





### objetivos

Los objetivos de este tema son:

1. Conocer la estructura básica del modelo: la mesa.
2. Saber describir los campos, con su dominio, y saber elegir la clave principal.
3. Saber definir otras restricciones (no nulo, único, ...)
4. Entender el concepto de clave externa, y saber definir las necesarias.
5. Saber traducir correctamente del Modelo Entidad-Relación al Modelo Relacional.
6. Redefinir las tablas necesarias atendiendo a motivos prácticos.



### conocimientos previos

Los conocimientos previos necesarios para el tema son los del tema 2: Modelo Entidad-Relación. En cierto modo este tema aclarará conceptos de aquel tema, ya que nos permitirá aplicarlo a algo palpable como son las mesas, directamente implementables en un SGBD Relacional. Así podremos concretar posibles ambigüedades del esquema Entidad-Relación, y saber si valen la pena determinadas restricciones.

Pero entender el Modelo Entidad-Relación y dominar la interpretación de un esquema será básico en el diseño de una Base de Datos. Siempre empezaremos por el Modelo Entidad-Relación, y posteriormente el traduciremos al Modelo Relacional, aplicando en todo caso el sentido común.

1970 **EF Codd** introdujo la teoría matemática de las relaciones en el campo de los SGBD. El modelo de datos que creó llama **MODELO RELACIONAL**, con una base matemática (la de las relaciones) muy sólida, donde los datos se estructuran en forma de relaciones (tablas) que son estructuras de datos simples y uniformes, que permiten una fácil comprensión.

En el momento en que surgió, los modelos que funcionaban eran el jerárquico y el de red (también llamado CODASYL, por el grupo de trabajo que estandarizar-lo). Como modelo los superó porque, como dijo el mismo Codd, "*proporciona un medio para describir los datos con su estructura únicamente, sin tener que superponer ninguna estructura adicional para representarse en la máquina*" (es decir sin punteros ni historias).

A pesar de ello estuvo unos años en competencia con aquellos, con mucha gente muy reticente, ya que los productos comerciales que salían no eran suficientemente eficientes, tal vez porque la tecnología de la época no lo permitía. A partir de los años 80, cuando la tecnología lo permitió, salieron productos mejores, como por ejemplo el **ORACLE** (1979), y entonces su implantación fue aplastante.

Los objetivos del Modelo Relacional son:

- **Fidelidad**, para originar esquemas que representan fielmente la información (los objetos y relaciones entre ellos) que existe en el dominio del problema.
- **Independencia física** [1], para que el modo de guardar los datos no influya en su manipulación lógica, y así los usuarios que acceden a estos datos no tengan que modificar sus programas por cambios en el almacenamiento físico.
- **Independencia lógica**, para que las vistas externas no se vean afectadas por cambios en el esquema conceptual de la BD
- **Flexibilidad**, para poder ofrecer los datos a cada usuario de la forma más adecuada a su aplicación.
- **Uniformidad**, las estructuras lógicas de los datos presentan un aspecto simple y uniforme (las tablas), lo que facilita la concepción y manipulación por parte de los usuarios.
- **Sencillez**, las características anteriores, unidas a unos lenguajes de usuario sencillos, hacen que el M. Relacional sea fácil de entender y de utilizar para el usuario final.

Recuerde que las **relaciones** del Modelo Relacional son las **tablas**. No son lo mismo que las relaciones del Modelo Entidad-Relación. Para evitar confusiones intentaremos llamarlas siempre tablas.

---

[1] En realidad tanto la independencia física como lógica las han intentadas conseguir todos los modelos.

## 2. Estructura del Modelo Relacional

El elemento básico del Modelo Relacional es la **RELACIÓN**, que será una tabla o matriz bidimensional con unas características o restricciones que comentaremos más adelante.

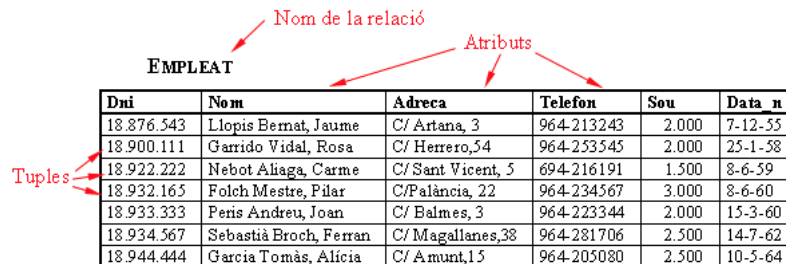


Diagrama de una relación relacional con anotaciones:

- Nom de la relació**: Puntero a la etiqueta **EMPLEAT**.
- Atributs**: Punteros a los encabezados de columna (**Dni**, **Nom**, **Adreca**, **Telefon**, **Sou**, **Data\_n**).
- Tuples**: Punteros a las filas de datos.

Dni	Nom	Adreca	Telefon	Sou	Data_n
18.876.543	Llopis Bernat, Jaume	C/ Artana, 3	964-213243	2.000	7-12-55
18.900.111	Garrido Vidal, Rosa	C/ Herrero, 54	964-253545	2.000	25-1-58
18.922.222	Nebot Aliaga, Carme	C/ Sant Vicent, 5	694-216191	1.500	8-6-59
18.932.165	Folch Mestre, Pilar	C/ Palància, 22	964-234567	3.000	8-6-60
18.933.333	Peris Andreu, Joan	C/ Balmes, 3	964-223344	2.000	15-3-60
18.934.567	Sebastià Broch, Ferran	C/ Magallanes, 38	964-281706	2.500	14-7-62
18.944.444	Garcia Tomàs, Alicia	C/ Amunt, 15	964-205080	2.500	10-5-64

Normalmente una relación tiene un **NOMBRE** (pe **Empleado**) aunque ocasionalmente no tendrá, por ejemplo una mesa que sea el resultado de una consulta poco frecuente.

