Bases de Datos

# Diagrama Básico

## Entidad Relación de Peter Chen

Una entidad se representa con un rectángulo con su nombre en el interior:

https://miro.medium.com/v2/resize:fit:83/1*5pakHuJEHjzZA8SIT9QVKA.png

Ilustración : Representación de una entidad

Una relación se representa con un rombo, con el nombre del rol o verbo en su interior, indicando también el tipo de relación debajo. Veamos cómo es:

https://miro.medium.com/v2/resize:fit:483/1*RpLxqpEGltoK2W99o0gUjA.png

Ilustración : Representación de una relación

Así se representarían los grados distintos de las relaciones:



Ilustración : Relaciones Binarias y Terciarias respectivamente

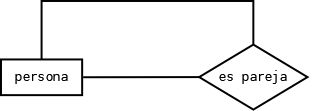


Ilustración : Relación reflexiva

Veamos cómo se representaría los tipos de relaciones que existen.

Un relación 1:1 se representa de la siguiente forma:

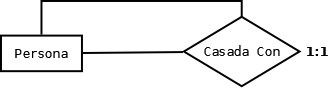


Ilustración : Representación de una relación 1:1

En este ejemplo hacemos una interrelación 1:1 con persona “es casada” con persona. Es una relación 1:1 porque una persona sólo puede estar casada con otra persona y sólo una. Normalmente este tipo de relación es un campo más de la entidad persona.

Los campos se representan mediante un óvalo y el nombre dentro. Las claves principales se representan con un trazado continuo y las claves alternativas con trazados discontinuos. Veamos un ejemplo:

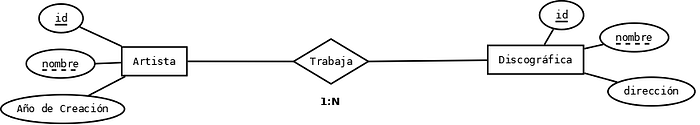


Ilustración : Representación campos

En este ejemplo vamos a suponer que no hay nombres de artistas y discográficas iguales, lo cual la usamos como clave alternativa.

## ****¿Cómo representamos las reflexivas?****

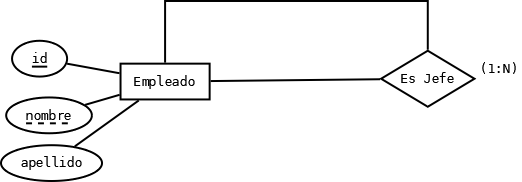
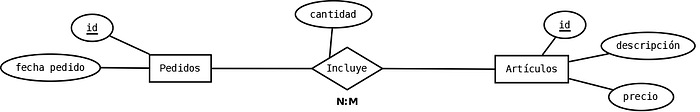


Ilustración : Representación de una relación reflexiva

En este ejemplo, vamos a interpretar que un empleado puede ser o no jefe pero como mínimo es un empleado y en el caso de que sea jefe, puedes serlo de 1 o más empleados, de ahí es donde sacamos que es un 1:N.

## ****¿Cómo representamos las N:M?****



Interpretamos esta relación como un pedido “incluye” varios artículos. Y un artículo se “incluye” en varios pedidos.

En este ejemplo se puede observar que tenemos un campo en el propio rol de la relación, es decir, podemos tener campos que no están incluidos en las propias entidades sino que nacen de la propia relación que hacemos.

## Cardinalidad

Con esto se pretende controlar el número de registros mínimos y máximos que puede haber en la relación de la entidades que intervienen. Se representan poniendo la etiqueta (0,1), (1,1), (0,n),(1,n). Veamos cómo funciona:

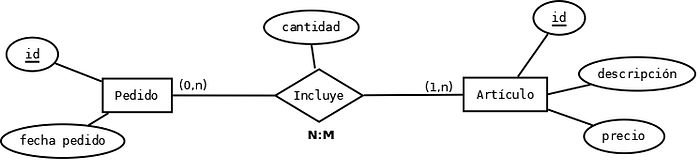


Ilustración : Representación de cardinalidades

Este se leería de la siguiente forma:

* Para la cardinalidad (1,n). Hablamos de pedidos. Un pedido “incluye” mínimo un artículo y como máximo varios. Si observamos el diagrama si hablamos de pedidos debemos poner la cardinalidad de ésta en la entidad contraria con la que se relaciona.
* Para la cardinalidad (0,n). Hablamos de artículos. Un artículo se “incluye” como mínimo en 0 pedidos y como máximo en varios. ¿Qué es eso de que en 0 pedidos? Es posible que tengamos artículos en los que no se hayan hecho en ningún pedido, por eso mismo se pone cardinalidad 0 y usamos la cardinalidad n para especificar que un artículo, un bolígrafo por poner un ejemplo, se puede “incluir” en varios pedidos distintos.

