El Modelo Relacional

El modelo relacional fue propuesto por Frank Codd en los laboratorios de IBM.

Permite representar la información del mundo real de manera intuitiva.

Se trata de un modelo lógico que establece una estructura sobre los datos, independientemente del modo en que luego los almacenemos.

El objetivo principal del modelo relacional es proteger al usuario de la obligación de conocer las estructuras de datos físicas con las que se representa la información de una base de datos. Permite que la base de datos se pueda implementar en cualquier gestor de base de datos relacional (Oracle, MySQL, DB2,…).

La relación es el elemento fundamental del modelo. Los usuarios ven la base de datos como una colección de relaciones. Estas relaciones se pueden operar mediante el Álgebra Relacional.

Al estar fundamentado en una fuerte base matemática se puede demostrar la eficacia del modelo a la hora de operar conjuntos de datos.

El nombre de modelo relacional viene de la estrecha relación entre el elemento básico de este modelo y el concepto matemático de relación. Si tenemos dos conjuntos A y B, una relación entre estos dos conjuntos sería un subconjunto del producto cartesiano AxB.

El producto cartesiano nos dará la relación de todos los elementos de un conjunto con todos los elementos de los otros conjuntos de ese producto. Al estar trabajando con conjuntos, no puede haber elementos repetidos.

1. Las relaciones en el modelo relacional.

A estas relaciones se las llama más habitualmente “**TABLAS**”.

Se define como un conjunto de atributos, cada uno de los cuales pertenece a un dominio y que posee un nombre que identifica la “relación”.

Las “relaciones” constan de:

- **Atributos**. Referido a cada propiedad de los datos que se almacenan en la relación. Por tanto son las características que describen a una relación. (nombre, dni,...).
- **Tuplas.** Referido a cada elemento de la relación. Por ejemplo si una relación almacena datos de personas, una tupla representaría a una persona en concreto.

Puesto que una relación se representa como una tabla; podemos entender que las columnas de la tabla son los atributos; y las filas, las tuplas.

Vehículo	Dueño	TeléfonoDueño	Matrícula	Cuota
Seat Ibiza TDI 1.9	Francisco	925884721	9918-FTV	92,24
Volskwagen Polo TDI 1.0	Pedro	918773621	4231-FHD	61,98
Renault Laguna Coupé	María	929883762	7416-GSJ	145,32
Fiat Punto 1.0	Ernesto	646553421	9287-BHF	45,77

2. Elementos del modelo relacional

2.1 Tupla

Como ya hemos dicho es cada una de las filas de la relación o tabla. Se corresponde con la idea clásica de **registro**.

Representa por tanto cada elemento individual de esa relación.

Tiene que cumplir que:

- Cada tupla se debe corresponder con un elemento del mundo real.
- No puede haber dos tuplas iguales (con todos los valores iguales).

2.2 Dominio

Un dominio es el conjunto de valores que puede tomar un determinado atributo es decir que están permitidos..

Dos atributos distintos pueden tener el mismo dominio.

Ejemplo : Cadena de caracteres, números enteros, los valores Sí o No…

La forma de indicar el contenido de un dominio se puede hacer utilizando dos posibles técnicas:

- **Intensión.** Se indica cómo se obtienen los valores. Una característica general que deben cumplir los valores.

  Ejemplo : Las edades legales para trabajar: números enteros entre el 16 y el 65

- **Extensión**. Se indican específicamente los valores.

  Ejemplo : El dominio localidad se podría definir por extensión así: Palencia, Valladolid, Villamuriel de Cerrato,...

2.3 Grado de una relación.

Indica el tamaño de una relación en base al número de columnas (atributos) de la misma. Lógicamente cuanto mayor es el grado de una relación, mayor es su complejidad al manejarla.

2.4 Cardinalidad.

Número de tuplas de una relación, o número de filas de una tabla.

3. Definición formal de relación

Una relación está formada por estos elementos:

1.- Nombre. Identifica la relación.

