Tema 2

DISEÑO LÓGICO DE BASES DE DATOS

1. Diseño de Bases de Datos

El **Diseño de Bases de Datos** es el proceso por el que se determina la organización de una Base de Datos, incluidas su estructura, contenido y las aplicaciones que se han de desarrollar.

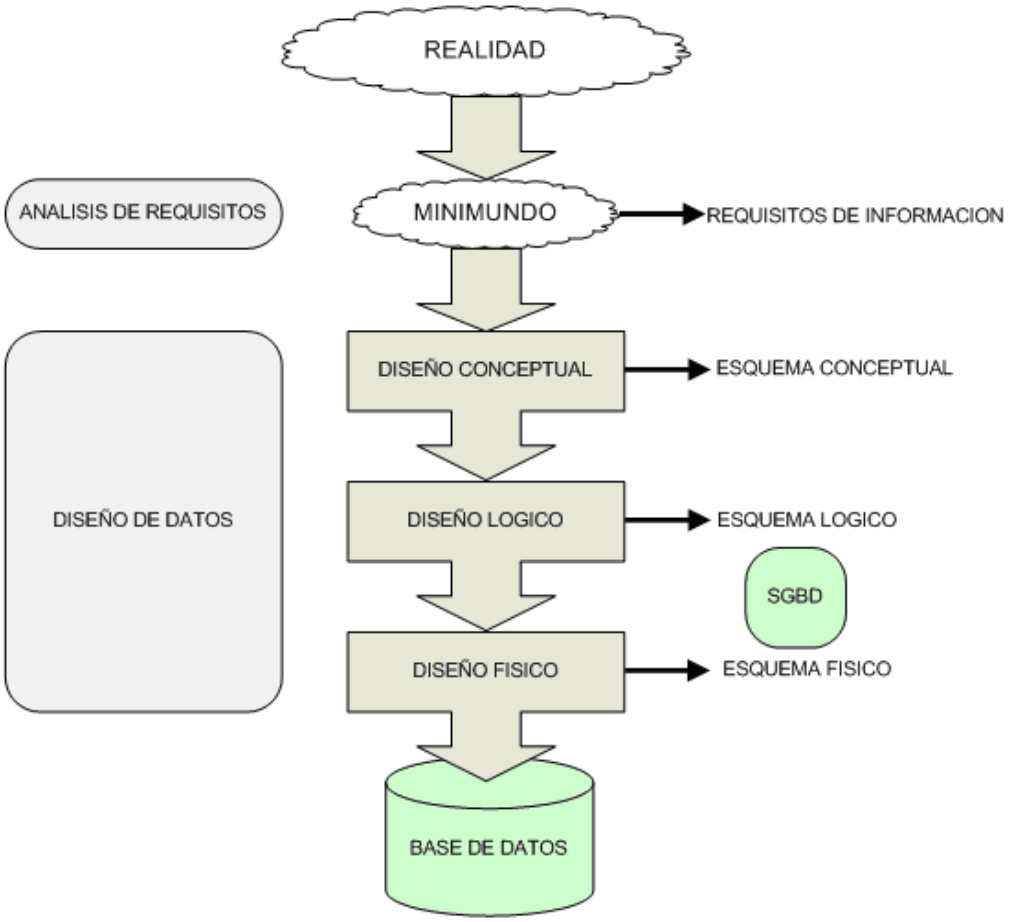
Durante mucho tiempo, el Diseño de Bases de Datos fue considerado una tarea para expertos: más un arte que una ciencia. Sin embargo, se ha progresado mucho en el Diseño de Bases de Datos y este se considera ahora una disciplina estable, con métodos y técnicas propios. Debido a la creciente aceptación de las Bases de Datos por parte de la industria y el gobierno en el plano comercial, y a una variedad de aplicaciones científicas y técnicas, el diseño de Base de Datos desempeña un papel central en el empleo de los recursos de información en la mayoría de las organizaciones. El diseño de Base de Datos ha pasado a constituir parte de la formación general de los informáticos, en el mismo nivel que la capacidad de construir algoritmos usando un lenguaje de programación convencional.

A finales de la década de los 60, cuando las Bases de Datos entraron por primera vez en el mercado del software, los diseñadores de Bases de Datos actuaban como artesanos, con herramientas muy primitivas: diagrama de bloques y estructuras de registros eran los formatos comunes para las especificaciones, y el Diseño de Bases de Datos se confundía frecuentemente con la implementación de las Bases de Datos. Esta situación ahora ha cambiado: los métodos y **modelos de Diseño de Bases de Datos** han evolucionado paralelamente con el progreso de la tecnología en los sistemas de Bases de Datos. Se ha entrado en la era de los **sistemas relacionales** de Bases de Datos, que ofrecen poderosos lenguajes de consultas, herramientas para el desarrollo de aplicaciones e interfaces amables con los usuarios. La tecnología de Bases de Datos cuenta ya con un marco teórico, que incluye la teoría relacional de datos, procesamiento y optimización de consultas, control de concurrencia, gestión de transacciones y recuperación, etc.

Según ha avanzado la tecnología de Bases de Datos, así se han desarrollado las metodologías y técnicas de diseño. Se ha alcanzado un consenso, por ejemplo, sobre la descomposición del proceso en fases, sobre los principales objetivos de cada fase y sobre las técnicas para conseguir estos objetivos.

El diseño de una Base de Datos es un proceso complejo que abarca decisiones a muy distintos niveles. La complejidad se controla mejor si se descompone el problema en subproblemas y se resuelve cada uno de estos subproblemas

independientemente, utilizando técnicas específicas. Así, el diseño de una Base de Datos se descompone en: diseño conceptual, diseño lógico y diseño físico.



- El **diseño conceptual** parte de las especificaciones de requisitos de usuario y su resultado es el esquema conceptual de la Base de Datos. Un esquema *conceptual* es una descripción de alto nivel de la estructura de la Base de Datos, independientemente del SGBD que se vaya a utilizar para manipularlo. Los procesos de definición de requisitos y del diseño conceptual exigen identificar las exigencias de información de los usuarios y representarlos en un modelo bien definido. Diseñaremos el esquema conceptual mediante el modelo Entidad-Relación.
- El **diseño lógico** es el proceso de construir un esquema de la información que utiliza la empresa, basándose en un modelo conceptual de base de datos específico, independiente del SGBD concreto que se

vaya a utilizar (salvo en el modelo) y de cualquier otra consideración física. En esta etapa, se transforma el esquema conceptual en un esquema lógico que utilizará las estructuras de datos del modelo de base de datos en el que se basa el SGBD que se vaya a utilizar, como puede ser el modelo relacional, el modelo de red, el modelo jerárquico o el modelo orientado a objetos. Conforme se va desarrollando el esquema lógico, éste se va probando y validando con los requisitos de usuario.

La *normalización* es una técnica que se utiliza para comprobar la validez de los esquemas lógicos basados en el modelo relacional, ya que asegura que las relaciones (tablas) obtenidas no tienen datos redundantes. Esta técnica se presenta en el siguiente tema (Tema 4. "Modelo Relacional").

El esquema lógico es una fuente de información para el diseño físico. Además, juega un papel importante durante la etapa de mantenimiento del sistema, ya que permite que los futuros cambios que se realicen sobre los programas de aplicación o sobre los datos, se representen correctamente en la base de datos.

> NOTA: Tanto el diseño conceptual, como el diseño lógico, son procesos iterativos, tienen un punto de inicio y se van refinando continuamente. Ambos se deben ver como un proceso de aprendizaje en el que el diseñador va comprendiendo el funcionamiento de la empresa y el significado de los datos que maneja. El diseño conceptual y el diseño lógico son etapas clave para conseguir un sistema que funcione correctamente. Si el esquema no es una representación fiel de la empresa, será difícil, sino imposible, definir todas las vistas de usuario (esquemas externos), o mantener la integridad de la base de datos. También puede ser difícil definir la implementación física o el mantener unas prestaciones aceptables del sistema. Además, hay que tener en cuenta que la capacidad de ajustarse a futuros cambios es un sello que identifica a los buenos diseños de bases de datos.

- El **diseño físico** es el proceso de producir la descripción de la implementación de la base de datos en memoria secundaria: estructuras de almacenamiento y métodos de acceso que garanticen un acceso eficiente a los datos.

  Para llevar a cabo esta etapa, se debe haber decidido cuál es el SGBD que se va a utilizar, ya que el esquema físico se adapta a él. Entre el diseño físico y el diseño lógico hay una realimentación, ya que algunas de las decisiones que se tomen durante el diseño físico para mejorar las prestaciones, pueden afectar a la estructura del esquema lógico.

En general, el propósito del diseño físico es describir cómo se va a implementar físicamente el esquema lógico obtenido en la fase anterior. Concretamente, en el modelo relacional, esto consiste en:

- Obtener un conjunto de relaciones (tablas) y las restricciones que se deben cumplir sobre ellas.
- Determinar las estructuras de almacenamiento y los métodos de acceso que se van a utilizar para conseguir unas prestaciones óptimas.
- Diseñar el modelo de seguridad del sistema.

2. Qué es un modelo

Un **modelo** es una representación de la realidad que conserva sólo los detalles relevantes.

Por ejemplo, consideremos una transacción bancaria tal como un depósito en una cuenta corriente. El departamento de contabilidad desea conservar ciertos detalles (número de cuenta, importe del depósito, fecha, número del cajero) e ignorar otros (las palabras que se han intercambiado durante la transacción, la longitud de la cola, la temperatura ambiental dentro del banco,…).

La realidad involucra un sin número de detalles, pero el departamento de contabilidad considerará la mayoría de ellos irrelevantes para la transacción. De modo que un modelo, desde el punto de vista del departamento de contabilidad, deberá considerar sólo aquellos detalles que este considere relevantes. Por supuesto, algunos detalles considerados irrelevantes para un usuario o grupo de usuarios pueden ser relevantes para otros. Ejemplo: la longitud de la cola puede ser interesante para el director del banco en el sentido de contratar a más cajeros para atender al público. Por tanto, diferentes usuarios o grupos de usuarios pueden tener distintos modelos de la realidad.