Podemos omitir el tipo de relación en el rombo, ya que las propias cardinalidades máximas nos lo especifican.

## ****¿Cómo sería en una reflexiva?****

Vamos a usar el mismo ejemplo de antes, el de los jefes y los empleados:

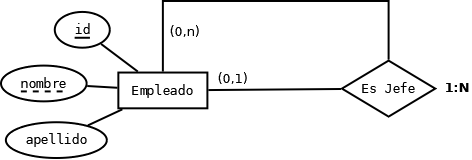


Ilustración : Representación de una reflexiva con cardinalidades

* Lado (0,1). Un empleado mínimo puede tener 0 jefe, porque el jefe es él o 1 en caso de que no sea jefe y sea un subordinado.
* Lado (0,n). Un empleado puede tener 0 o muchos subordinados, es decir, si es jefe tendrá empleados a su cargo pero en el caso de que no lo sea, será un subordinado.

# Diagrama Avanzado

## Entidad Relación Extendido

Este método es un poco más avanzado ya que existen otras muchas cosas a tener en cuenta a la hora de hacer esquemas.

## ****Entidades Fuertes y Débiles****

Hasta ahora sólo habíamos visto una forma de representar las entidades pero en realidad, existen formas de hacerlo dependiendo de qué tipo sea:

**Entidades Fuertes o regulares.** Son entidades cuyo registros o tuplas no necesitan de la existencia de otras entidades, tienen existencia propia. Este tipo se representa como lo hemos estado haciendo hasta ahora.

https://miro.medium.com/v2/resize:fit:83/1*5pakHuJEHjzZA8SIT9QVKA.png

Ilustración : Representación de una entidad

**Entidades Débiles.** Son entidades cuyo registros necesitan la existencia en otras entidades regulares, de tal forma que si se borra el registro de la entidad fuerte se borrarán todas las tuplas de las que depende.

Un ejemplo de esto sería un libro y las diferentes copias que puede tener una biblioteca. Las diferentes copias que tenemos de un libro dependen totalmente de que el libro exista:



Ilustración : Representación de relaciones con entidades débiles

* Lado (1,n). Un libro puede tener mínimo un ejemplar y como máximo varios, para poder prestarlos en la biblioteca.
* Lado (1,1). Una copia concreta, sólo puede pertenecer como mínimo a un libro y como máximo a uno, no podemos tener una copia de dos libros distintos a la vez.
* Las entidades débiles se representan con un doble recuadro a la entidad.

Ahora bien basándonos en entidades débiles y fuertes existen básicamente dos tipos de relaciones:

* **Relaciones regulares.** Son relaciones entre dos entidades regulares o fuertes.
* **Relaciones débiles.** Son relaciones entre una entidades débil y otra fuerte de la que depende.

En las relaciones débiles existen dos tipos de dependencias:

**Dependencia por identificación.** Esto sucede cuando la entidad débil no es capaz de identificar un registro con sus propios atributos y necesita complementarlo con la clave principal de la entidades fuerte. Veamos el ejemplo de libro y ejemplar:

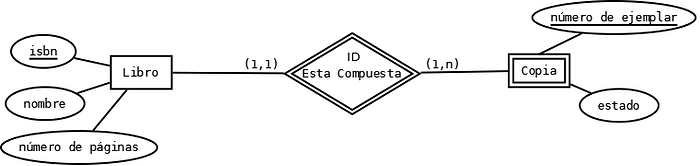


Ilustración : Relación con dependencia por identificación

Las dependencias por identificación se representan poniendo el rombo de la relación con doble subrayado y colocando ID dentro.