R

2.- Cabecera de relación. Conjunto de todos los pares atributo-domino de la relación:

$$\{(A_i : D_i)\}_{i=1}^{n}$$

*Donde **n** es el **grado de la relación**.*

3.*- Cuerpo de la relación.** Representa el conjunto de ***m tuplas {t1, t2,... tn} que forman la relación. Cada tupla es un conjunto de ***n*** pares atributo-valor :

$$\{(A_i : V_{ij})\}$$

*donde **Vij** es el valor **j** del dominio **Di** asociado al atributo **Ai.***

4.- Esquema de la relación. Se forma con el nombre R y la cabecera. Es decir:

$$R\{(A_i: D_i)\}_{i=1}^{n}$$

5.- Estado de la relación. Lo forman el esquema y el cuerpo.

Ejemplo :

Esquema*: Impuestos vehículos (Vehículo:"Marcas", Dueño:"Nombre", Teléfono:"entero de longitud 9", Matricula:"Cadena de caracteres de longitud 7", Cuota:"decimal de longitud 4+ 2)*
Cuerpo: *{(Vehículo:* Seat Ibiza*, Dueño:* Francisco*, Telefono:* 925884721*,Matricula:* 9918-FTV *Cuota:* 92.24*), (Vehículo:* Volskwagen Polo*,Nombre:Pedro, Telefono:* 918773621*,Matricula:* 4231-FHD *,Cuota:* 61.98*), (Vehículo:* Renault Laguna*, Dueño:María, Telefono:* 929883762*, Matricula: 7416-*GSI*,Cuota: 145.32) (Vehículo:* Fiat Punto*, Dueño:Ernesto, Telefono:646556421, Matricula: 9287-*BHF*,Cuota: 45.77 }.*

4. Tipos de datos

La información que vamos a guardar va a estar relacionada en forma de filas y columnas. Las columnas son los atributos o información que nos interesa incluir del mundo real que estamos modelando.

Los atributos se mueven dentro de un dominio, que formalmente es un conjunto de valores. Pues bien, en términos de sistemas de base de datos, se habla más de tipos de datos que de dominios. Al crear la relación (tabla) decidimos qué conjunto de datos deberá ser almacenado en las filas de los atributos que hemos considerado. Tenemos que asignar un tipo de dato a cada atributo.

Con la asignación de tipos de datos, también habremos seleccionado un dominio para un atributo.

Cada campo:

- debe poseer un **Nombre** (relacionado con los datos que va a contener) y
- debe tener asociado un **Tipo de dato.**

Existen distintas formas de nombrar los tipos de datos dependiendo del lenguaje que utilicemos (C, Java, PHP, MySQL, SQL, Pascal, etc.).

Los tipos de datos más comunes con los que nos encontraremos generalmente son:

- **Texto**: almacena cadenas de caracteres (números con los que **no** vamos a realizar operaciones matemáticas, letras o símbolos).
- **Numérico**: almacena números con los que vamos a realizar operaciones matemáticas.
- **Fecha/hora**: almacena fechas y horas.
- **Sí/No**: almacena datos que solo tienen dos posibilidades (verdadero/falso).
- **Autonumérico**: valor numérico secuencial que el SGBD incrementa de modo automático al añadir un registro (fila).
- **Memo**: almacena texto largo (mayor que un tipo texto).
- **Moneda**: se puede considerar un subtipo de Numérico ya que almacena números, pero con una característica especial, y es que los valores representan cantidades de dinero.
- **Objeto OLE**: almacena gráficos, imágenes o textos creados por otras aplicaciones.

*5. C*laves

5.1 Clave candidata

Conjunto de atributos que identifican unívocamente cada tupla de la relación.
Es decir columnas cuyos valores no se repiten en ninguna otra tupla de esa tabla.
Toda tabla en el modelo relacional debe tener al menos una clave candidata (puede incluso haber más).
Un alumno viene identificado por su DNI, el DNI+Nombre, el NUSS….

5.2 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.
Si la tabla anterior de alumnos tuviese un código de alumno, éste sería mejor candidato (y por lo tanto clave principal) porque es mejor identificador para ese contexto.

5.3 Clave alternativa

Cualquier clave candidata que no sea primaria.

5.4 Clave externa, ajena, foránea o secundaria

Son los atributos de una relación que son claves en otra tabla.