Las **FILAS**, donde tenemos la información de las ocurrencias, los individuos, también se llaman **TUPLAS** (a veces por similitud con archivos también se llaman **REGISTROS**).

Las **COLUMNAS**, que serán características que nos interesan los individuos y que en cada tupla toma un valor, las llamaremos también **ATRIBUTOS** (o **CAMPOS**).

El conjunto de valores posibles que puede tomar un atributo determinado se llama **DOMINIO**. Más adelante veremos que los dominios se intentarán definir lo mejor posible, para prevenir errores.

El **ESQUEMA o ESTRUCTURA DE LA RELACIÓN** es la definición de la relación, es decir, atributos que tiene, dominios de estos y restricciones que podremos definir, que veremos en la siguiente pregunta.

El **ESTADO DE LA RELACIÓN** es la información que contiene en un determinado momento. Normalmente el estado variará continuamente a lo largo del tiempo, bien porque se añaden nuevas tuplas (aumenta la cardinalidad), bien porque se modifica el valor de algún atributo en alguna tupla. En cambio el esquema difícilmente cambiará.

Una **CLAVE CANDIDATA** es un atributo o conjunto de atributos que identifican unívocamente cada tupla de la relación. En el ejemplo podrían ser claves candidatas **Dni**, **Nombre**, incluso nos podríamos plantear combinaciones, como el conjunto (**Nombre**, **Data\_n**), ya que parece imposible que dos personas de la empresa se digan igual y encima hayan nacido el mismo día. De entre todas las claves candidatas elegiremos una, que será la **LLAVE PRINCIPAL** o **CLAVE PRIMARIA**, y servirá para identificar de forma efectiva en el Modelo cada una de las tuplas.

Podría darse el caso de que un atributo no coja ningún valor para una tupla determinada, por ejemplo, un empleado que no tenga teléfono. Entonces le daremos el **VALOR NULO**.

Por último, las relaciones o tablas pueden ser **permanentes** o **TEMPORALES**. Las primeras se guardan. Las segundas, normalmente resultado de una consulta ocasional, no.

Representaremos la tabla con el nombre de la tabla en mayúsculas seguido, entre paréntesis, en minúsculas y separados por comas, por los nombres de los campos, con la clave principal subrayada. También es conveniente huir de los caracteres especiales (vocales acentuadas, ç, ñ, guión, ...) para no tener problemas cuando vamos a implementar en un SGBD determinado (Access, Oracle, PostgreSQL, ...). Para una mejor lectura intentaremos poner siempre la clave principal al principio, el o los primeros campos.

```
EMPLEADO ( dni , nombre, direccion, telefono, sueldo, data_n)
```

También podemos utilizar una forma alternativa de representarla, con un recuadro que coge toda la tabla, arriba el nombre de la tabla, y bajo cada uno de los campos, poniendo la clave principal en negrita o subrayada.

EMPLEAT
•dni
°nom
°adrec
°telefon
°sou
°data_n

Al igual que en otros modelos de datos, en el Modelo Relacional existen restricciones, es decir, estructuras u ocurrencias no permitidas.

Estas restricciones pueden ser de dos tipos fundamentales: **restricciones inherentes** , que son impuestas por el propio modelo, y **restricciones de usuario** (también llamadas restricciones **semánticas** ) en las que es el usuario quien prohíbe, porque el modelo se lo permite, determinadas circunstancias para poder definir mejor la Base de Datos.

## 3.1 Restricciones inherentes

---

Como hemos dicho son las que impone el propio modelo. Algunas son características que deben cumplir las relaciones. Por lo tanto no cualquier tabla matemática es una relación. Podemos considerar las siguientes:

- **Valores atómicos** : cada valor de la tabla, es decir, cualquier valor de cualquier atributo de cualquier tupla debe ser simple, no divisible. Por lo tanto no valen atributos compuestos o repetitivos.

Así, si consideramos **Nombre** en la relación **Empleado** como nombre de pila más apellidos, no será divisible (no podré coger posteriormente el nombre de pila por un lado y los apellidos por otra; si lo quisiera hacer, se deberían definir los atributos simples **Apellido1** , **Apellido2** y **Nombre** ).

Tampoco valen valores repetitivos, por ejemplo un vector de 12 entradas. Quedan por tanto descartados los atributos ***multivaluados*** .

- **Tuplas distintas** : no pueden haber dos tuplas iguales. Esto es una diferencia respecto a las tablas matemáticas donde sí se pueden duplicar filas.
- **El orden de las tuplas no es significativo** .
- **El orden de los atributos no es significativo** .

Las anteriores son condiciones, imposiciones que nos da el mismo modelo.

Las realmente interesantes para nosotros son las restricciones de usuario, también llamadas restricciones semánticas, ya que serán condiciones que podremos poner nosotros para que el esquema de la BD explique lo mejor posible la realidad, y evitar posibles errores en los datos.



### 3.2.1 Restricción de dominio

El valor de un atributo debe ser un valor atómico del dominio. Definiendo claramente el dominio nos aseguramos (dentro de lo posible) que el atributo no pueda coger valores incorrectos.

Por un lado, el dominio será de un tipo determinado, elegido de una gama bastante extensa: entero corto, entero, entero largo, real, doble precisión, carácter, cadena de caracteres (texto), fecha, hora, ...

Así, por ejemplo, definiendo el *Sois* como un número real impediremos que por error pueda coger el valor 2. **R** 00'00, o que la *fecha de nacimiento* , con dominio de tipo fecha, cualquiera 15- **14** -1958 o **31** - **2** -1958.

También se podrán definir dominios que estén en un determinado intervalo (nota de un examen: 0-10) o de un tipo enumerado (nota de evaluación: MD, IN, SUF, BE, NOT, EXC).