## ****Bueno y ¿por qué es débil por identificación?****

* Vamos a suponer que en la entidad copia se almacenan progresivamente el número de ejemplar, es decir, si tienes tres copias habrá 3 tuplas diferentes con número de ejemplar 1, 2, 3….
* Si observamos bien, por sí solas, las tuplas de la entidad copia no se identifican por sí solas con sus atributos ya que pueden haber libros que también tengan 3 copias e incluso más y las tuplas se repiten.
* Para solucionar esto debemos usar la clave principal ISBN en la entidad copia y usarlo con el número de ejemplar, de esta forma identificamos cada copia distinta que hay de cada libro. A esto es lo que se le llama débil por identificación.

**Dependencia por existencia.** Cuando en una relación débil no depende por identificación, siempre será débil por existencia, ya que no puede existir la débil si no existe la fuerte. Vamos a ver un ejemplo:

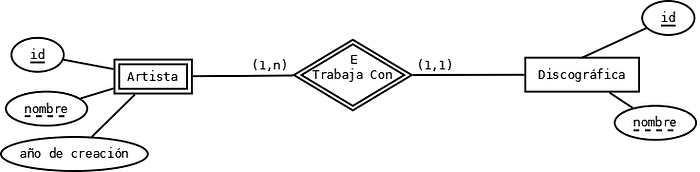


Ilustración : Relación con dependencia por existencia

* En este ejemplo no hay dependencia por identificación. Las dos entidades se identifican sus tuplas por sí solas con sus propios atributos.
* Al ser la cardinalidad mínima a 1 en las dos entidades, obliga a que no pueden existir artistas que no tenga una discográfica con la que trabajar para poder publicar su álbumes.
* Esto conlleva que, por la relación, hace débil por existencia a los artistas, ya que se le obliga a que exista una discográfica para poder subsistir.

# Jerarquías

## Relaciones de Herencia

Conocidas también por relaciones ISA o de herencia. Ocurre cuando se necesita unificar entidades agrupándolas en una principal (generalización) o dividir una general en otras más específicas (especialización).

**¿Y cómo se sabe si es o no una jerarquía?** En ocasiones podemos ver entidades que son muy parecidas y se diferencian en pocos atributos, en estos casos se debería hacer una jerarquía con las entidades que intervienen. Veamos un ejemplo:

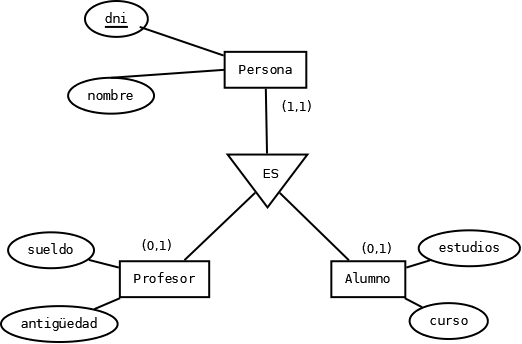


Ilustración : Representación de jerarquías

* Las jerarquías se representan con un triángulo invertido.
* La entidad principal persona se llama súper-tipo o súper-entidad.
* Los diferentes ramales profesor y alumno se llama sub-tipo o sub-entidad.
* Unas de características principales de las jerarquías es la herencia de sus propiedades, es decir, el súper-tipo tiene atributos que heredarán los subtipos de forma que DNI y Nombre también pertenecen a las entidades Profesor y Alumno.

Existen diferentes tipos de jerarquías.

* **Generalización.** En toda jerarquía los campos se heredan del súper-tipo, pero es posible que cada entidad tenga su propia clave principal, de forma que se puede identificar de forma independiente. En estos casos la cardinalidad mínima será siempre en todas las entidades (0,1). Podemos identificar una generalización cuando los súper-tipos y subtipos tienen sus propias claves.