6. Restricciones.

6.1. Inherentes

Son aquellas que no son determinadas por los usuarios, sino que son definidas por el hecho de que la base de datos sea relacional. Las más importantes son:

1. No puede haber dos tuplas iguales
2. El orden de las tuplas no es significativo
3. El orden de los atributos no es significativo
4. Cada atributo sólo puede tomar un valor en el dominio en el que está inscrito
5. Cada tabla tiene un nombre distinto
6. Cada atributo de la tabla toma un solo valor en cada tupla
7. Cada atributo tiene un nombre distinto en cada tabla (aunque puede coincidir en tablas distintas)

6.2. Semánticas

6.2.1.Restricciones de clave.

Se llama clave principal, clave primaria o PRIMARY KEY a uno o más atributos que identifican de forma única la tabla.
Por tanto y muy importante, estos atributos no podrán repetir valores ni tampoco dejarlos vacíos.

6.2.2. Restricciones de valor único (UNIQUE)

Impide que un atributo así marcado pueda repetir su valor.
Todos los atributos clave cumplen esta restricción.
También se debe indicar en todas las claves candidatas.

6.2.3.Restricciones de valor NULO (NULL O NOT NULL)

Un atributo será obligatorio si no admite el valor NULO o NULL, es decir el valor del desconocimiento de la información.
Si admite el valor NULL el valor será opcional.

6.2.4. Restricciones de dominio.

Esta restricción exige que el valor que pueda tomar un campo esté dentro del dominio definido.
Por ejemplo : El campo nota de un alumno tiene que ser numérico.

6.2.5.Restricciones de verificación (check)

Permite especificar condiciones que deban cumplir los valores de los atributos.Cada vez que se realice una inserción o una actualización de datos se comprueba si los valores cumplen la condición.Rechaza la operación si no se cumple.
Por ejemplo : La nota de un alumno tiene que estar comprendida entre 0-10.

6.2.6.Restricciones Genéricas o aserciones (ASSERT)

Son parecidas a la anterior, pero en este caso en lugar de afectar a una relación como CHECK, puede afectar a dos o más relaciones .La condición se establece sobre elementos de distintas relaciones. Pueden implicar a subconsultas en la condición.

Por ejemplo, el siguiente código contiene dos aserciones:

```
x := 5;
{x > 0}
x := x + 1
{x > 1}
```

x > 0 y x > 1, y las dos son ciertas en dichos puntos de la ejecución.

Las aserciones suelen ser útiles para especificar programas y para razonar la corrección de los mismos. Por ejemplo, una precondición (una aserción al comienzo de una sección de código) determina que se espera que el conjunto de sentencias que la siguen sean ejecutadas. Una postcondición (colocada al final) describe el estado que se espera alcanzar al final de la ejecución.

6.2.7.Restricciones de integridad referencial o restricción de clave ajena (FOREIGN KEY)

Se da cuando en una tabla (llamada secundaria) existe referencia a algún valor de otra tabla (llamada principal).
En este caso la restricción exige que exista el valor referenciado en la otra tabla. Por ejemplo : No se puede poner una nota a un alumno si este no existe.

Si el atributo de la tabla principal es clave de esa tabla, tenemos en la tabla secundaria una foreign key, también llamada secundaria y foránea.

Los atributos marcados de esta forma sólo podrán contener valores que estén relacionados con la clave principal de la tabla que relacionan (llamada tabla principal). Eso causa problemas en las operaciones de borrado y modificación de registros; ya que si se ejecutan esas operaciones sobre la tabla principal (*si* se modifica o borra un alumno) quedarán filas en la tabla secundaria con la clave externa haciendo referencia a un valor que ya no existe en la tabla principal.

Para solventar esta situación se puede hacer uso de estas opciones:

1. **Prohibir la operación** (***no action)***.

2. **Transmitir la operación en cascada (CASCADE**). Es decir si se modifica o borra un alumno; también se modificarán o borrarán las notas relacionados con él.

3. **Colocar nulos *(set null***) Las referencias al alumno en la tabla de notas se colocan como nulos.

4. **Usar el valor por defecto *(default).*** Se coloca un valor por defecto en las claves externas relacionadas. Este valor se indica al crear la tabla (opción **default**).