Se pueden definir reglas de verificación o validación ( **CHECK** ) que nos ayudan a perfilar muy bien el dominio. Este sería un ejemplo de exigir que el trabajador tenga de 18 a 65 años. La sintaxis es inventada, sólo para entenderlo, con una hipotética función que saca el año, y otra que nos da la fecha de hoy.

```
CHECK (Año (hoy - data_n) >= 18) and (Año (hoy - data_n) < 65)
```

#### Nota

En **Access** la regla de validación sería:

```
AgregFecha ( "aaaa"; 18; [data_n]) <= Ahora () Y AgregFecha ( "aaaa"; 65; [data_n]) >= Ahora ()
```

donde se ve que la fórmula es añadir 18 años a la fecha de nacimiento y comprobar que no supera la fecha de hoy; y añadir 65 años a la fecha de hoy y comprobar que sí supera la fecha de hoy.

Por ejemplo, **Empleado** podría quedar:

```
EMPLEADO ( dni : carácter (10), nombre: carácter (30), direccion: carácter (30), teléfono: entero (9), sueldo: numérico (6,2), data_n: fecha)
```

con la regla de validación:

```
CHECK Sois > 0
```

aparte de la mencionada anteriormente.

Si utilizamos el modo de representar alternativa, lo tendríamos así:

EMPLEAT	
*dni	carácter(10)
°nom	carácter(30)
°adreca	carácter(30)
°telefon	enter(9)
°sou	numeric(6,2)
	sou > 0
°data_n	data

La siguiente imagen muestra como se haría en **Access** la primera regla de validación, aplicada al campo **data\_n**

Base de datos1 : Base de datos (Access 2007) - Micr...

Inicio Crear Datos externos Herramientas de base de datos **Diseño**

Ver Vistas Clave principal Generador de reglas de validación Probar reglas de validación Eliminar filas Insertar filas Columna de búsqueda Hoja de propiedades Índices Mostrar u ocultar

**Advertencia de seguridad** Se ha deshabilitado parte del contenido de la base de datos Opciones...

Todas las ta... << **Empleat** x

Empleat Empleat : Tabla

Nombre del campo	Tipo de datos	Descripción
dni	Texto	
nom	Texto	
adrec	Texto	
telefon	Número	
sou	Número	
data_n	Fecha/Hora	

Propiedades del campo

General	Búsqueda
Formato	
Máscara de entrada	
Título	
Valor predeterminado	
Regla de validación	AgregFecha("aaaa";18;[data_n]<=Ahora() Y AgregFecha("aaaa";65;[data_n]>=Ahora())
Texto de validación	
Requerido	No
Indexado	No
Modo IME	Sin Controles
Modo de oraciones IME	Nada
Etiquetas inteligentes	
Alineación del texto	General
Mostrar el Selector de fecha	Para fechas

Un nombre de campo puede tener hasta 64 caracteres de longitud, incluyendo espacios. Presione F1 para obtener ayuda acerca de los nombres de campo.

Vista Diseño. F6 = Cambiar paneles. F1 = Ayuda. Bloq Num

## 3.2.2 Restricción de clave principal

Permite declarar un atributo o un conjunto de atributos como **CLAVE PRINCIPAL** o **PRIMARIA** (Primary Key). Esta clave principal servirá para identificar unívocamente cada una de las filas.

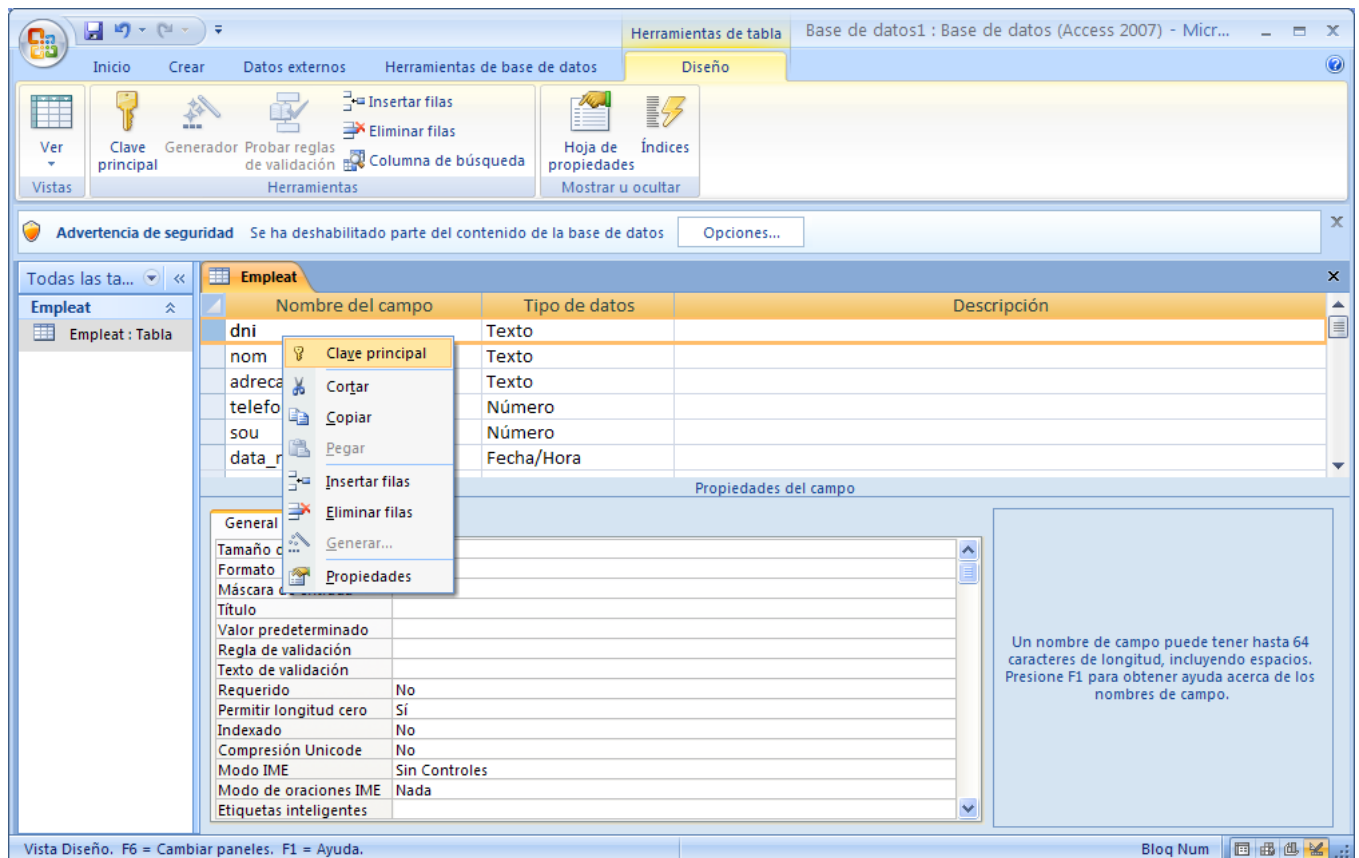
Como consecuencia de lo anterior, la clave principal no podrá tomar valores nulos ni repetidos, ya que de lo contrario no se podría asegurar la identificación de las filas.