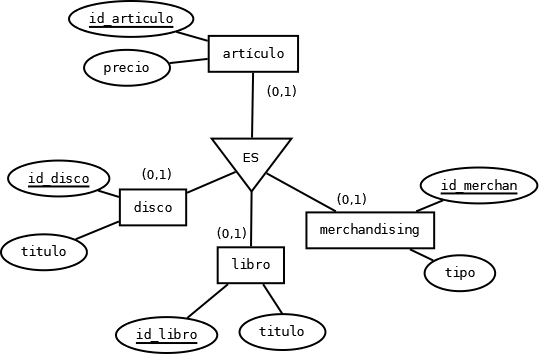


Ilustración : Jerarquía de generalización

* **Especialización.** En este tipo de jerarquía también se heredan los campos del súper-tipo, pero la diferencia que hay con la generalización, es que en la especialización todas las entidades intervinientes comparten la clave del súper-tipo. En estos casos la cardinalidad del súper-tipo será (1,1).

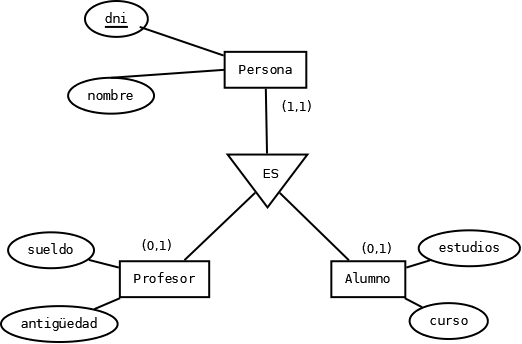


Ilustración : Jerarquía de especialización

## Tipos de relaciones en las jerarquías

**Relaciones de jerarquía total.** Ocurre cuando no existen registros del súper-tipo que no estén relacionados con el subtipo, es decir, usando el ejemplo anterior no pueden existir personas que no sean ni profesor ni alumnos, siempre tiene que estar relacionados con alguno de ellos. Se representa con la cardinalidad mínima a uno en la súper-entidad y con un circulo en la línea donde une el súper-tipo con la relación.

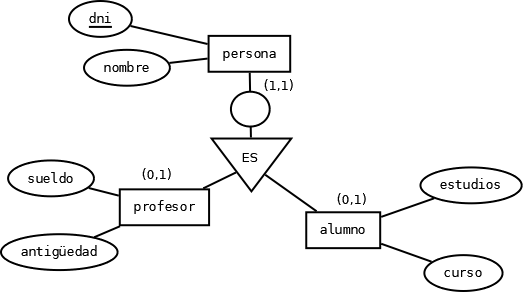


Ilustración : Jerarquía relación total

**Relaciones de jerarquía parcial.** Esto pasa cuando registros del súper-tipo no están relacionados con ninguno de los subtipos. Un ejemplo:

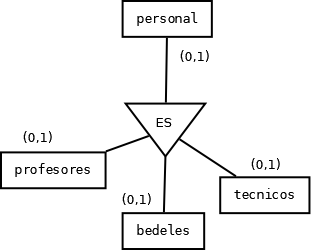
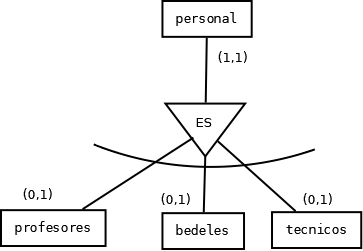


Ilustración : Jerarquía relación parcial

A la hora de interpretar esto lo que quiere decir es que puede haber personal que no sea ni profesor, ni bedel, ni técnico. Se representan con cardinalidad mínima a 0 en la súper-entidad.

**Relaciones de jerarquía exclusiva.** Ocurre cuando un registro de la súper-entidad está relacionado de forma exclusiva con las sub-entidades, es decir, un personal sólo puede ser profesor, bedel o técnico, no puede ser dos cosas a la vez, solamente una. Se representan con arco por debajo del triángulo invertido.



**Relaciones de jerarquía solapada.** Esto pasa cuando un registro del súper-tipo puede estar relacionado con más de un registro de los subtipos, es decir, un personal puede ser profesor y técnico a la vez. Como es normal esto pasa en relaciones que no son exclusivas y son totales, ya que obligamos a que esté relacionada con alguna de ellas.