6.2.8. Disparadores o triggers.

Se trata de pequeños programas grabados en la base de datos que se ejecutan automáticamente cuando se cumple una determinada condición. Sirven para realizar una serie de acciones cuando ocurre un determinado evento (cuando se añade una tupla, cuando se borra un dato, cuando un usuario abre una conexión…)

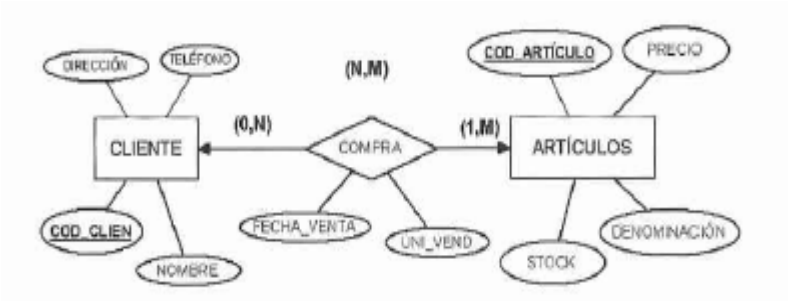
7. Transformación del modelo E-R al modelo relacional

Las reglas básicas para transformar un esquema conceptual E-R a un esquema relacional son las siguientes:

- **Entidades.** Las entidades se transforman en tablas
- **Atributos**. Los atributos se transforman en columnas dentro de una tabla.
- **Identificadores principales**. Pasan a ser claves primarias
- **Identificadores candidatos**. Pasan a ser claves candidatas.

7.1. Relaciones de cardinalidad N:M

Toda relación N:M se transforma en una tabla que tendrá como clave primaria la concatenación de los atributos clave de las entidades que asocia.



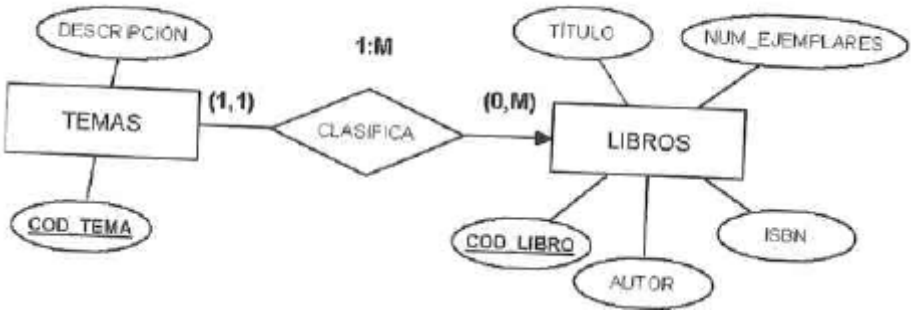
CLIENTE (**COD_CLIEN**, NOMBRE, DIRECCIÓN, TELEFONO)
ARTICULOS (**COD_ARTICULO**, PRECIO, STOCK, DENOMINACIÓN)

COMPRA (**COD_CLIEN (FK)**, **COD_ARTICULO(FK)**, UNI_VEND, FECHA_VENTA)

7.2. Relaciones de cardinalidad 1:N

En la transformación de relaciones 1:N existen dos soluciones :

- **Transformar la relación en una tabla**. Se hace como si se tratara de una relación N:M. Esta solución se realiza cuando se prevé que en un futuro la relación se convertirá en N:M y cuando la relación tiene atributos propios. También se crea una nueva tabla cuando la cardinalidad es opcional, es decir, (0,1) y (0,M).La clave de esta tabla es la de la entidad del lado muchos
- **Propagar la clave** . Este caso se aplica cuando la cardinalidad es obligatoria,es decir,cuando tenemos cardinalidad (1,1) y (0,M) o (1,M).Se propaga el atributo principal de la entidad que tiene de cardinalidad máxima 1 a la que tiene de cardinalidad máxima N,desapareciendo el nombre de la relación.Si existen atributos propios en la relación,estos también se propagarán.