Una Base de Datos incorpora un modelo de la realidad. El SGBD gestiona la Base de Datos de modo que cada usuario pueda registrar, acceder y manipular los datos que constituyen su modelo de la realidad. Manipulando los datos los usuarios pueden obtener información necesaria que les sea útil en su vida. Por tanto, los modelos son herramientas poderosas para eliminar los detalles irrelevantes y comprender la realidad de los usuarios individuales.

Para modelar debemos asociar/identificar elementos de la realidad con elementos del modelo. Si esta asociación se hace correctamente, entonces el

modelo se puede usar para resolver el problema. De lo contrario, el modelo probablemente conducirá a una solución incompleta o incorrecta.

3. El modelo Entidad-Relación (Entidad-Interrelación)

El **modelo Entidad-Relación** es un modelo conceptual de datos orientado a entidades. Se basa en una técnica de representación gráfica que incorpora información relativa a los datos y las relaciones existentes entre ellos, para darnos una visión de mundo real, eliminando los detalles irrelevantes.

El modelo Entidad-Relación (E-R) fue propuesto por **Peter Chen** en 1976 en un artículo muy famoso actualmente: "The Entity-Relationship Model: Toward a Unified View of Data".

Según Chen: "El Modelo Entidad-Interrelación puede ser usado como una base para una vista unificada de los datos", adoptando "el enfoque más natural del mundo real que consiste en entidades y relaciones (interrelaciones)". Posteriormente, otros autores han ampliado el modelo (modelo entidad-relación extendido), con importantes aportaciones, formándose en realidad una familia de modelos.

Este tema describe el Modelo Entidad-Relación, sin discriminar de manera detallada los elementos originales y los extendidos. El objetivo es disponer de un buen modelo para representar datos de cara a diseñar bases de datos.

CARACTERÍSTICAS DEL MODELO

- Refleja tan solo la existencia de los datos, no lo que se hace con ellos.
- Se incluyen todos los datos relevantes del sistema en estudio.
- No está orientado a aplicaciones específicas.
- Es independiente de los SGBD.
- No tiene en cuenta restricciones de espacio, almacenamiento, ni tiempo de ejecución.
- Está abierto a la evolución del sistema.
- Es el modelo conceptual más utilizado.

ELEMENTOS DEL MODELO

Los elementos básicos del modelo E-R original son:

- ENTIDAD (*entity*)
- ATRIBUTO (*attribute*)
- DOMINIO (*domain*)
- RELACION (*relationship*)

A lo largo de este tema describiremos esos elementos básicos:

4. Entidades

Entidad: Cualquier objeto (real o abstracto) que existe en la realidad y acerca del cual queremos almacenar información en la B.D.

"Algo con realidad objetiva que existe o puede ser pensado" (Hall, 1976).

Las entidades poseen un predicado asociado que hace que los ejemplares lo cumplen. Las entidades se representan gráficamente mediante rectángulos con su nombre en el interior.

Ejemplo: La entidad PROFESOR, cuyo predicado asociado es "persona que enseña una materia", tiene un ejemplar 'Juana' que pertenece a ese tipo de entidad, ya que cumple dicho predicado (o al menos lo intenta ;)

Ejemplo: MATERIA es una entidad. 'Fundamentos de Programación', 'Inglés' y 'Física' son ocurrencias de la entidad MATERIA.

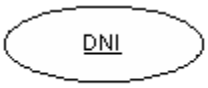
Ejemplo: POBLACION es una entidad. 'Jerez', 'Barcelona', 'Jimena', 'Mérida' son ocurrencias de la entidad POBLACION.

5. Atributos

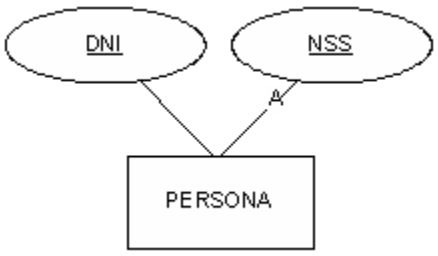
Atributo: Cada una de las propiedades o características que tiene una entidad.

Los atributos se representan mediante un óvalo con el nombre del atributo dentro. Pueden clasificarse según:

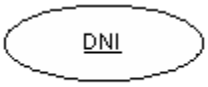
- **Identificadores:** son atributos *que identifican de manera unívoca cada ocurrencia de una entidad*. Toda entidad debe tener al menos un atributo identificador.



Identificador **primario** e identificadores **alternativos**: Una entidad puede tener más de 1 atributo identificador; en ese caso, elegimos un atributo como identificador primario (P), quedando el resto como identificadores alternativos (A).



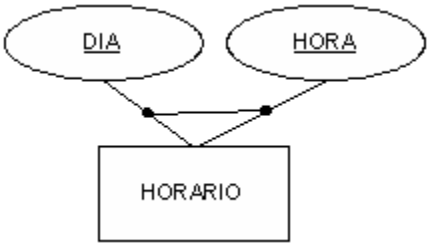
Los atributos identificadores **simples** se representan subrayando el nombre del atributo:



Los atributos identificadores **compuestos** se pueden representar de dos formas (a elegir):

Forma 1

Forma 2



- **Simples y compuestos**

Simples: son atributos que no están formados por otros atributos.

Compuestos: son atributos que están formados por otros atributos que a su vez pueden ser simples o compuestos.

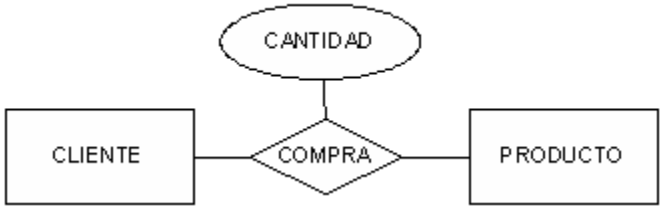
- **Monovaluados y multivaluados**

  Monovaluados: son atributos que representan un solo valor para una determinada ocurrencia de una entidad en un momento determinado. Pueden ser simples o compuestos.

  Multivaluados: son atributos que pueden representar varios valores simultáneamente para una misma ocurrencia de una entidad. Se representan mediante un doble óvalo.

- **Derivados (o calculados):** son atributos cuyo valor se obtiene aplicando una fórmula (normalmente a partir del valor de otros atributos). Son atributos que a la postre no se almacenarán en la base de datos. Su valor se obtendrá en el momento en que sea necesario aplicando la fórmula asociada a ellos. En el diccionario de datos debe especificarse esta fórmula o método para calcular su valor. Se representan en un diagrama ER mediante un óvalo con línea discontinua.

- **Propios:** son los atributos de las relaciones. Se representan unidos al rombo de la relación.



CARDINALIDADES DE ATRIBUTOS

Para cada atributo de una entidad se puede especificar una cardinalidad (min,max); la cual indicará cuantos valores puede almacenar el atributo para una ocurrencia determinada de la entidad.

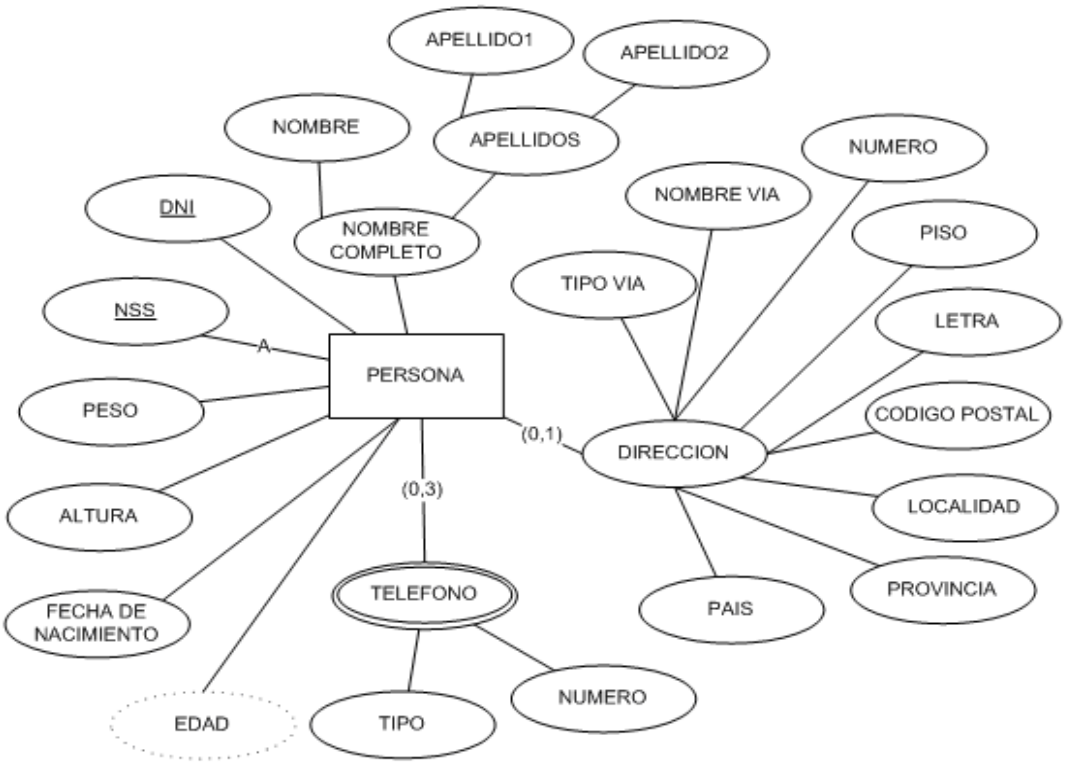
Por defecto (si no ponemos nada), la cardinalidad de un atributo asociado a una entidad es (1,1); es decir, el atributo debe obligatoriamente tener exactamente un valor para toda ocurrencia de la entidad.

Para atributos multivaluados la cardinalidad por defecto es (1,n).

Pondremos como cardinalidad de atributo (0,1) si queremos indicar que un atributo puede contener un valor nulo (NULL).

Para atributos compuestos, si no especificamos nada, entonces es obligatorio que tengan valor todos sus atributos componentes. Si especificamos una cardinalidad para un atributo compuesto pero no para los atributos componentes, entonces todos los atributos componentes "heredan" esa cardinalidad. Por ejemplo: si pusiéramos (0,1) como cardinalidad del atributo DIRECCION, entonces todos sus componentes podrían contener un valor nulo.