Estas últimas características también las podrán tener otros atributos, lo que da lugar a los tipos de restricción que se ven en los siguientes puntos.

Nosotros siempre definiremos una clave principal. Si tenemos claves candidatas, elegimos una de ellas como clave principal. Si no tenemos, nos inventamos un campo que servirá de clave principal (bien un número o bien un código alfanumérico).

No es conveniente que la clave principal esté formada por un número excesivo de campos. Podríamos decir que 3 es el máximo. Si la clave candidata está formada por más de 3 campos, o bien elegimos otra clave candidata, o bien nos inventamos una.

En **Access** se marca la clave principal con una figura de una clave. En la figura se ve como hacer que el campo **dni** sea la clave principal de la tabla **EMPLEADO** :



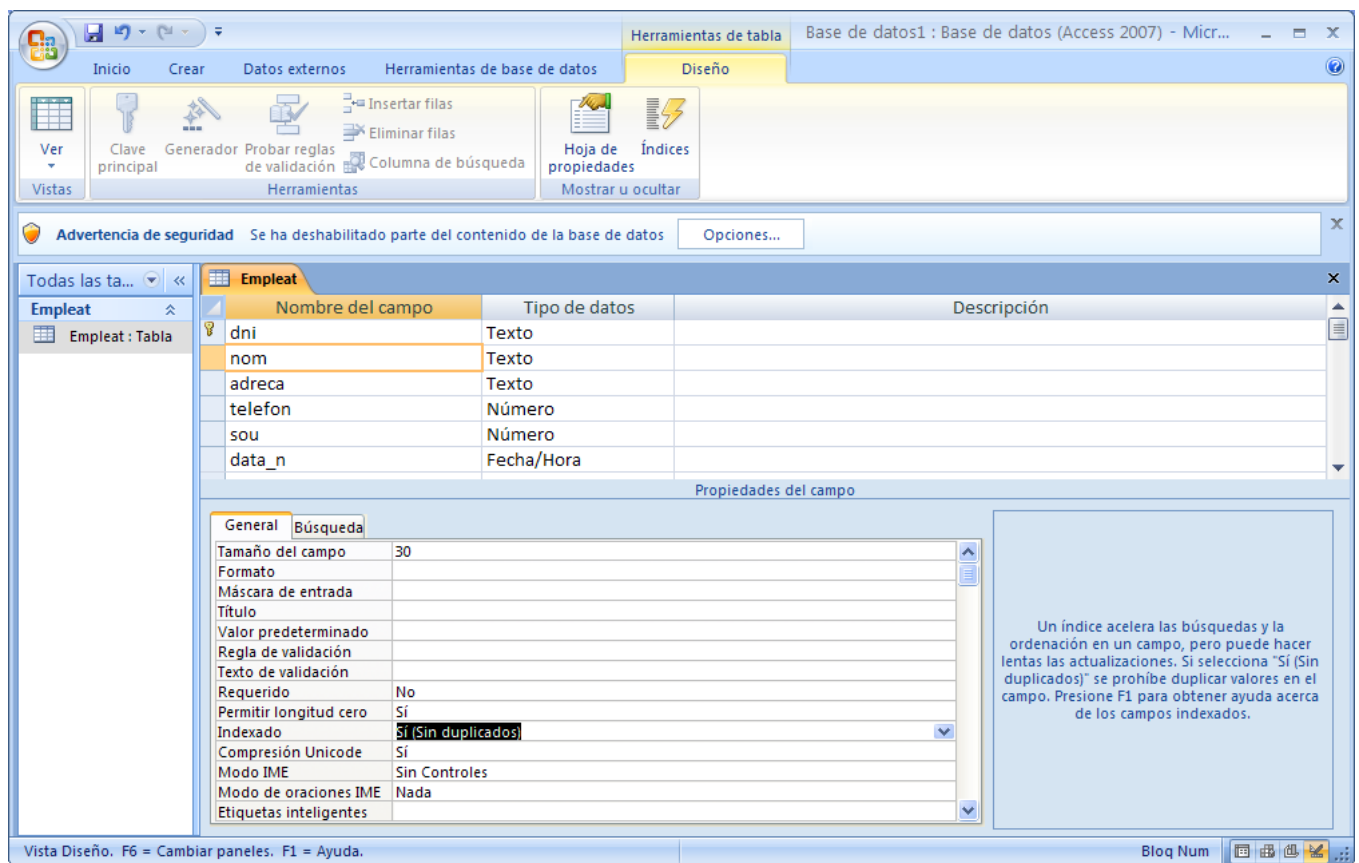
### 3.2.3 Restricción de unicidad

Si en un campo, o en un conjunto de campos, definimos la restricción de unicidad ( **UNIQUE** ), esto obliga a que, en caso de tener valores el campo, no se puedan repetir. Supongamos, por ejemplo, los alumnos de un Instituto. La clave no puede ser el DNI, ya que algunos alumnos no tendrán, pero en caso de tenerlo, es claro que no se podrá repetir.