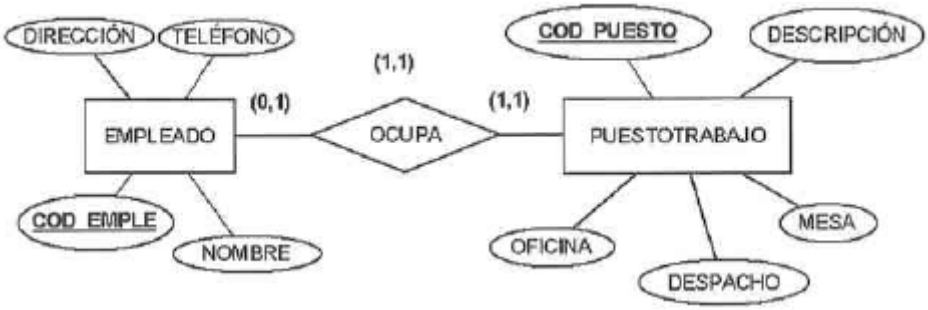
TEMAS (**COD_TEMA** ,DESCRIPCIÓN)
LIBROS (**COD_LIBRO**,AUTOR,ISBN,TÍTULO,NUM_EJEMPLARES,**COD_TEMA(FK**))

7.3. Relaciones de cardinalidad 1:1

En la transformación de relaciones 1:1 se tienen en cuenta las cardinalidades de las entidades que participan.Existen dos soluciones:

- **Transformar la relación en una tabla**. Si las entidades poseen cardinalidades (0,1) la relación se convierte en tabla.

- **Propagar la clave** :
  - Si una de las entidades posee cardinalidad (0,1) y la otra (1,1) conviene propagar la clave de la entidad con cardinalidad (1,1) a la tabla resultante de la entidad de cardinalidad (0,1).
  - Si ambas entidades poseen cardinalidades (1,1) se puede propagar la clave de cualquiera de ellas a la tabla resultante de la otra.En este caso ,también se pueden añadir los atributos de una entidad a otra,resultando una única tabla con todos los atributos de las entidades y de la relación,si los hubiera,eligiendo como clave primaria una de las dos.



PUESTOTRABAJO (**COD_PUESTO**, DESCRIPCIÓN, OFICINA, DESPACHO, MESA)
EMPLEAD (**COD_EMPLE**, NOMBRE, DIRECCIÓN, TELEFONO, **COD_PUESTO**(FK))

7.4. Relaciones reflexivas .

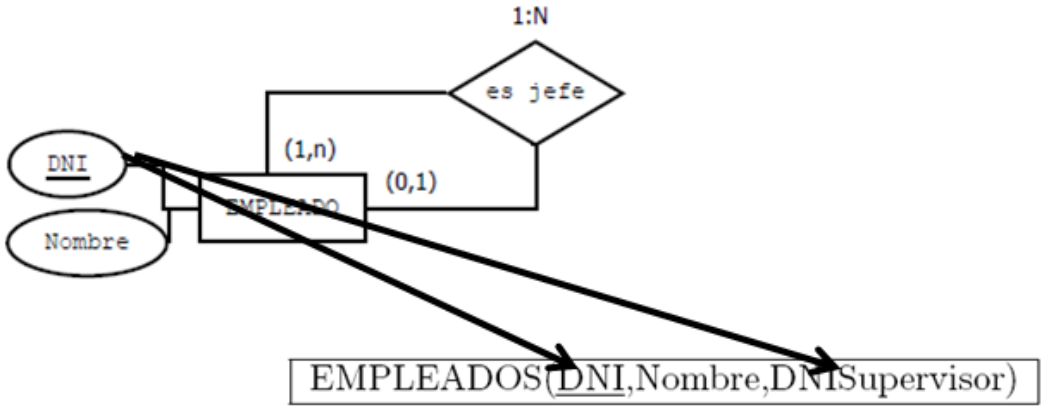
Son relaciones binarias en las que participa un tipo de entidad. En el proceso de convertir un relación reflexiva a tabla hay que tener en cuenta sobre todo la cardinalidad.Lo normal es que toda relación reflexiva se convierta en dos tablas, una para la entidad y otra para la relación.