Como resumen, vamos a ver qué tipos de esquemas nos podemos encontrar con las jerarquías:

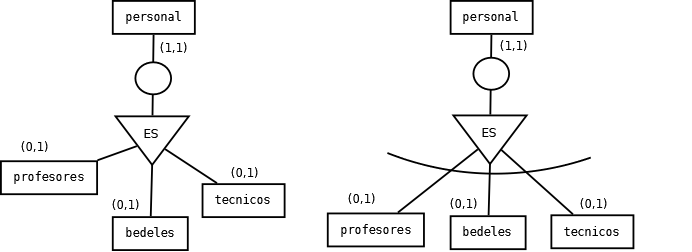


Ilustración : Total Solapada y Total Exclusiva respectivamente

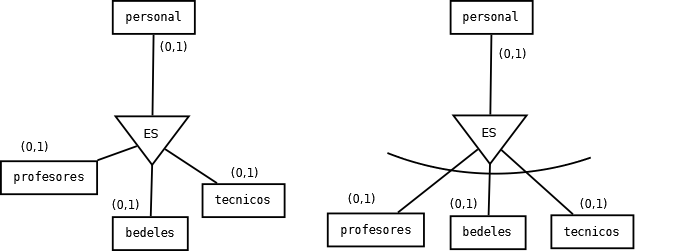


Ilustración : Parcial Solapada y Parcial Exclusiva respectivamente

* Como es natural si es parcial no puede ser total, si es solapada no puede ser exclusiva.
* Otra cosa que podemos deducir es que la jerarquía total sólo pasa con las especializaciones y la parcial con las generalizaciones.

# Modelo Relacional

## Paso del Modelo E-R al Relacional

El modelo relacional es un paso previo para la conversión, en la práctica, de las tablas y atributos que pueda haber en una base de datos.

En esta parte lo que se a intentar lograr es pasar el esquema que hicimos en pasos anteriores a un esquema de tablas previo para luego reflejarlo en la práctica a través de SQL. Empecemos:

## ****Transformación de Entidades****

* Las entidades pasan a ser tablas directamente.
* Los atributos pasan ser campos o columnas de la tabla.
* Los identificadores principales pasan a ser claves primarias.
* Los identificadores candidatos pasan ser claves candidatas o alternativas.

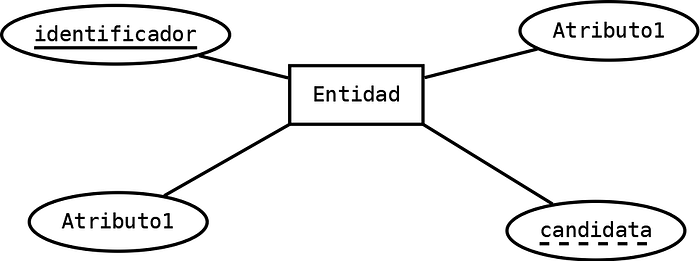


Ilustración : Entidad fuerte

**Nota:** Como no tengo forma de poder subrayar el identificador y las claves foráneas en medium voy a cambiar un poco la leyenda.

* Negrita claves principales.
* Cursivas clave foráneas o externas.
* En las candidatas usaré su propia palabra.

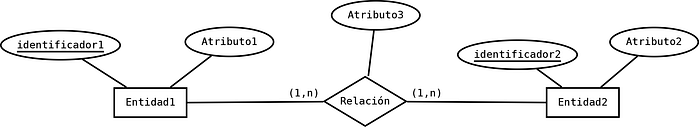
Quedaría de la siguiente forma:

Entidad: **Identificador**, Atributo1, Atributo2, Candidata.

## ****Relaciones Varios a Varios****

Relaciones de entidades cuya cardinalidad máxima será n en las dos entidades.

* Se creará una tabla por cada entidad con sus campos, atributos, identificadores y claves alternativas si las hubiere.
* En todas las relaciones N:M se genera otra tabla para relación en la que se compondrá de las claves principales de las entidades relacionadas y los atributos de la relación que pudiera haber.
* En la tabla de relación la clave principal serán los dos campos identificadores y al mismo tiempo serán de forma independiente clave foránea de cada una de las entidades intervinientes.

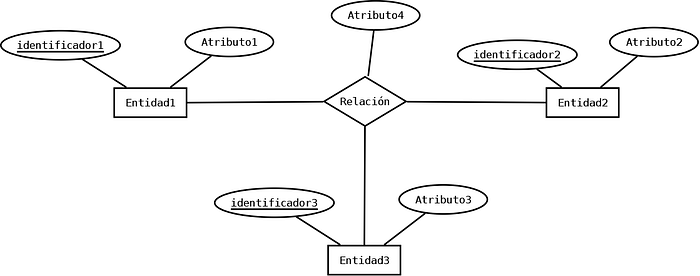


Relación N:M.