Representaremos que un campo es único, poniendo **único** entre paréntesis bajo del campo. Por ejemplo, si consideramos que el campo nombre de la tabla EMPLEADO debe ser único, lo representaremos así:

```
EMPLEADO ( dni , nombre, direccion, telefono, sueldo, data_n)  
           (Único)
```

En Access la restricción de unicidad ( **UNIQUE** ) se define poniendo en el apartado *indexadas* el valor **Sí (sin duplicados)** . En la figura se muestra cómo hacer que el campo **nombre** de la tabla **EMPLEADO** sea único.



### 3.2.4 Restricción de valor no nulo

Obliga a que el campo coja siempre un valor. Por ejemplo, el campo **Nombre** es un buen candidato a ser no nulo.

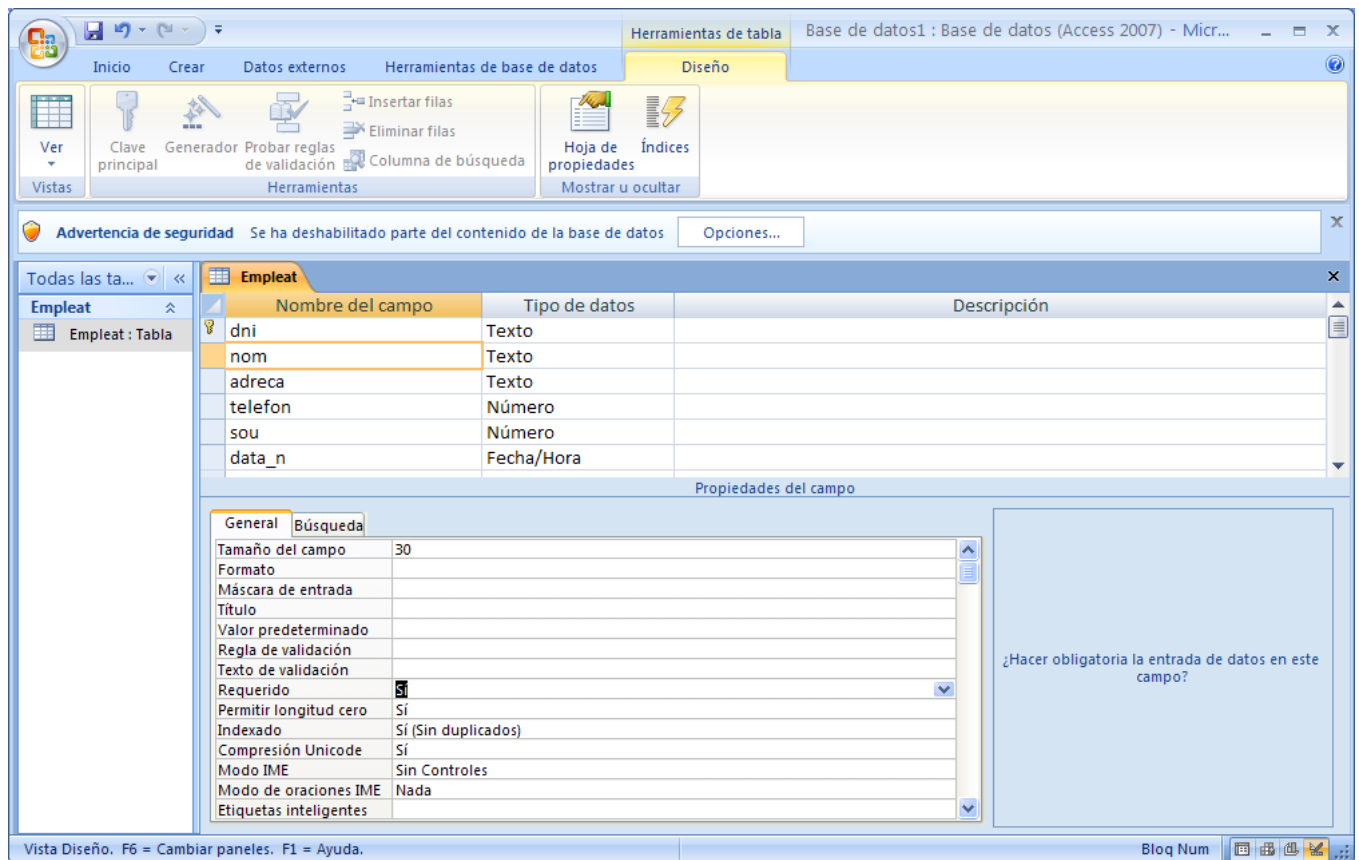
Lo representaremos poniendo **no nulo** entre paréntesis bajo del campo.

```
EMPLEADO ( dni , nombre, direccion, telefono, sueldo, data_n)  
          (No nulo)
```

Por medio de la representación alternativa, podemos marcar con un punto negro delante del campo no nulo.