Se pueden presentar los siguientes casos:

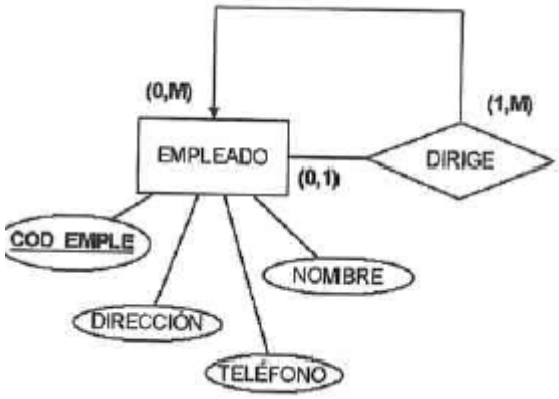
- Si la relación es 1:1, la clave de la entidad se repite, con lo que la tabla resultante tendrá dos veces ese atributo, una como clave primaria y otra como clave ajena de ella misma. No se crea la segunda tabla.

- Si la relación es 1:M, podemos tener dos casos:

    - Caso de que la entidad muchos sea siempre obligatoria se procede como en el caso 1:1.



- Si no es obligatoria, se crea una nueva tabla cuya clave será la de la entidad del lado muchos,y además se propaga la clave a la nueva tabla como clave ajena



EMPLEADO (**COD EMPLE**, DIRECCIÓN, TELÉFONO, NOMBRE)
DIRIGE (**COD EMPLE(FK)**, **COD_DIREC(FK)**)

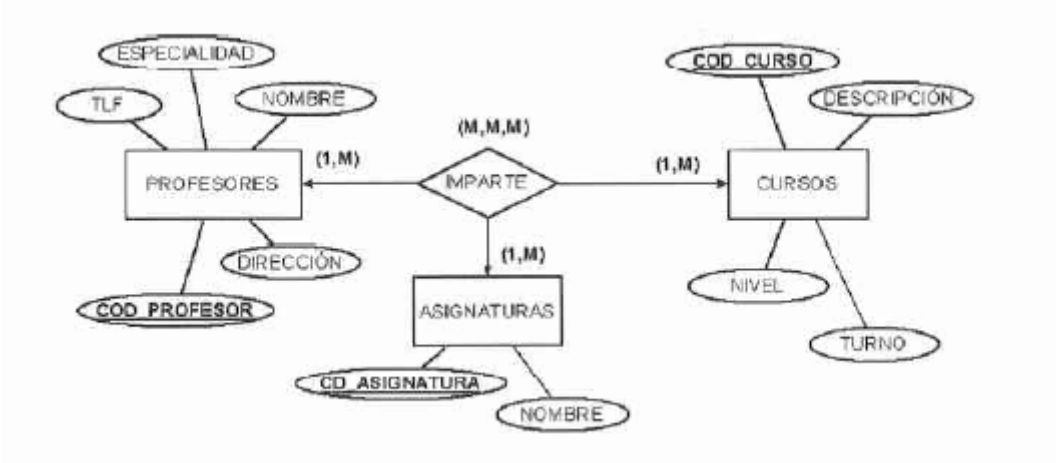
En este caso se considera que un empleado puede dirigir a muchos empleados o a ninguno si es el que dirige.Y un empleado es dirigido por un director o por ninguno si el es el que dirige.En este caso no hay obligatoriedad en la entidad muchos.

- Si la relación es N:M, la tabla resultante de la relación contendrá dos veces la clave primaria de la entidad del lado M más los atributos de la relación si los hubiera.La clave de la nueva tabla será la combinación de las dos.

7.5. Relaciones Ternarias.

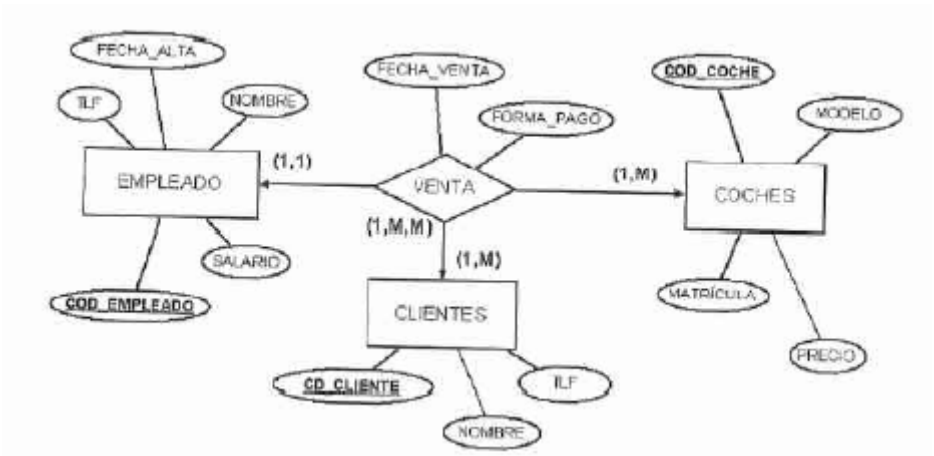
Cada entidad se convierte en tabla,así como la relación que va a contener los atributos propios de ella más las claves de todas las entidades.La clave de la tabla resultante será la concatenación de las claves de las entidades.