Para atributos multivaluados podemos especificar un rango finito. Por ejemplo: para el atributo multivaluado compuesto TELEFONO podemos decir que su cardinalidad es (0,3), de tal manera indicamos que una persona puede tener de 0 a 3 teléfonos como máximo.



6. Dominios

Dominio: Conjunto de valores homogéneos con un nombre que lo identifica. Cada atributo simple de una entidad está asociado a un dominio, el cual representa el conjunto de valores que puede tomar el atributo. Para cada ocurrencia de una entidad un atributo tendrá un valor perteneciente al dominio del atributo.

Varios atributos distintos (incluso de distintas entidades o relaciones) pueden pertenecer al mismo dominio. Un dominio lleva siempre asociado un predicado que permite comprobar si un determinado valor pertenece al dominio.

$$D = \{v_i \mid P(v_i)\}$$

Ejemplo: DDNI= $\{v_i \mid P(v_i)\}$ donde P(v) indica que v es un documento nacional de identidad con la letra de un ciudadano español.

Para cada dominio especificaremos el tipo de datos al cual pertenecen los valores que constituyen el mismo. Asimismo, podremos especificar el formato y la unidad correspondientes.

Los dominios se especificarán en el diccionario de datos.

Es obligatoria la especificación del nombre del dominio, el tipo y la descripción

Es opcional la especificación del formato y la unidad.

El formato se especificará acorde a la siguiente notación:

Tipo	Fórmula
CONCATENACIÓN	Componente1 + Componente2
DISYUNCIÓN	[Componente1|Componente2]
OPCIONALIDAD	(Componente)
REPETICIÓN	$\{Componente\}_{min,max}$ $\{Componente\}_{x}$ (ponemos x si min=max)

Ejemplo de definición de dominios en el DICCIONARIO DE DATOS del esquema conceptual:

Dominio	Tipo	Formato	Unidad	Valores	Descripcón
DDNI	Cadena(9)	$\{Dígito\}_8$+{Letra}			Números de Documento Nacional de Identidad (con la letra) de ciudadanos españoles.
DNSS	Cadena(12)	{Provincia}+$\{Dígito\}_1$ {Provincia}=$\{Dígito\}_2$			Número de la Seguridad Social de España
DNOMBRE	Cadena(30)	$\{Letra\}_{1,30}$			Nombres de personas

DAPELLIDO	Cadena(40)	{Letra}$_{1,40}$			Apellidos de personas
DPESO	Número	{Dígito}$_{1,3}$	Kg.		Pesos de personas
DALTURA	Número	{Dígito}$_{1,3}$	cm.		Alturas de personas
DTELTIPO	Cadena(5)	{Letra}$_{3,5}$		'FIJO' 'MOVIL' 'FAX'	Tipos de teléfonos
DTELNUMERO	Número	{Dígito}$_{9}$			Números de teléfono de España
...	...	...	...	...	...
DEDAD (*)	Número	{Dígito}$_{1,3}$	Años	*FechaActual-FechaNacimiento*	Edades de personas
...	...	...	...	...	...

7. Relaciones (Interrelaciones)

Relación (interrelación, vínculo): es una correspondencia o asociación entre 2 o más entidades.

Las relaciones se representan gráficamente mediante rombos y su nombre aparece en el interior. Normalmente son verbos o formas verbales.

Matemáticamente una relación se puede representar de la siguiente manera:

$\{<e_1, e_2, \ldots, e_n>\}$

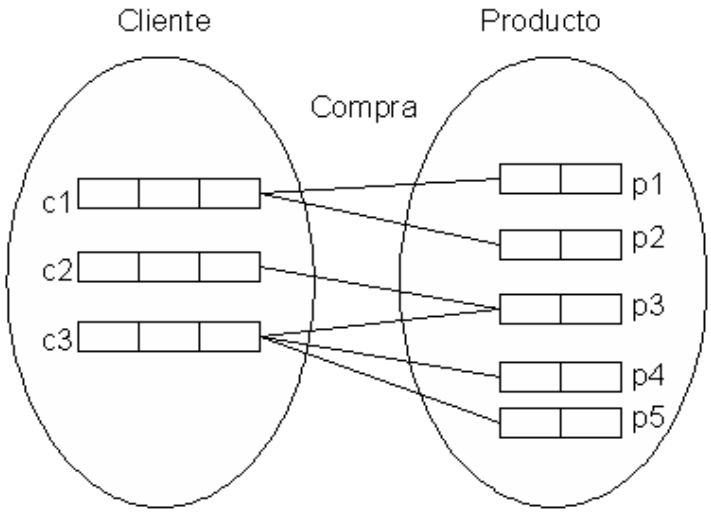
donde

e_i=ejemplares de la entidad e_i

n=grado de la relación

En el siguiente ejemplo la relación sería:

Compra = {<c1, p1>, <c1, p2>, <c2, p3>, <c3, p3>, <c3, p4>,<c3, p5>}



TIPOS DE CORRESPONDENCIAS (CARDINALIDAD DE LA RELACION)

Cardinalidad: la cardinalidad de una relación es el número de ocurrencias de una entidad asociadas a una ocurrencia de la otra entidad.

Existen tres tipos de correspondencias:

Uno a uno (1:1)

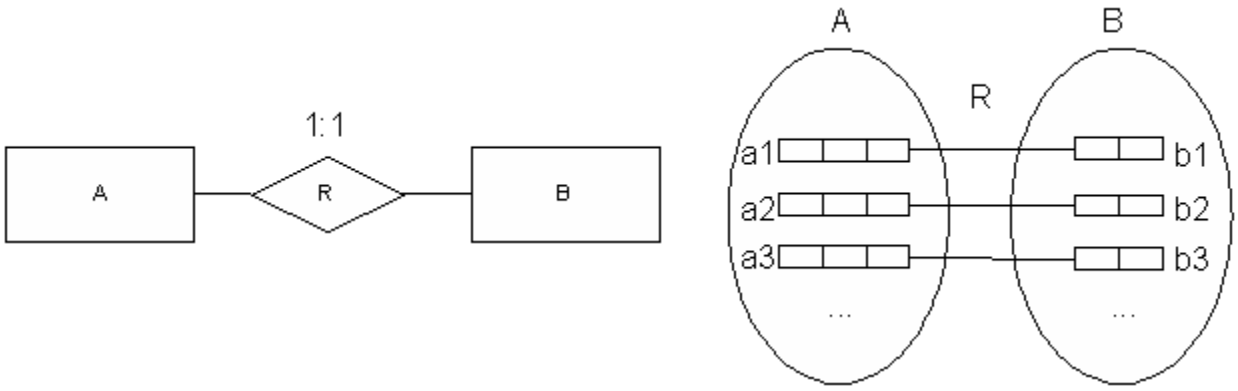
Uno a muchos (1:N)

Muchos a muchos (N:N)

Supongamos 2 entidades A y B unidas mediante la relación R. La cardinalidad se coloca sobre la relación R.

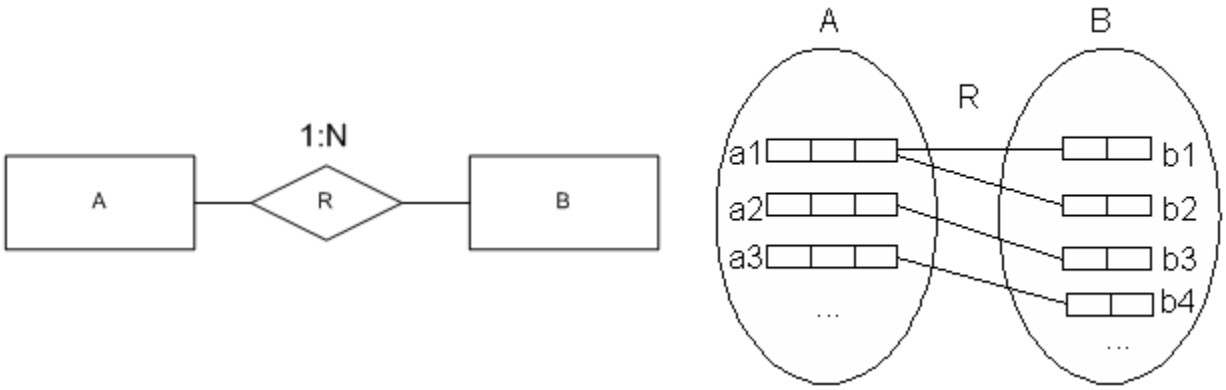
Uno a uno (1:1)

A cada ocurrencia de la entidad A le corresponde una ocurrencia de la entidad B, y viceversa.



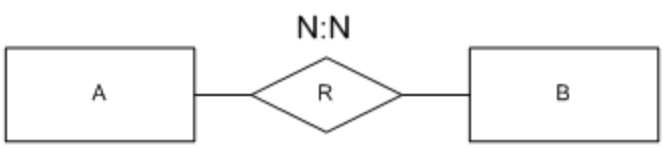
Uno a muchos (1:N)

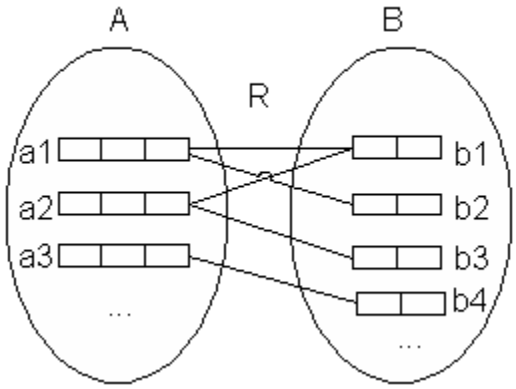
A cada ocurrencia de la entidad A le pueden corresponder varias ocurrencias de la entidad B. Pero a cada ocurrencia de la entidad B sólo le corresponde una ocurrencia de la entidad A.