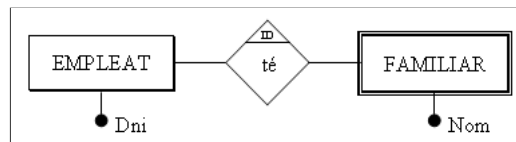
EMPLEAT
*dni
*nom
*adrec
*telèfon
*sou
*data_n

En Access la manera de poner un campo como **NOT NULL** será poner en el apartado **Requerido** el valor **Sí**



## 3.2.5 Integridad referencial

Para poder explicarla nos apoyaremos en un ejemplo.



que podría traducirse en las siguientes tablas:

EMPLEAT		FAMILIAR			
dni	nom	dni_emp	nom	data_n	parentesc
18.876.543	Llopis Bernat, Jaume	18.876.543	Jaume Llopis Doménech	01/06/85	Fill
18.900.111	Garrido Vidal, Rosa	18.900.111	Felip Gomis Pitarch	31/03/57	Conjuge
18.922.222	Nebot Aliaga, Carme	18.932.165	Josep Serra González	09/05/60	Conjuge
18.932.165	Folch Mestre, Pilar	18.932.165	Xavier Serra Folch	05/04/90	Fill
18.933.333	Peris Andreu, Joan	18.933.333	Silvia Peris Prades	17/07/95	Filla
18.934.567	Sebastià Broch, Ferran	18.933.333	Silvia Prades Arnau	15/03/62	Conjuge
18.944.444	Garcia Tomàs, Alicia	18.944.444	Andreu Garcia Torró	15/09/22	Pare
		18.944.444	Laura Almela Garcia	05/12/92	Filla
		18.944.444	Lluís Almela Garcia	10/05/94	Fill
		18.944.444	Marc Almela Tomeu	25/03/64	Conjuge

Si en una mesa R2 ( **Familiar** ) tenemos un atributo ( **dni\_emp** ) que es clave (primaria o candidata) de otra tabla R1 ( **Empleado -> dni** ), todo valor de ese atributo debe concordar con un valor de la clave de R1 (no he de poder poner en familiar un Dni que no lo tenga ningún empleado de la empresa). El atributo en R2 es, por tanto, una **CLAVE EXTERNA**.

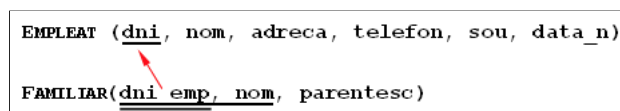
Debe ser imposible poner en **Familiar** el Dni 18754321, porque no está en la otra, y por tanto no es un Dni de un empleado de la empresa.

Las relaciones R1 y R2 no tienen por qué ser distintas, pueden ser la misma. Así, si consideramos el supervisor, éste debe ser de la empresa:

EMPLEAT						
dni	nom	adreca	telefon	sou	data_n	supervisor
18.876.543	Llopis Bernat, Jaume	C/ Artana, 3	964.213243	2.000	7-12-55	18.933.333
18.900.111	Garrido Vidal, Rosa	C/ Herrero, 54	964.253545	2.000	25-1-58	18.932.165
18.922.222	Nebot Aliaga, Carme	C/ Sant Vicent, 5	694.216191	1.500	8-6-59	18.944.444
18.932.165	Folch Mestre, Pilar	C/ Palància, 22	964.234567	3.000	8-6-60	
18.933.333	Peris Andreu, Joan	C/ Balmes, 3	964.223344	2.000	15-3-60	18.944.444
18.934.567	Sebastià Broch, Ferran	C/ Magallanes, 38	964.281706	2.500	14-7-62	
18.944.444	Garcia Tomàs, Alicia	C/ Amunt, 15	964.205080	2.500	10-5-64	18.933.333

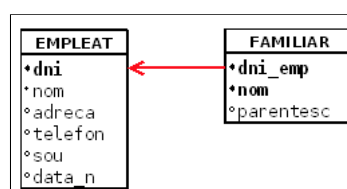
Supervisor es una clave externa, pero de la misma mesa. No todos los SGBD permiten una clave externa reflexiva.

Una manera de representar las claves externas en el esquema es la siguiente:



Donde el doble subrayado indica una clave externa (que en este ejemplo, además, forma parte de la clave principal, pero no siempre será así) que "apunta" a la llave principal de la otra tabla.

La otra manera, utilizando la forma alternativa, será esta, que también es muy fácil de entender:



Esto nos impedirá que introducimos valores no correctos, no existentes. ¿Pero qué pasará si borramos un empleado, o si modificamos su Dni? ¿Qué hacemos con los familiares? Pues en principio tres podrían ser las acciones a realizar, y dependiendo de la situación particular elegiríamos una u otra:

- No dejar borrarlo o modificarlo ( **NO ACTION** ).

Seguramente no es la opción más adecuada para el ejemplo de los empleados y los familiares. Pero pensemos en otro ejemplo, con clientes y facturas. ¿Qué hacemos si se borra un cliente y tenemos facturas de él? Seguramente el más adecuado será no hacer la acción, es decir, no borrar el cliente (sobre todo si las facturas están pendientes de cobrar ...).

- Borrar también los familiares o cambiarlos **en cascada** ( **CASCADE** ).

Seguramente esta es la opción más adecuada para el caso de los familiares, que se eliminan automáticamente.

- Cambiar el valor de la clave externa al valor nulo o un valor predeterminado ( **SET NULL** o **SET DEFAULT** ).

En el ejemplo de los familiares no tiene sentido ya que no nos interesan los familiares de los que no son de la empresa. Pero imaginemos, por ejemplo, un proveedor que nos ha proporcionado unos artículos. Por el hecho de no trabajar ya con el proveedor y quitarlo de la BD no deberíamos eliminar los artículos. Sería suficiente con dar un valor nulo al proveedor de este artículo [\[1\]](#) .

Hay SGBD que incluso permiten acciones distintas para el caso de borrado y de actualización de la clave, como por ejemplo Access.

En Access las restricciones de integridad (claves externas) se definen con las **relaciones** entre tablas. El siguiente video muestra cómo crear la clave externa de Familiar hacia Empleado, y se puede observar como la manera de representarlo es muy parecida a la forma alternativa que estamos utilizando.