- Si la relación es N:N: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



PROFESORES (**COD_PROFESOR**, DIRECCIÓN, NOMBRE, TLF, ESPECIALIDAD)
CURSOS (**COD_CURSO**, DESCRIPCIÓN, NIVEL, TURNO)
ASIGNATURAS (**CD_ASIGNATURA**, NOMBRE)
IMPARTE (**COD_PROFESOR(FK)**, **COD_CURSO(FK)**, CD_ASIGNATURA(FK))

- Si la relación es 1:N:N ,la clave de la entidad con cardinalidad máxima 1 no pasa a formar parte de la clave de la tabla resultante,pero forma parte de la relación como un atributo más.



CLIENTES (**CD_CLIENTE**, NOMBRE, TLF)
EMPLEADO (**COD_EMPLEADO**, NOMBRE, TLF, SALARIO, FECHA_ALTA)
COCHES (**COD_COCHE**, MATRICULA, MODELO, PRECIO)
VENTA (**COD_COCHE(FK)**, **CD_CLIENTE(FK)**, COD_EMPLEADO, FORMA_PAGO, FECHA_VENTA)

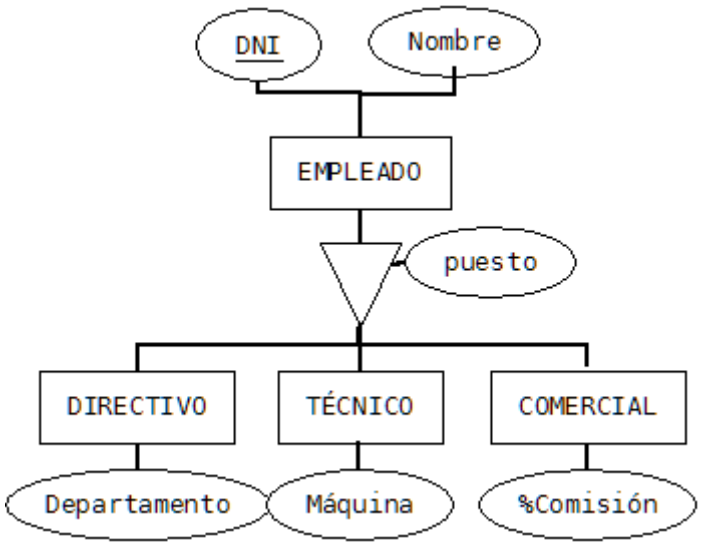
7.6. Generalizaciones y especializaciones

El modelo relacional no dispone de mecanismos para la representación de las relaciones jerárquicas,así pues,las relaciones jerárquicas se tienen que eliminar.Para ello hay que tener en cuenta :

- ✓ La especialización que los subtipos tienen respecto a los supertipos,es decir,los atributos diferentes que tengan asociados cada uno de los subtipos,que son los que se diferencian con el resto de atributos de los otros subtipos.
- ✓ El tipo de especialización que representa el tipo de relación jerárquica : total o parcial exclusiva y total o parcial solapada
- ✓ Otros tipos de relación que mantengan tanto los subtipos como el supertipo
- ✓ La forma en la que se va a acceder a la información que representan tanto el supertipo como el subtipo.