Muchos a muchos (N:N)

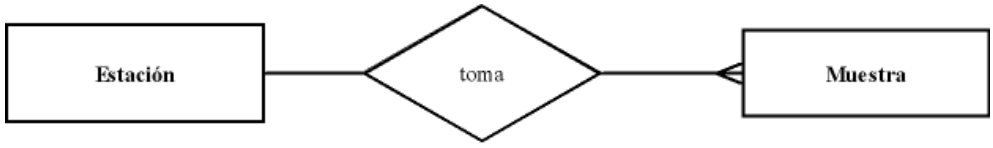
A cada ocurrencia de la entidad A le pueden corresponder varias ocurrencias de la entidad B. Y a cada ocurrencia de la entidad B le pueden corresponder varias ocurrencias de la entidad A.





Para obtener la cardinalidad de una relación, se debe fijar una ocurrencia en concreto de una entidad y averiguar cuántas ocurrencias de la otra entidad le corresponden. Después, realizar lo mismo en el otro sentido.

NOTA: Otra forma de representar las relaciones a muchos (N) es indicando en el extremo de la línea que une las interrelaciones una "ramificación". Por ejemplo:



PARTICIPACIÓN DE LAS ENTIDADES EN LAS RELACIONES

Cada entidad podrá participar en la relación con un mínimo y un máximo de ocurrencias.

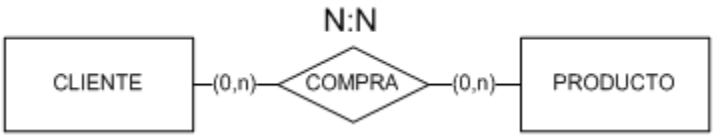
Para obtener las participaciones fijamos una ocurrencia en una entidad A y calculamos con cuantas ocurrencias de la entidad B se puede relacionar como mínimo y cómo máximo; posteriormente, hacemos lo mismo al revés.

Ejemplo 1:

Un profesor es tutor de 0 a n alumnos y un alumno tiene exactamente 1 tutor (de 1 a 1)

Ejemplo 2:

Un cliente puede comprar de 0 a n productos y un producto puede ser comprado por de 0 a n clientes.



Para obtener el tipo de correspondencia y consecuentemente las cardinalidades de la relación, se miran los máximos de las participaciones.

Especial atención requieren las participaciones mínimas:

- Participación mínima cero: significa que puede haber ocurrencias de una entidad que no estén asociadas a ninguna ocurrencia de la otra entidad.
- Participación mínima uno: significa que toda ocurrencia de una entidad debe estar asociada a una ocurrencia de la otra entidad.

En el ejemplo 1 anterior, un profesor puede no ser tutor de ningún alumno (participación mínima 0). Mientras que un alumno tendrá siempre un tutor (participación mínima 1).

GRADO DE UNA RELACION

Grado de una relación: Es el número de entidades que participan en la relación.

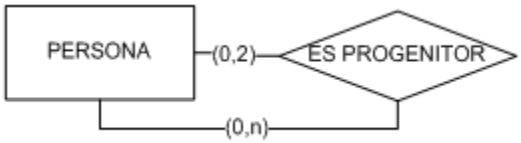
TIPOS DE RELACIONES

Las relaciones pueden ser REFLEXIVAS, BINARIAS, TERNARIAS, ... según su grado y FUERTES-DÉBILES según su dependencia.

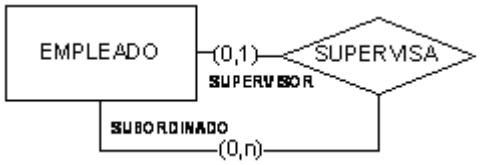
REFLEXIVAS (GRADO 1)

Son relaciones donde participa sólo 1 entidad. Se relacionan ocurrencias de la entidad con otras ocurrencias de la propia entidad.

Ejemplo 1: Es progenitor



Ejemplo 2: En este tipo de relaciones reflexivas se suelen especificar roles. Un rol es el papel que desempeña una ocurrencia de una entidad participante en una relación.



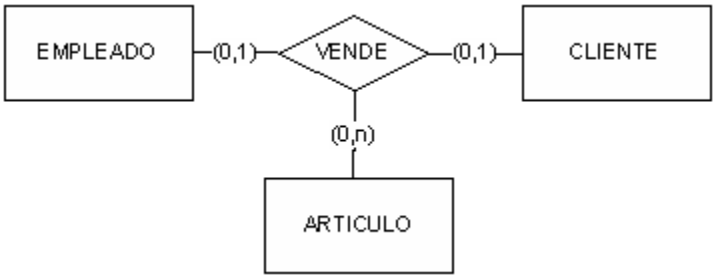
BINARIAS (GRADO 2)

Son relaciones donde participan 2 entidades.

TERNARIAS (GRADO 3)

Son relaciones donde participan 3 entidades. Para calcular las participaciones mínimas y máximas se compara un par de ocurrencias (a,b) de las entidades A y B con una ocurrencia c de la entidad C (y así con las otras 2 combinaciones).

Ejemplo: Empleados de un supermercado que venden artículos a clientes.



Para obtener las participaciones hemos pensado:

1. Una pareja (ocurrencias de las entidades EMPLEADO y CLIENTE) determinada (empleado,cliente) puede relacionarse con cómo mínimo **0** artículos y como máximo con **n**. Es decir, "un empleado puede venderle a un cliente entre 0 y n artículos". Ejemplo: ver filas 1 y 2 de la tabla.
2. Una pareja (ocurrencias de las entidades EMPLEADO y ARTICULO) determinada (empleado,artículo) puede relacionarse con cómo mínimo **0** clientes y como máximo con **1**. Es decir, "un empleado puede vender un artículo cómo mínimo a 0 clientes (no vende ese artículo nunca) y como máximo 1" (puede vender el artículo determinado una sola vez y sólo a un cliente; no puede vender el mismo artículo a varios clientes a la vez). Ejemplo: ver filas 1-4 de la tabla.
3. Una pareja (ocurrencias de las entidades CLIENTE y ARTICULO) determinada (cliente,artículo) puede relacionarse con cómo mínimo **0** empleados y como máximo con **1**. Es decir, "a un cliente le puede vender un artículo cómo mínimo a 0 empleados (no compra ese artículo nunca) y como máximo 1" (puede comprar el artículo determinado una sola vez y sólo a un empleado determinado; el artículo se lo vende un empleado -no varios- a un cliente)". Ejemplo: ver filas 1-4 de la tabla.

Ejemplo de ocurrencias de la relación VENDE (artículos):

FILA	EMPLEADO	CLIENTE	ARTICULO
1	E1	C1	A1
2	E1	C1	A2
3	E1	C2	A3
4	E1	C2	A4
~~5~~	~~E1~~	~~C2~~	~~A4~~

La ultima ocurrencia (E1, C2, A4) de la relación no podría existir en el caso de que el artículo A4 ya hubiera sido vendido (los artículos son objetos físicos independientes y únicos). Una vez que un empleado venda un artículo a un cliente ya no puede vender ese mismo artículo.

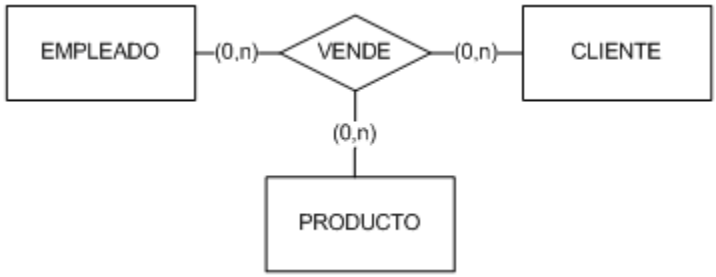
Ejemplo de relación ternaria (con PRODUCTO en vez de ARTICULO)

En este caso las participaciones varían, ya que:

1) Una pareja determinada empleado-cliente puede estar relacionada con de 0 a n productos (un mismo empleado puede vender a un mismo cliente varios productos). Ejemplo: ver filas 3 y 4 de la tabla.

2) Una pareja determinada empleado-producto puede estar relacionada con de 0 a n clientes (el mismo empleado puede vender el mismo producto a varios clientes en distintas ocasiones). Ejemplo: ver filas 4 y 5 de la tabla.

3) Una pareja determinada producto-cliente puede estar relacionada con de 0 a n empleados (el mismo producto puede haber sido vendido al mismo cliente por varios empleados en distintas ocasiones). Ejemplo: ver filas 1 y 2 de la tabla.



Ejemplo de ocurrencias de la relación VENDE (productos):

FILA	EMPLEADO	CLIENTE	PRODUCTO
1	E1	C1	P1
2	E2	C1	P2
3	E1	C2	P3
4	E1	C2	P4
5	E1	C3	P4

FUERTE-DÉBIL

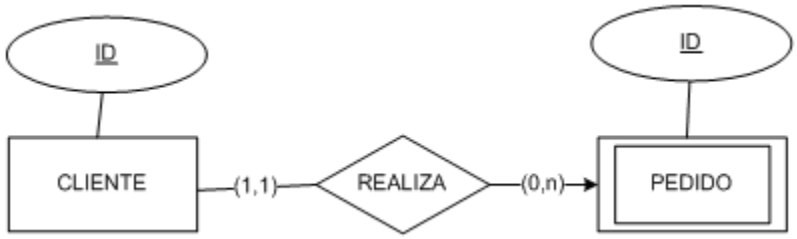
Cuando una entidad participa en una relación puede adquirir un papel fuerte o débil.

- **Dependencia de Existencia**

  Una entidad débil queda definida siempre a través de una relación especial que representa la dependencia de esta entidad de otra de orden superior (que puede ser a su vez una entidad fuerte o débil). Toda entidad débil tiene una dependencia en existencia de la entidad de orden superior, definiéndose entre ellas una jerarquía de dos niveles.

  Una instancia de la entidad débil está vinculada a una instancia de la entidad de orden superior, de modo que no puede existir sin ella; es decir para existir la débil, debe existir previamente la de orden superior y si desaparece la instancia de orden superior, entonces deben desaparecer todas las instancias de la entidad débil que están vinculadas.