---

[\[1\]](#) Observamos, sin embargo, que esta clave externa debería admitir valores nulos. Si no lo permite, mejor un valor por defecto.

### 3.2.6 Restricciones externas

---

A pesar de todas las restricciones anteriores, que normalmente los SGBDR cumplen, hay otros que no se pueden expresar por medio del Modelo Relacional, y por tanto no pueden cumplir directamente los SGBDR. Son las **restricciones externas al esquema relacional**. Estarán normalmente las heredadas de Modelo E / R, y habrá algunas más que sí se podían expresar en el Modelo E / R, pero no en el Modelo Relacional. Las veremos en la pregunta 4.

Los SGBDR más avanzados, más potentes, permitirán un tratamiento a estas restricciones externas, consistentes en ejecutar un procedimiento definido por el usuario después de una actualización. Son los **Disparadores ( triggers )**. Este concepto es muy potente, ya que da una respuesta procedimental donde se puede hacer cualquier cosa.

En Access dos serán las herramientas para hacer esto, las **Macros** y procedimientos en Visual Basic.



## 4. Transformación del M. E / R al M. Relacional

---

A continuación veremos las reglas de transformación del esquema en el Modelo E / R al Modelo Relacional.

De esta manera continuamos el proceso de diseño de una Base de Datos. En el Tema 2 hemos aprendido a hacer el esquema en el Modelo Entidad-Relación. Ahora el traduciremos al Modelo Relacional, y ya se podrá implementar en cualquier SGBD Relacional. Faltaría sólo el proceso de Normalización (tema 4) para acabar de dejar las tablas perfectamente diseñadas. De todos modos las Bases de Datos que diseñamos nosotros, con el proceso descrito anteriormente, tendrán unas tablas muy "normalizadas", siempre que diseñamos bien.

Toda entidad se transformará en una mesa, con todos sus atributos, que se considerarán como simples. Se elige uno (o un conjunto) como clave principal, y lo denotaremos subrayando el. Las entidades débiles las estudiaremos mejor más adelante.

En nuestro ejemplo, como teníamos 4 entidades, nos saldrán de momento 4 mesas:

```
EMPLEAT (dni, nom, adreca, telefon, sou, data_n)
DEPARTAMENT (num_d, nom_d)
PROJECTE (num_p, nom_p)
FAMILIAR (nom, data_n, parentesc)
```

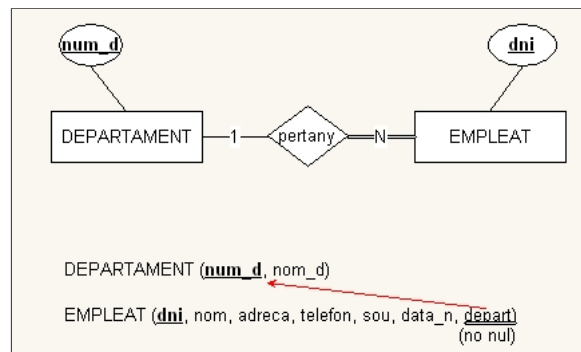
No consideraremos los atributos multivaluados. Los trataremos y solucionaremos en el tema siguiente, el de **Normalización** .

Para cada relación 1: N entre las entidades **A** y **B**, donde **A** es la que participa con grado de cardinalidad 1, y **B** con grado N, se incluye un nuevo campo en **B** (del mismo tipo que la clave principal de **A**) que además será **clave externa** que apuntará a **a**, más concretamente en la clave principal de **a**. En muchas ocasiones el campo nuevo de **B** se le pone el mismo nombre que la clave principal de **A**, pero no es necesario, depende del gusto de cada uno.

También se incluirán en **B** todos los posibles atributos de la relación.

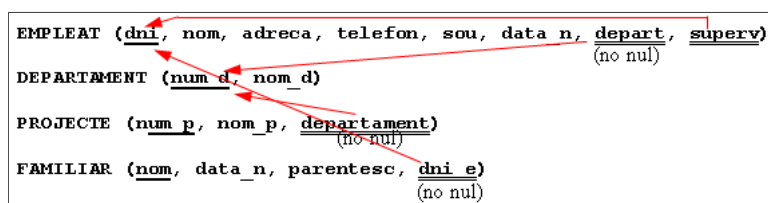
La siguiente animación intenta explicarlo mejor:

Si además la entidad que participa con grado N lo hace de forma **total** (como en la figura de abajo), la clave externa **no puede ser nula** (es decir siempre debe tener un valor).



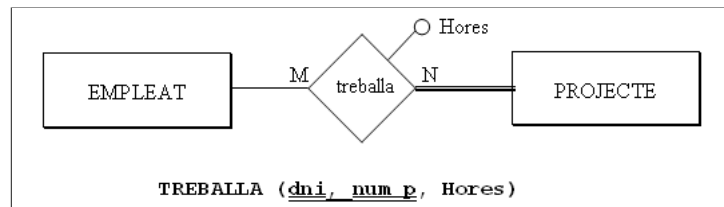
En el ejemplo nuestro:

- Por la relación **Pertenece** (figura de arriba) incluiremos el atributo **departamento** a **Empleado**, que además deberá ser no nulo.
- Por la relación **Controla** incluiremos el atributo **departamento** a **Proyecto** (no nulo).
- Por la relación **Supervisa** incluiremos el atributo **supervisor** a **Empleado** (es reflexiva), pero este sí puede ser nulo. Aunque parezca extraño, un campo puede ser clave externa que apunta a la clave principal de la misma mesa.
- Por la relación **Tiene** entre empleado y familiar, incluiremos en **FAMILIAR** el atributo **dni\_e**, pero como Familiar es débil la veremos mejor un poco más adelante.