Entidad: **Identificador1**, Atributo1.  
Entidad2: **Identificador2**,Atributo2.  
Relación: **Identificador1**, **Identificador2**, Atributo3.

# Relaciones de Orden n.

Las relaciones ternarias, cuaternarias…etc, que unen más de dos relaciones se tratarán como las N:M, es decir, se crea una tabla por entidad con todos sus atributos, identificadores y candidatas que hubiere. Y para finalizar se generará otra tabla de relación con todos los identificadores de las entidades intervinientes y los posibles atributos que pudiera haber en la relación.

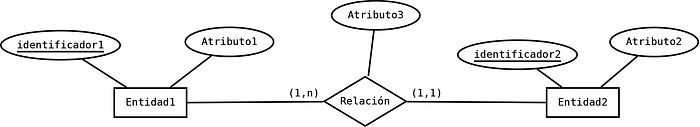


Relaciones de Orden N.

Entidad1: **Identificador1**, Atributo1.  
Entidad2: **Identificador2**, Atributo2.  
Entidad3: **Identificador3**, Atributo3.  
Relación: **Identificador1**, **Identificador2**, **Identificador3**, Atributo4.

# Relaciones 1:N.

En el tipo de relación binaria 1:N, no es necesario generar otra tabla, lo que se hace incluir en la entidad del lado de N el identificador de la otra tabla y los posibles campos de la relación si los hubiere. Veamos como funciona:



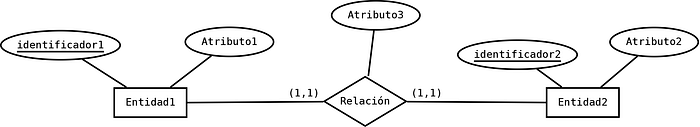
Relación 1:N.

Entidad1: **Identificador1**, Atributo1 Identificador2, Atributo3.  
Entidad2. **Identificador2**, Atributo2.

# Relaciones 1:1.

**Relaciones donde todas las cardinalidades sean 1 en todas sus entidades. Hay dos formas de proceder:**

* Se crea una tabla por cada entidad con todos sus atributos y se coloca la clave principal de una de ellas en la otra entidad como clave externa, da igual en cual y los posibles campos de la relación si los hubiere, teniendo en cuenta que pasará a ser clave foránea y alternativa.
* Se crea una única tabla con todos los campos de las entidades intervinientes, incluyendo los atributos de la relación si los hubiere y además una de las dos claves primarias, da igual cual, será la clave principal y la otra alternativa.



Relaciones 1:1.

Primera Solución.

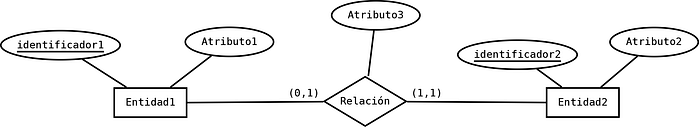
Entidad1: **Identificador1**, Atributo1, Identificador2, Atributo3.  
Entidad2: **Identificador2**, Atributo2.

Segunda Solución.

Entidad1: **Identificador1**, Atributo1, Identificador2(candidata), Atributo2, Atributo3.

**Relaciones 1:1 cuya cardinalidad máxima sera 1 en las dos entidades pero una de ellas la cardinalidad mínima es 0.**

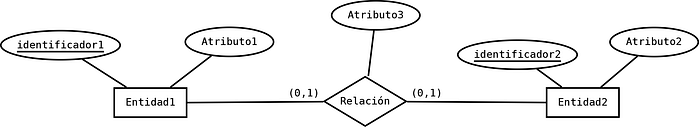
En este caso no conviene generar una única tabla, ya que habría muchos valores nulos en la tabla. Lo que se haría es generar una tabla por cada entidad y en la que la cardinalidad es 0, se coloca la clave principal de la otra tabla.