  Las entidades débiles se representan mediante un doble rectángulo, es decir, un rectángulo con doble línea.



a) No puede existir una ocurrencia de un pedido si no se conoce el cliente.

b) Un pedido no puede estar vinculado a varios clientes. Sólo corresponde a uno.

c) Un cliente puede tener de 0 a n pedidos realizados.

d) Si se elimina la instancia de un cliente, no pueden existir las ocurrencias de pedidos que tenía vinculadas.

e) La flecha está orientada de la entidad de orden superior (CLIENTE) a la entidad débil en existencia (PEDIDO).

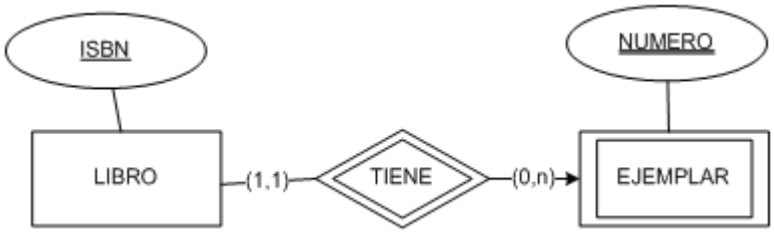
f) Un pedido queda identificado de manera unívoca por el identificador del pedido (ID), de modo que no pueden existir dos pedidos con el mismo identificador.

- **Dependencia de Identificación**

  Existen algunas entidades débiles que no tienen suficientes atributos para garantizar la identificación o distinción de sus ocurrencias. En estos casos es necesario forzar el mecanismo de identificación de dicha entidad débil con la composición de atributos primarios de la entidad de orden superior y algunos atributos de la entidad débil. Una dependencia en identificación implica también dependencia en existencia.

  La DEPENDENCIA EN IDENTIFICACION se representa mediante una relación débil (rombo con línea doble) y una entidad débil (rectángulo con línea doble). La flecha hacia la entidad débil es opcional.

Ejemplo 1:



Para poder identificar unívocamente las ocurrencias de la entidad EJEMPLAR necesitamos el identificador de la entidad fuerte LIBRO (ISBN) y el identificador de la entidad débil (NUMERO).

El par de atributos <ISBN, NUMERO> sería capaz de identificar unívocamente todos los ejemplares de todos los libros. Tengamos en cuenta que muchos libros pueden tener el ejemplar número 1 (siendo ejemplares distintos de libros distintos).

El identificador (débil) de la entidad débil ***en la dependencia de identificación*** lo representamos mediante un óvalo con el nombre del atributo doblemente subrayado.

Si eliminamos un libro desaparecen los ejemplares de ese libro ("Una dependencia en identificación implica también dependencia en existencia").

8. Ejemplo (Modelo Entidad Relación)

Describir del proceso

Se trata de una base de datos que debe almacenar la información sobre varias estaciones meteorológicas, en una zona determinada. De cada una de ellas recibiremos y almacenaremos un conjunto de datos cada día: temperatura máxima y mínima, precipitaciones en litros/m^2, velocidad del viento máxima y mínima, y humedad máxima y mínima. El sistema debe ser capaz de seleccionar, añadir o eliminar estaciones. Para cada una almacenaremos un identificador, su situación geográfica (latitud, longitud) y su altitud.

Identificar conjuntos de entidades

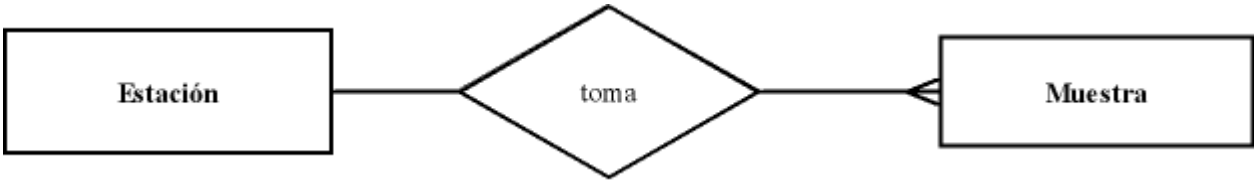
A primera vista, tenemos dos conjuntos de entidades: estaciones y muestras. Podríamos haber usado sólo un conjunto, el de las muestras, pero nos dicen que debemos ser capaces de seleccionar, añadir y borrar estaciones, de modo que parece que tendremos que usar un conjunto de entidades para ellas.

Identificar conjuntos de interrelaciones

Las relaciones son más simples, ya que sólo hay una: cada estación estará interrelacionada con varias muestras. Es una relación 1:N.

Trazar primer diagrama

Podemos trazar ya, por lo tanto, nuestro primer diagrama:



Identificar atributos

El siguiente paso es identificar los atributos para cada conjunto de entidades.

Para las muestras tendremos que elegir los que nos da el enunciado: temperatura máxima y mínima, precipitaciones, velocidades del viento máxima

y mínima y humedad máxima y mínima. Además hay que añadir la fecha de la muestra.

Para las estaciones también nos dicen qué atributos necesitamos: identificador, latitud, longitud y altitud.

Seleccionar claves principales

Las estaciones disponen de varias claves candidatas. Tenemos, por una parte, el identificador, que es único para cada estación, y por otra su situación geográfica, ya que no puede haber dos estaciones en el mismo sitio. Parece lógico usar la primera como clave principal, ya que es un único atributo.

Pero en el caso de las muestras no existen claves candidatas claras. De hecho, el conjunto total de atributos puede no ser único: dos estaciones próximas geográficamente, podrían dar los mismos datos para las mismas fechas.

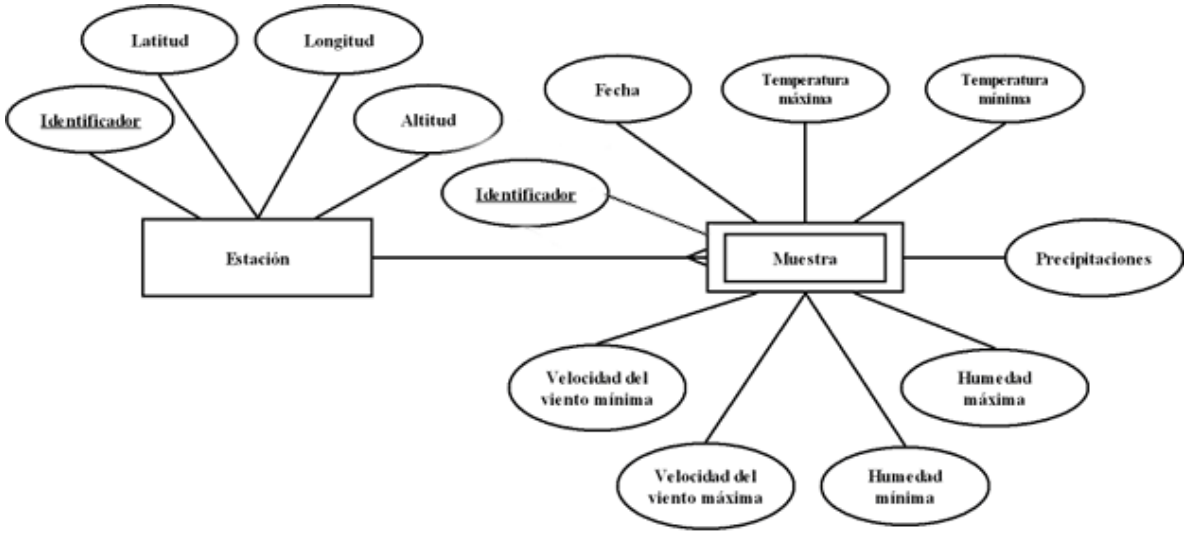
Tenemos una opción para solucionar el problema: crear una clave principal artificial, un número entero que se incremente de forma automática para cada muestra.

Otra alternativa es considerar las muestras como entidades débiles subordinadas a las entidades estación. En ese caso, la clave primaria de la estación se almacena como una clave foránea en cada muestra.

Como entidad débil, las muestras no necesitan una clave primaria, de hecho, esa clave se forma con la unión de la clave primaria de la estación y la fecha de la muestra.

En la mayoría de los casos, y con el fin de prever futuras ampliaciones, optaremos por aplicar las dos soluciones.

Verificar el modelo



Diccionario de Datos

ENTIDADES

ESTACION. Estaciones Meteorológicas

ATRIBUTOS	DOMINIOS	DESCRIPCION	DEFECTO	RESTRICCIONES
identificador	DIDENTIFICADOR	Identificador de Estación		Auto Incremento, sin signo
latitud	DLATITUD	Distancia al Ecuador		No nulo
longitud	DLONGITUD	Distancia al Meridiano de Greenwich		No nulo
altitud	DALTURA	Altura con respecto al nivel del mar		No nulo

MUESTRA. Muestras tomadas en distintas fechas para cada estación meteorológica

ATRIBUTOS	DOMINIOS	DESCRIPCION	DEFECTO	RESTRICCIONES
identificador	DIDENTIFICADOR	Identificador de Muestra		Auto Incremento, sin signo
identificador_estacion	DIDENTIFICADOR	Identificador de Estación		Debe existir en ESTACION
fecha	DFECHA	Fecha de la muestra	Fecha actual	No nulo
temperaturaminima	DTEMPERATURA	Temperatura mínima		
temperaturamaxima	DTEMPERATURA	Temperatura máxima		
precipitaciones	DPRECIPITACIONES	Precipitaciones		Sin signo

humedadminima	DHUMEDAD	Humedad mínima		Sin signo
humedadmaxima	DHUMEDAD	Humedad máxima		Sin signo
velocidadminima	DVELOCIDAD	Velocidad del viento mínima		Sin signo
velocidadmaxima	DVELOCIDAD	Velocidad del viento máxima		Sin signo

DOMINIOS

DOMINIO	TIPO	FORMATO	UNIDAD	VALORES	DESCRIPCION
DIDENTIFICADOR	MEDIUMINT	{Dígitos}$_{1,5}$			Claves Primarias
DLATITUD	VARCHAR	{Dígitos}$_2$+{Letra} {Letra}=N | S	Grados-Orientación		Grados con respecto al Ecuador (Norte o Sur) Valores de 0 a 90 en grados N o S en la letra
DLONGITUD	VARCHAR	{Dígitos}$_3$+{Letra} {Letra}=E | W	Grados-Orientación		Grados con respecto al meridiano Greenwich (Este o Oeste –W -) Valores de 0 a 180 en grados E o W en la letra
DALTURA	MEDIUMINT	{Dígitos}$_{1,6}$	Pies (1 pie=30.48 cm)		Altura con respecto al nivel del mar
DFECHA	DATE	{aaaa-mm-dd}			
DTEMPERATURA	TINYINT	{Dígitos}$_{1,2}$	Grados centígrados		Valores de temperatura
DPRECIPITACIONES	SMALLINT	{Dígitos}$_{1,3}$	Litros por metro cuadrado		Valores de precipitaciones
DHUMEDAD	TINYINT	{Dígitos}$_{1,2}$	Porcentaje		Porcentaje de humedad en el aire
DVELOCIDAD	SMALLINT	{Dígitos}$_{1,3}$	Kilómetros por hora		Velocidad del viento

9. El modelo relacional

El modelo entidad-relación es un modelo conceptual que sirve para cualquier tipo de SGBD, en cambio, el modelo relacional es un modelo lógico que sólo sirve para SGBD relacionales (y no para jerárquicos, o Codasyl, por ejemplo).