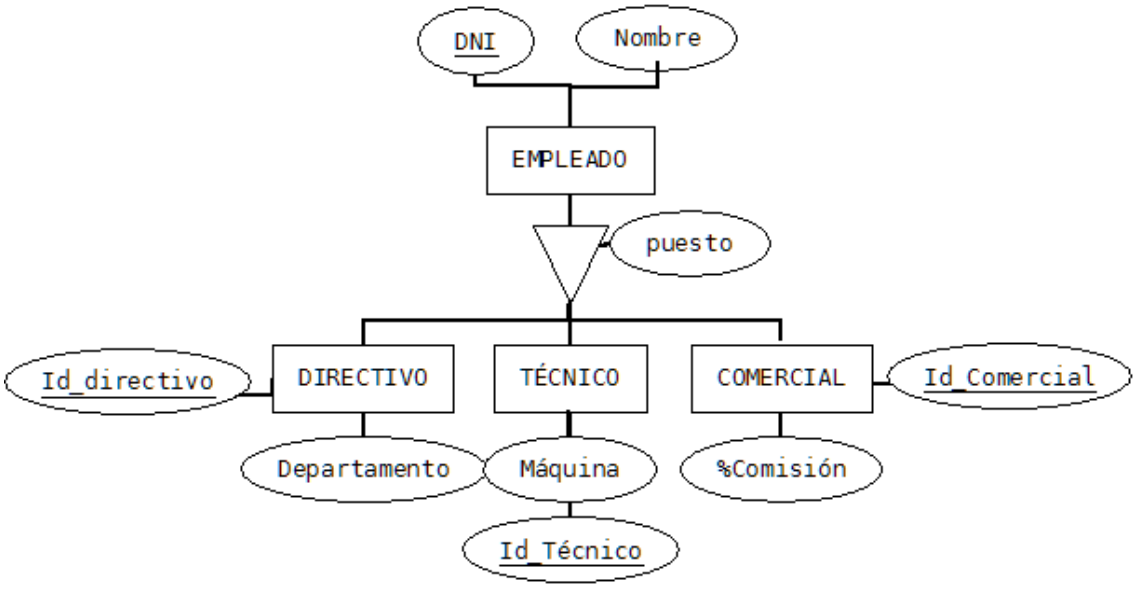
Caso general :

Se crea una tabla para el supertipo ó superclase y otras tantas para cada subclase ó subtipo incorporando el campo clave de la superclase a las tablas de las subclases.



EMPLEADOS (**DNI**, Nombre, Puesto)
DIRECTIVOS (**DNI ,** Departamento)
TÉCNICOS(**DNI** ,Máquinas)
COMERCIALES(**DNI** , Comisión)

En el caso de que las subentidades no hereden el identificador con la superentidad, se colocará en las subentidades el identificador de la superentidad como clave secundaria, además será clave alternativa



EMPLEADOS (**DNI,** Nombre, Puesto)
DIRECTIVOS (**Id_Directivo** , Departamento, DNI(FK))
TÉCNICOS(**ID_Técnico**, Máquinas, DNI(FK))
COMERCIALES(**Id_Comercial** , Comisión, DNI(FK))

En ocasiones y dependiendo del tipo de relación puede convenirnos optar por uno de los casos siguientes, hay que estar seguro que la semántica no se pierde para optar por una de estas opciones. El caso general es válido en todos los casos.

a) Se puede crear una tabla para cada subclase incorporando los atributos de la clase padre y no crear una tabla para la supeclase. Esto define bien a las **totales**.

DIRECTIVOS (**DNI**, Nombre, Puesto, Departamento)
TÉCNICOS(**DNI**, Nombre, Puesto, Máquinas)
COMERCIALES(**DNI,** Nombre, Puesto,Comisión)

b) Se puede crear una sola tabla para la superclase, incorporando los atributos de todas las subclases y añadir, para distiguir el tipo de la superclase, un

campo llamado “tipo” , que contendrá el tipo de subclase al que representa cada tupla.
Este tipo de opción se adapta muy bien a las especializaciones **exclusivas.**

EMPLEADOS (**DNI** ,Nombre, Puesto, Departamento ,Máquinas, Comisión,Tipo)

c) Se puede crear una sola tabla para la superclase pero esta vez en lugar de añadir un solo campo “tipo” Se añaden campos que indican si cumplen un perfil.

Esta forma representa bien las **especializaciones inclusivas.**

EMPLEADOS (**DNI** ,Nombre, Puesto, Departamento ,Máquinas, Comisión, EsDirectivo, EsComercial, EsTécnico)