Relación 1:1

Entidad1: **Identificador1**, Atributo1, Identificador2(candidata), Atributo3.  
Entidad2: **Identificador2**, Atributo2

En el que caso de que nos encontremos con cardinalidades (0,1) en ambas entidades, la solución es casi la misma, la diferencia está en que la clave alternativa no estará en la Entidad1, pasaría a la Entidad2 pero no sería clave alternativa pero si tendría una restricción de unicidad (Unique).

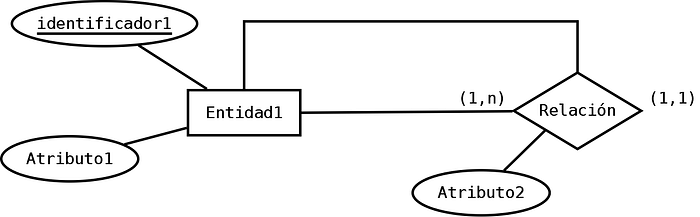


Relación 1:1.

Entidad1: **Identificador1**, Atributo1, Atributo3.  
Entidad2: **Identificador2**, Atributo2, Identificador2(unicidad)

# Relaciones Recursivas o Reflexivas.

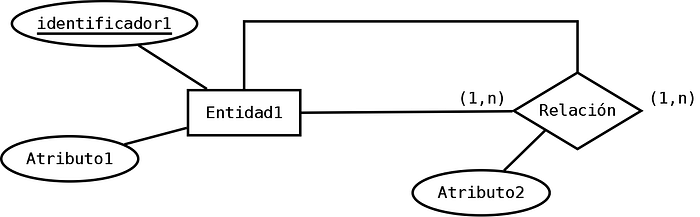
En este tipo de relaciones se trata como una relación normal, la diferencia está es que es posible que al pasarlo al Modelo Relacional se repita dos veces un mismo campo para relacionarlo con sí mismo.



Relación Reflexiva.

* Se tratará como una relación 1:N. Se generará una única tabla con todos los atributos que pueda tener, incluyendo los posibles campos de relación si los hubiere, haciendo repetición de la clave primaria como clave foránea para relacionarla consigo mismo.

Entidad1: **Identificador1**, Atributo1, Identificador1, Atributo2.



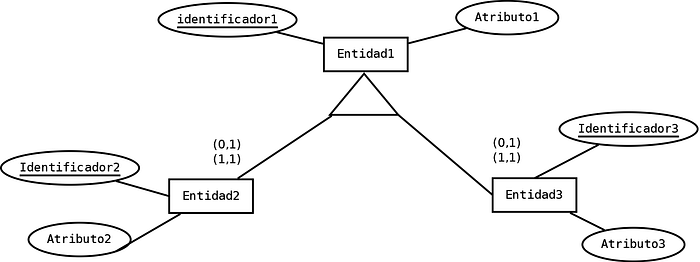
* Se tratará como una N:M. Se generará dos tablas, una de la propia entidad y sus atributos y otra de la relación en los que incluirá las dos claves externas y los atributos de la relación si hubiere.

Entidad1: **Identificador1**, Atributo1.  
Relación: **Identificador1**, **Identificador1**, Atributo2.

# Relaciones de Herencia, Jerárquicas o ISA.

Para transformar al modelo relacional las relaciones ISA se hace teniendo en cuenta la siguientes normas:

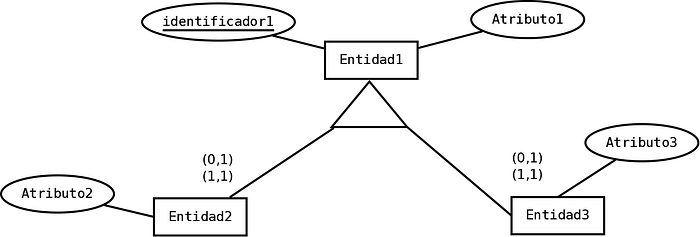
* Se generarán tablas tanto para la superentidad como las subentidades, con sus correspondientes campos o atributos.
* En el caso de que las subtipos no hereden la clave principal del supertipo, se colocará el identificador de la superentidad como clave secuendaria o alternativa.



Relaciones ISA.