Todos los diseñadores y administradores de bases de datos relacionales usan esquemas conceptuales entidad-relación porque se adaptan muy bien a este modelo.

Hay que tener en cuenta la diferencia de la palabra **relación** en ambos modelos. En el modelo relacional una relación es una tabla mientras que en el entidad/relación es la asociación que se produce entre dos entidades.

RELACIÓN (TABLA)

Según el modelo relacional el elemento fundamental es lo que se conoce como **relación**, aunque más habitualmente se le llama **tabla**. Se trata de una estructura formada por filas y columnas que almacena los datos referentes a una determinada entidad o relación del mundo real.

NOMBRE

atributo 1	atributo 2	atributo 3	....	atributo n	
valor 1,1	valor 1,2	valor 1,3	....	valor 1,n	← tupla 1
valor 2,1	valor 2,2	valor 2,3	....	valor 2,n	← tupla 2
.....	.....	.....	....	.....	....
valor m,1	valor m,2	valor m,3	....	valor m,n	← tupla m

Acerca de una tabla, además de su nombre, podemos distinguir lo siguiente:

- **Atributo**

Representa una propiedad que posee esa tabla. Equivale al atributo del modelo E-R. Se corresponde con la idea de **campo o columna**.

- **Tupla**

Cada una de las filas de la tabla. Se corresponde con la idea de **registro**. Representa por tanto cada elemento individual (ejemplar, ocurrencia) de esa tabla.

- **Dominio**

Un dominio contiene todos los posibles valores que puede tomar un determinado atributo. Dos atributos distintos pueden tener el mismo dominio. Un domino en realidad es un conjunto finito de valores del mismo tipo. Los dominios poseen un nombre para poder referirnos a él y así poder ser reutilizable en más de un atributo.

- **Grado**

Número de columnas de la tabla (número de atributos).

- **Cardinalidad**

Número de tuplas de una tabla (número de filas).

CLAVE

- **Clave candidata**

Conjunto de atributos que identifican unívocamente cada tupla de la relación. Es decir columnas cuyos valores no se repiten para esa tabla.

- **Clave primaria**

Clave candidata que se escoge como identificador de las tuplas. Se elige como primaria la candidata que identifique mejor a cada tupla en el contexto de la base de datos. Por ejemplo una campo con el DNI sería clave candidata de una tabla de clientes, aunque si en esa relación existe un campo de código de cliente, este sería mejor candidato para clave principal, porque es mejor identificador para ese contexto.

- **Clave alternativa**

Cualquier clave candidata que no sea primaria.

- **Clave externa, ajena o foránea**

Atributo cuyos valores coinciden con una clave candidata (normalmente primaria) de otra tabla.

RESTRICCIÓN

Una restricción es una condición de obligado cumplimiento por los datos de la base de datos. Las hay de varios tipos.

- Aquellas que son definidas por el hecho de que la base de datos sea relacional:

    - **No puede haber dos tuplas iguales**
    - **El orden de las tuplas no es significativo**
    - **El orden de los atributos no es significativo**
    - **Cada atributo sólo puede tomar un valor en el dominio en el que está inscrito**

- Aquellas que son incorporadas por los usuarios:

    - **Clave primaria (*primary key*)**

    Hace que los atributos marcados como clave primaria no puedan repetir valores. Además obliga a que esos atributos no puedan estar vacíos. Si la clave primaria la forman varios atributos, ninguno de ellos podrá estar vacío.

    - **Unicidad (*unique*)**

    Impide que los valores de los atributos marcados de esa forma, puedan repetirse. Esta restricción debe indicarse en todas las claves alternativas.

    - **Obligatoriedad (*not null*)**

    Prohíbe que el atributo marcado de esta forma no tenga ningún valor (es decir impide que pueda contener el valor nulo, **null**).

    - **Integridad referencial (*foreign key*)**

    Sirve para indicar una clave externa. Cuando una clave se marca con integridad referencial, no se podrán introducir valores que no estén incluidos en los campos relacionados con esa clave.

    Esto último causa problemas en las operaciones de borrado y modificación de registros, ya que si se ejecutan esas operaciones sobre la tabla principal quedarán filas en la tabla secundaria con la clave externa sin integridad. Esto se puede manipular agregando las siguientes cláusulas:

        - **RESTRICT:** esta opción impide eliminar o modificar filas en la tabla referenciada si existen filas con el mismo valor de clave foránea.
        - **CASCADE:** borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado de clave, implica borrar las filas

con el mismo valor de clave foránea o modificar los valores de esas claves foráneas.

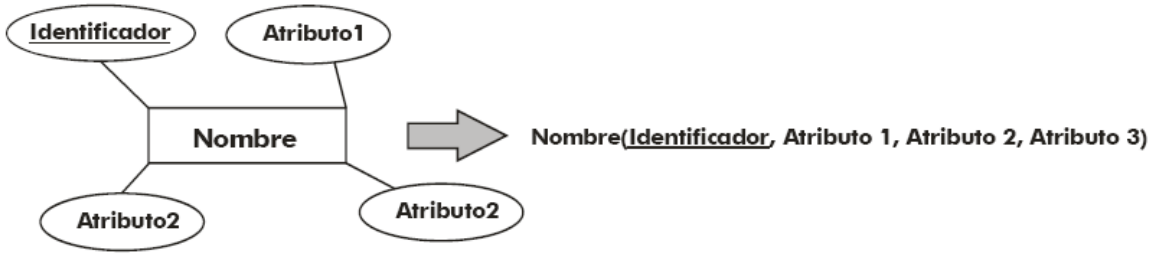
- **SET NULL:** borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado de clave, implica asignar el valor *NULL* a las claves foráneas con el mismo valor.
- **NO ACTION:** las claves foráneas no se modifican, ni se eliminan filas en la tabla que las contiene.
- **SET DEFAULT:** borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado implica asignar el valor por defecto a las claves foráneas con el mismo valor.

ESQUEMA RELACIONAL

Una relación, en el esquema relacional, se define de la siguiente forma:

<nombre_relacion> (<atributo 1>, <atributo 2>, ...)

donde el atributo clave principal aparece subrayado y donde también se señala, de alguna forma, cuáles son claves foráneas (por ejemplo, con un *).

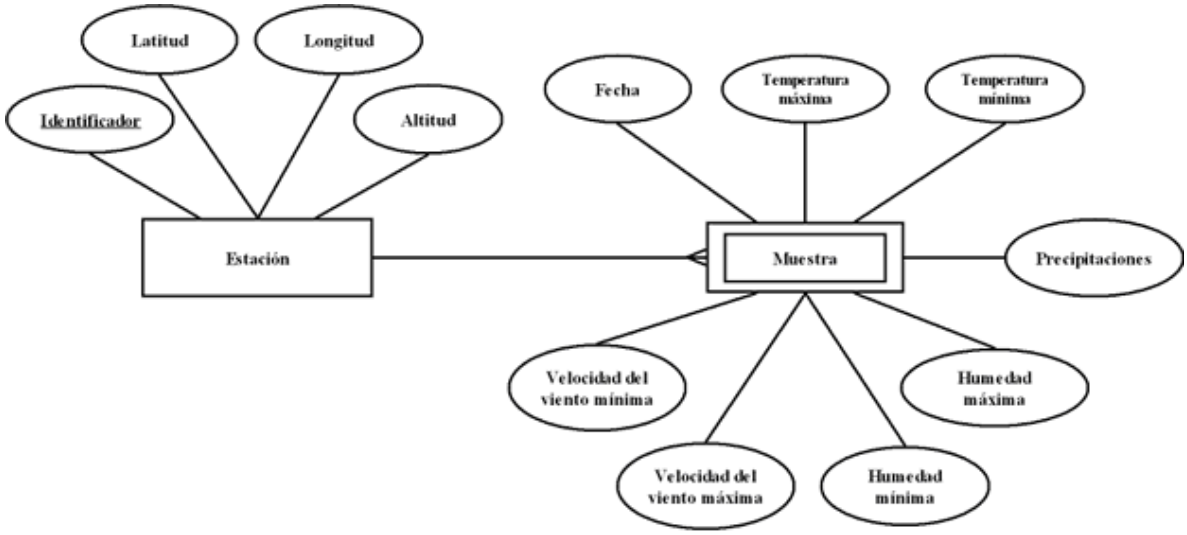


Por ejemplo, el esquema conceptual del ejercicio estaciones meteorológicas se expresa con los siguientes esquemas relacionales:

```
Estacion (Identificador, Latitutd, Longitud, Altitud )

Muestra (IdentificadorEstacion*, Fecha, TemperaturaMinima,
         TemperaturaMaxima, Precipitaciones, HumedadMinima,
         HumedadMáxima, VelocidadVientoMinima,
         VelocidadVientoMaxima)
```

que corresponden al modelo entidad-relación:



10. Normalización

La normalización es una técnica que busca dar eficiencia y fiabilidad a una BD relacional. Su objetivo es, por un lado, llevar la información a una estructura donde prime el aprovechamiento del espacio; y por otro lado, que el manejo de información pueda llevarse a cabo de forma rápida.

