conserje	1.530	10
123455-B	45-223344	Antonio	Sanz	Compras	Gestión	2.200	7

18. Normalizar hasta *3FN* la siguiente relación *VENTA*, utilizada para almacenar información sobre los artículos que un dependiente vende, además de información del propio dependiente.

DNI	Calle	Ciudad	Comunidad	Código	cantidad
413245-B	Bravo Murillo	Madrid	Madrid	1	10
413245-B	Bravo Murillo	Madrid	Madrid	2	3
123123-C	Bravo Murillo	Barcelona	Cataluña	1	4
123455-B	Goya	Sevilla	Andalucía	3	7