Entidad1: **Identificador1**, Atributo1, Identificador1.  
Entidad2: **Identificador2**, Atributo2, Identificador1.  
Entidad3: **Identificador3**, Atributo3, Identificador1.

* En el caso de que hereden la clave principal del supertipo.



Relación ISA.

Entidad1: **Identificador1**, Atributo1.  
Entidad2: **Identificador1**, Atributo2.  
Entidad3: **Identificador1**, Atributo3.

# Creación de usuarios y permisos.

Vamos a hablar ahora de algo que yo creo que es muy importante a la hora de gestionar bases de datos; los usuarios y los permisos. Mysql al ser un administrador y servidor de base de datos, necesitamos tener usuarios y darle permisos a esos usuarios para controlar que puedan hacer, ya que no es muy seguro usar siempre el usuario root para todo, ya que si la cagamos podemos liarla bien.

## Crear un usuario.

CREATE USER ‘newuser’@’localhost’ IDENTIFIED BY ‘password’;

* En este ejemplo con el comando Create se crea el usuario.
* Newuser. Nombre del usuario nuevo.
* Localhost. Aquí lo estamos haciendo de forma local, pero en la mayoría de los casos a ser un servidor de base de datos debemos poner la ip del servidor.
* Identified by. Aquí introducimos la clave del nuevo usuario.

Ahora mismo el usuario que se ha creado no vale para nada, debemos darle permisos para que pueda hacer algo en la base de datos ya que no tiene otorgado nada, está ahí sólo sin poder moverse.

GRANT ALL PRIVILEGES ON \* . \* TO 'newuser'@'localhost';  
FLUSH PRIVILEGES;

* Grant all privileges on \*.\*. Este comando especifica los permisos que le estamos dando al usuario. Los asteriscos significa “base de datos.tabla”.
* To. Especifica el usuario al que le estamos dando permisos.
* Flush privileges. Esto sirve para refrescar los permisos en el servidor, es decir, una especie de actualizar todo para que surjan efectos los cambios que hemos realizado.

En este ejemplo le estamos dando todos los permisos de todas las bases de datos y tablas que haya en el servidor. Esto es una locura porque prácticamente estamos dándole permisos root a este usuario.

Estos son los permisos con los que podemos jugar:

* ALL PRIVILEGES: Permite dar todos los permisos habidos y por haber al usuario especificado.
* CREATE: Permite crear nuevas tablas o bases de datos.
* DROP: Permite eliminar tablas o bases de datos.
* DELETE: Permite eliminar filas de las tablas.
* INSERT: Permite insertar filas en las tablas.
* SELECT: Les permite usar el comando **SELECT** para realizar consultas.
* UPDATE: Permite actualizar las filas de las tablas.
* GRANT OPTION: Permite otorgar o eliminar privilegios de otros usuarios.

¿Entonces como lo hago de forma correcta?

GRANT tipo de permiso ON base de datos.tabla TO 'username'@'localhost';  
FLUSH PRIVILEGES;

Normalmente no damos un permiso sólo, para especificar varios permisos sólo hay que separarlos mediante comas.

GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP ON base de datos.tabla TO 'username'@'localhost';  
FLUSH PRIVILEGES;

* Siempre debemos especificar lo que realmente queremos que ese usuario pueda o no acceder y que puede hacer.

¿Y si quiero quitar permisos? Pues se hace exactamente igual, salvo que cambia el comando a REVOKE:

REVOKE tipo de permiso ON base de datos.tabla FROM 'username'@'localhost';

Podemos revocar varios permisos a la vez, sólo hay que separarlos mediante comas.

REVOKE SELECT, INSERT, UPDATE, DELETE, CREATE, DROP ON base de datos.tabla FROM 'username'@'localhost';

Otras cosas que podemos hacer con los usuarios:

SHOW GRANTS FOR 'username'@'localhost'; Muestra los permisos del usuario.DROP USER 'username'@'localhost'; Borra un usuario.  
SELECT User FROM mysql.user; Listado de todos los usuarios.

ahora si queremos probar el nuevo usuario lo que debemos hace es salir del servidor y volver a iniciar sesión en MySql.

quit \* Salgo del servidor.mysql -u usuario -p contraseña \* Vuelvo a iniciar sesión con el nuevo usuario