Cuando realizamos un diseño en el modelo relacional existen diferentes alternativas, pudiéndose obtener diferentes esquemas relacionales. No todos ellos serán equivalentes y unos representarán mejor la información que otros.

Las tablas obtenidas pueden presentar problemas:

- **Redundancia.** Se llama así a los datos que se repiten continua e innecesariamente por las tablas de las bases de datos. Cuando es excesiva es evidente que el diseño hay que revisarlo, es el primer síntoma de problemas y se detecta fácilmente.
- **Ambigüedades.** Datos que no clarifican suficientemente el registro al que representan. Los datos de cada registro podrían referirse a más de un registro o incluso puede ser imposible saber a qué ejemplar exactamente se están refiriendo.
- **Pérdida de restricciones de integridad.** Normalmente debido a *dependencias funcionales***.** Más adelante se explica este problema.

- **Anomalías en operaciones de modificación de datos.** El hecho de que al insertar un solo elemento haya que repetir tuplas en una tabla para variar unos pocos datos. O que eliminar un elemento suponga eliminar varias tuplas necesariamente (por ejemplo que eliminar un cliente suponga borrar seis o siete filas de la tabla de clientes, sería un error muy grave y por lo tanto un diseño terrible).

La normalización nos permite eliminar estos problemas, forzando a la división de una tabla en dos o más.

Para comprender bien las formas normales es necesario identificar lo que significa **dependencia funcional:**

Se dice que existe dependencia funcional entre dos atributos de una tabla si para cada valor del primer atributo existe un sólo valor del segundo.

Por ejemplo, en el esquema

```
Nota (nombre_alumno, asignatura, nota, ciudad)
```

existe dependencia funcional nombre_alumno -> ciudad, es decir para un valor de nombre de alumno existe un solo valor de ciudad.

FORMAS NORMALES

Las formas normales se corresponden a una teoría de normalización iniciada por Edgar F. Codd y continuada por otros autores (entre los que destacan Boyce y Fagin). Codd definió en 1970 la primera forma normal. Desde ese momento aparecieron la segunda, tercera, la Boyce-Codd, la cuarta y la quinta forma normal. En este documento sólo consideraremos las tres primeras formas normales.
Una tabla puede encontrarse en primera forma normal y no en segunda forma normal, pero no al contrario. Es decir los números altos de formas normales son más restrictivos.

- **Primera forma normal (1FN)**

Es una forma normal inherente al esquema relacional, por lo que su cumplimiento es obligatorio; es decir toda tabla realmente relacional la cumple. Se dice que una tabla se encuentra en primera forma normal si

impide que un atributo de una tupla pueda tomar más de un valor. La siguiente relación no cumple la primera forma normal:

TRABAJADOR

NOMBRE	**Departamento**
Elías	Informática
David	Coordinación Mantenimiento

Para resolver este problema simplemente se descomponen aquellas tuplas en los que los atributos tienen más de un valor en tantas tuplas como valores haya, cada una con un valor. La siguiente relación sí cumple la primera forma normal:

TRABAJADOR

NOMBRE	**Departamento**
Elías	Informática
David	Coordinación
David	Mantenimiento

- **Segunda forma normal (2FN)**

Ocurre si una tabla está en primera forma normal (1FN) y además cada atributo que no sea clave, depende de forma funcional completa respecto de cualquiera de las claves. Toda la clave principal debe hacer dependientes al resto de atributos, si hay atributos que dependen sólo de parte de la clave, entonces esa parte de la clave y esos atributos formarán otra tabla.

ALUMNO

DNI	**CodCurso**	**Nombre**	**Nota**
31777999	34	Elías	10
31777999	25	Elías	9
31555222	34	Luisa	8
31456712	25	David	6
31456712	34	David	7

Suponiendo que el DNI y el código de curso forman la clave principal para esta tabla, sólo la nota tiene dependencia funcional completa. El nombre

depende de forma completa del DNI. La tabla no satisface la segunda forma normal (no es 2FN). Para arreglarlo separamos la tabla en dos:

ALUMNO

DNI	**Nombre**
31777999	Elías
31555222	Luisa
31456712	David

CURSOS

DNI	**CodCurso**	**Nota**
31777999	34	10
31777999	25	9
31555222	34	8
31456712	25	6
31456712	34	7

- **Tercera forma normal (3FN)**

Ocurre cuando una tabla está en segunda forma normal (2FN) y además ningún atributo que no sea clave depende funcionalmente de forma transitiva de la clave primaria.

ALUMNO

DNI	**Nombre**	**CodProvincia**	**Provincia**
31777999	Elías	11	Cádiz
31777111	Pepe	41	Sevilla
31555222	Rosa	29	Málaga
31717171	Juana	11	Cádiz
12002003	Manuela	08	Madrid

La provincia depende funcionalmente del código de provincia, lo que hace que no esté en 3FN. El arreglo sería:

ALUMNO

DNI	**Nombre**	**CodProvincia**
31777999	Elías	11
31777111	Pepe	41
31555222	Rosa	29
31717171	Juana	11

12002003	Manuela	08

PROVINCIA

CodProvincia	**Provincia**
11	Cádiz
29	Málaga
08	Madrid
41	Sevilla

11. Paso de Entidad-Relación al modelo relacional

Previo a la aplicación de las reglas de transformación de esquemas entidad-relación a esquemas relacionales es conveniente la preparación de los esquemas entidad-relación mediante la aplicación de unas reglas que faciliten y garanticen la fiabilidad del proceso de transformación.

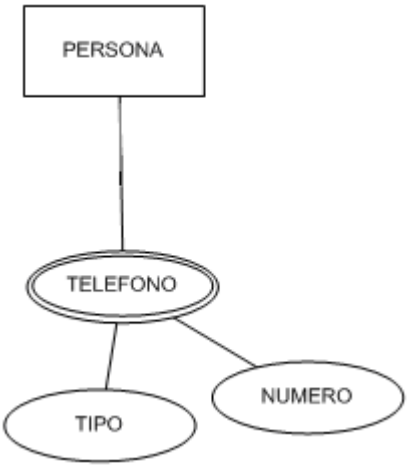
Estas reglas preparatorias se basan en la aplicación de la 1FN y su objetivo es eliminar las siguientes anomalías:

- Atributos con valores múltiples
- Atributos compuestos

ELIMINACIÓN DE ATRIBUTOS MÚLTIPLES

Todos los atributos múltiples se deben transformar en un tipo de entidad débil por existencia con una relación de muchos a muchos o de uno a muchos, según sea el caso, con el tipo de entidad sobre el cual estaba definido. Si se considera que la nueva entidad creada resulta ambigua, se le pueden añadir atributos o heredarlos de la otra entidad.

Suponemos para el siguiente ejemplo que una persona puede tener varios números de teléfono.



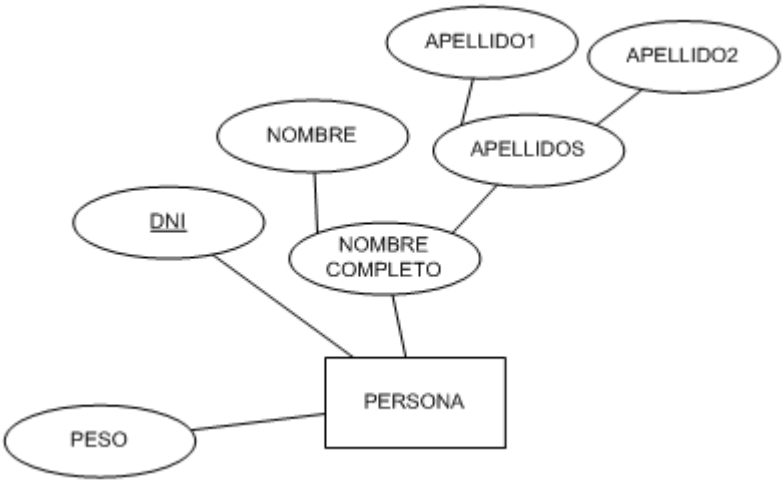
En este caso sería necesario expresar el esquema relacional de la forma:

```
PERSONA ( ... lista de atributos ...)
TELEFONO(idPersona*, numero, tipo)
```

ELIMINACIÓN DE ATRIBUTOS COMPUESTOS

Todos los atributos compuestos deben ser descompuestos en atributos simples que quedan asociados a la misma entidad.

El esquema entidad-relación:



Se expresaría mediante el siguiente esquema relacional:

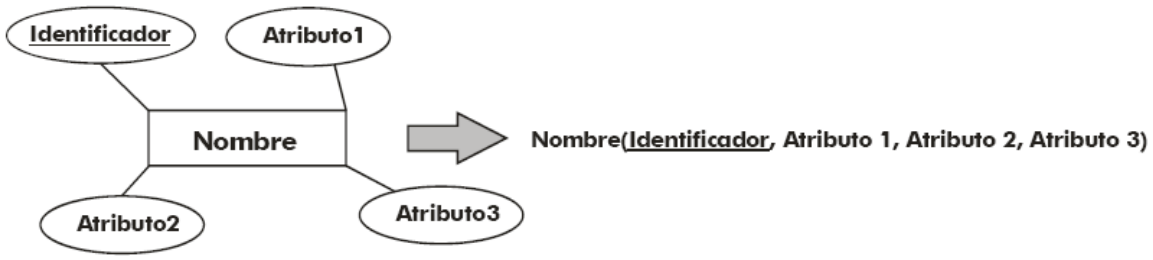
PERSONA (DNI, nombre, apellido1, apellido2, peso)

TRANSFORMACIÓN DE LAS ENTIDADES FUERTES

En principio las entidades fuertes del modelo E-R son transformadas al modelo relacional siguiendo estas instrucciones:

- **Entidades.** Las entidades pasan a ser **tablas**.
- **Atributos**. Los atributos pasan a ser **columnas**.
- **Identificadores principales**. Pasan a ser **claves primarias**.
- **Identificadores candidatos**. Pasan a ser claves candidatas.

Esto hace que la transformación se produzca según este ejemplo:

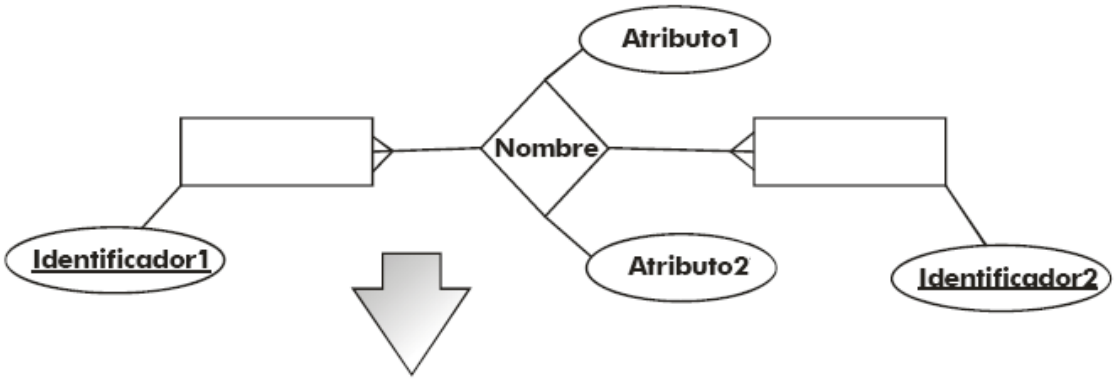


TRANSFORMACIÓN DE RELACIONES

La idea inicial es transformar cada relación en una tabla, pero hay que distinguir según el tipo de relación.

- **Relaciones varios a varios**

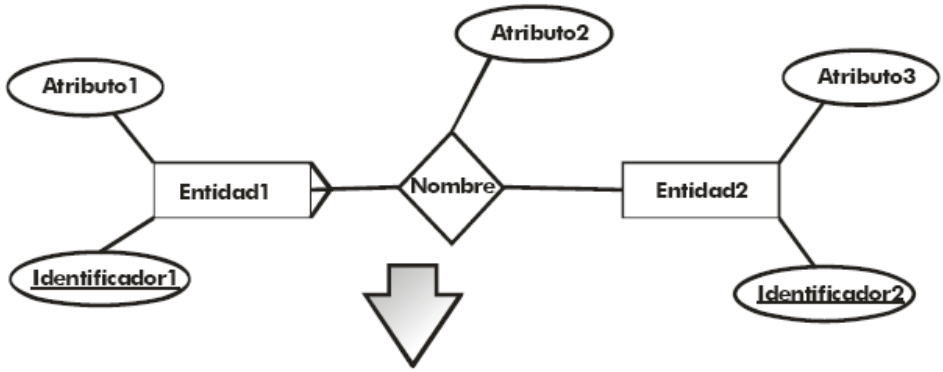
En las relaciones varios a varios la relación se transforma en una tabla cuyos atributos son: los atributos de la relación y las claves de las entidades relacionadas (que pasarán a ser claves externas). La clave de la tabla la forman todas las claves externas.



Nombre(Identificador1,Identificador2,Atributo1,Atributo2)

- **Relaciones uno a varios o uno a uno**

Las relaciones de tipo uno a varios no requieren ser transformadas en una tabla en el modelo relacional. En su lugar la tabla del lado *varios* (**tabla relacionada**) incluye como clave externa el identificador de la entidad del lado *uno* (**tabla principal**). En el caso de que el número mínimo de la relación sea de *cero* (puede haber ejemplares de la entidad uno sin relacionar), se deberá permitir valores nulos en la clave externa *identificador2*. En otro caso no se podrán permitir (ya que siempre habrá un valor relacionado).



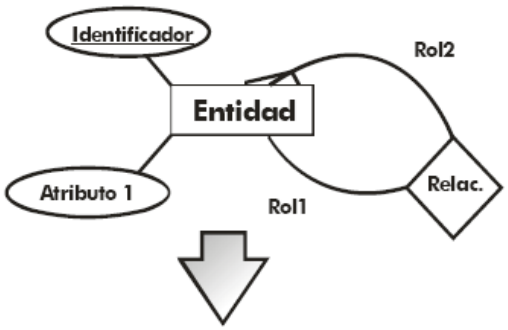
Entidad2(Identificador2,Atributo3)
Entidad1(Identificador1,Atributo1,Identificador2,Atributo2)

En el caso de las relaciones uno a uno, ocurre lo mismo: la relación no se convierte en tabla, sino que se coloca en una de las tablas (en principio daría igual cuál) el identificador de la entidad relacionada como clave externa. En el caso de que una entidad participe opcionalmente en la

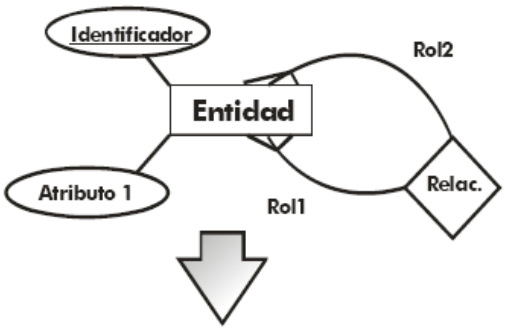
relación, entonces es el identificador de ésta el que se colocará como clave externa en la tabla que representa a la otra entidad.

- **Relaciones reflexivas**

Las relaciones reflexivas o recursivas se tratan de la misma forma que las otras, sólo que un mismo atributo puede figurar dos veces en una tabla como resultado de la transformación.



Entidad(Identificador,Atributo1,Identificador Rol 1)

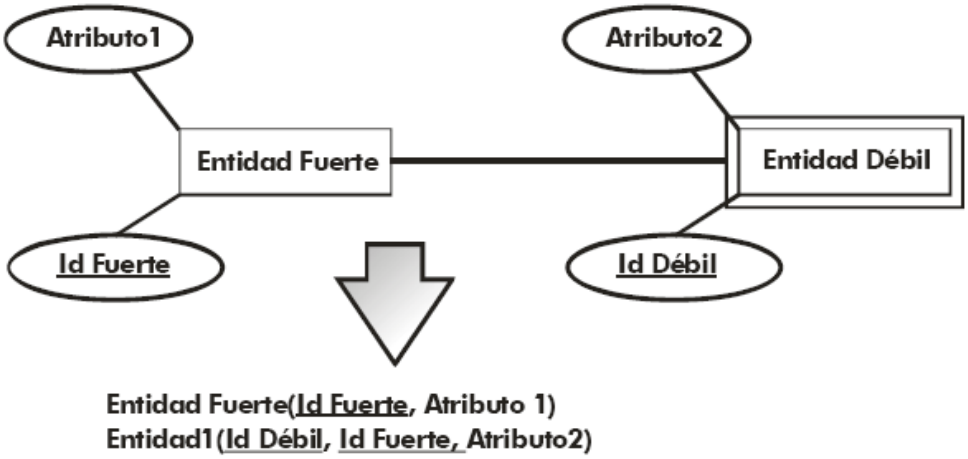


Entidad(Identificador,Atributo1)
Relac(Identificador Rol 1, Identificador Rol 2,Atributo1)

TRANSFORMACIÓN DE LAS ENTIDADES DÉBILES

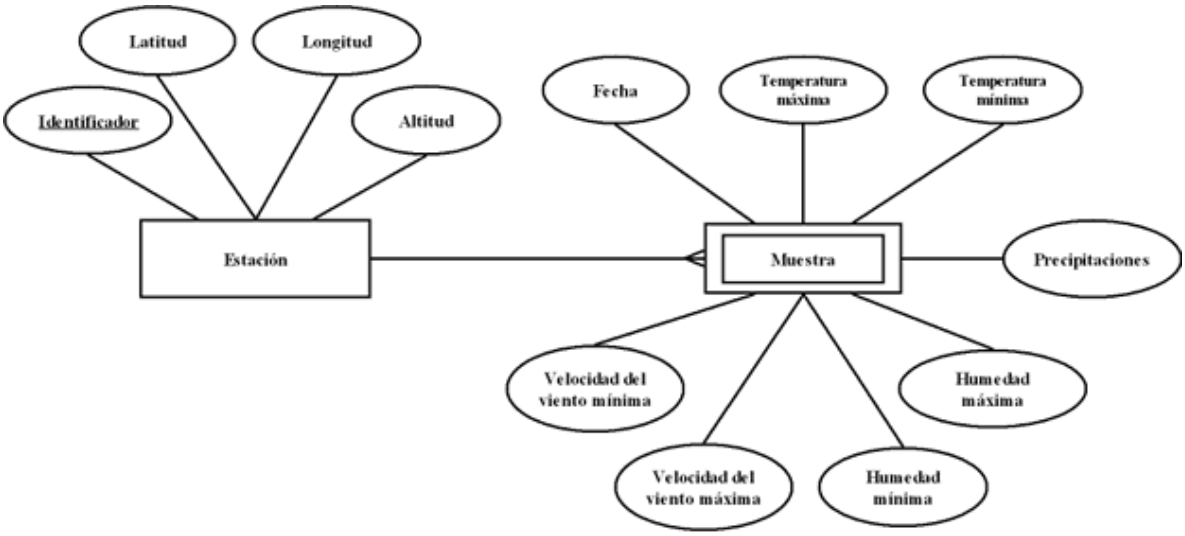
Toda entidad débil incorpora una relación implícita con una entidad fuerte. Esta relación no necesita incorporarse como tabla en el modelo relacional. Sí se necesita incorporar la clave de la entidad fuerte como clave externa en la entidad débil. Es más, normalmente esa clave externa forma parte de la clave principal de la tabla que representa a la entidad débil.

En ocasiones el identificador de la entidad débil es suficiente para identificar los ejemplares de dicha entidad, entonces ese identificador quedaría como clave principal, pero el identificador de la entidad fuerte seguiría figurando como clave externa en la entidad débil.



12. Ejemplo (Modelo relacional)

En el ejemplo anterior diseñamos el esquema entidad-relación sobre las muestras recogidas en varias estaciones meteorológicas.



El esquema relacional sería el siguiente:

estacion (id, latitutd, longitud, altitud)

muestra (estacion id*, fecha, temperaturaminima, temperaturamaxima, precipitaciones, humedadminima, humedadmáxima, velocidadminima, velocidadmaxima)

- Representación gráfica (DBDesigner)