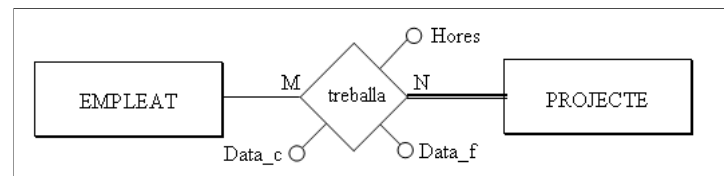
Para cada relación M: N construiremos una nueva tabla donde se incluirán como clave externa las claves principales de las dos entidades, y además su combinación constituirá (o formará parte de) la clave principal. Incluiremos también los posibles atributos de la relación.

En el ejemplo:



Cada vez que nos encontramos con una relación M: N y significar una nueva tabla tendremos que **preguntarnos si es suficiente con la clave principal** formada por las dos claves externas, o si hace falta añadir otro campo. Debemos ser conscientes de que la clave principal no pueda repetirse, que haya 2 filas con la misma clave principal. Así, en el ejemplo, deberíamos hacer la siguiente pregunta: ¿puede un empleado trabajar en el mismo proyecto más de una vez? En este caso la contestación es negativa, y por tanto es suficiente con esta clave principal.

Pero supongamos que la respuesta es que sí puede trabajar más de una vez a lo largo del tiempo. En este caso sería una especie de histórico, donde haría falta, además, saber cuándo empieza y cuando termina de trabajar en un proyecto un determinado trabajador (serían atributos de la relación):



Entonces, como no es suficiente con la clave principal formada por las dos claves externas, incluiremos otro campo en la clave principal. Parece que lo más adecuado sería **Data\_c** (ya no se puede dar el caso de que el mismo trabajador trabaje más de una vez en el mismo proyecto, comenzando el mismo día)

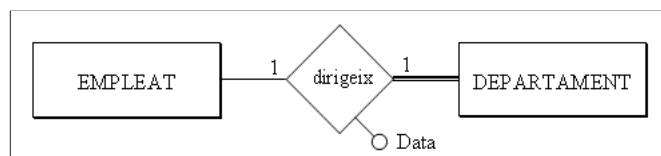
**TREBALLA ( dni, num\_p, data\_c, data\_f, hores )**

Pero por otra parte, las claves principales muy largas no son operativas, y aunque la traducción literal sea como hemos dicho, por motivos prácticos podemos cambiar la clave principal. Consideraremos que el número máximo de campos en la clave principal es de 3. 4 ya son demasiado, y entonces buscaremos otra clave principal (un código de trabajo, por ejemplo). Las claves externas continuarían siéndolo.

No hay una forma única de traducir estas relaciones. Tres serán las posibles traducciones, según la participación total o parcial de las entidades en la relación, y también según lo que nos diga el " *sentido común* ".

- Si de las dos entidades que entran la relación, **A** y **B**, una de ellas y sólo una, participa de forma total, por ejemplo **B**, traduciremos la relación 1: 1 como una **clave externa** en la tabla correspondiente a la entidad que participa de forma total ( **B** ). Podemos obligar también a que este campo que será clave externa sea **no nulo** (ya que todas las ocurrencias de **B** entran en la relación). También podemos hacer que este campo sea **único** (no se podrá repetir, ya que si se pudiera repetir sería una relación 1: N). Además, incluiremos en **B** todos los posibles atributos de la relación.

Por ejemplo, la relación **dirige**, que es 1: 1 entre **EMPLEADO** y **DEPARTAMENTO** :

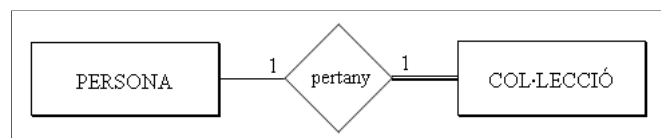


Como la entidad de la derecha participa de forma total, elegiremos **DEPARTAMENTO** :

**DEPARTAMENT** (num\_d, nom\_d, director, Data)  
(no nul, únic)

Lo hacemos de esta manera que todos los departamentos tienen director, pero no todos los empleados son directores. Si pusiéramos la clave externa en la tabla **EMPLEADO** (llamaría por ejemplo **dep\_que\_dirigeix**) muchas veces estaría vacío, ya que relativamente son pocos los empleados que dirigen un departamento.

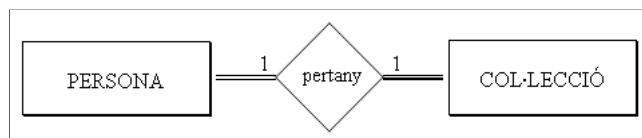
Veamos otro ejemplo de relación 1: 1, el de las mariposas. Teníamos una relación 1: 1 entre **PERSONA** y **COLECCIÓN**



Elegiríamos **COLECCIÓN**, ya que entra de forma plena o total en la relación.

**COL·LECCIÓ**(cod\_col, ... (dades col·lecció)... , dni-prop)  
(no nul, únic)

- Si las dos entidades participan de forma total, se puede considerar todo (las dos entidades y la relación con sus posibles atributos) como una sola mesa. En la práctica esto será bastante extraño, porque ya lo habríamos considerado una sola entidad. Por ejemplo, consideramos que todas las personas que estudiamos tienen una colección:



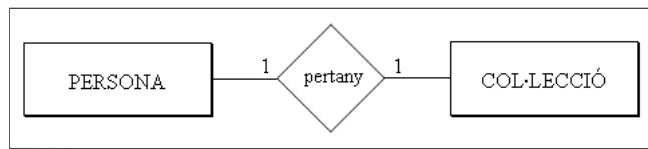
Entonces podríamos considerar una única tabla, que contenga los datos de la persona y de su colección:

**PERSONA**(Dni, ... (dades pers)... , ... (dades col·lecció)... )

De todos modos, puede ser nos interese (para separar claramente los dos tipos de información) dos tablas. Entonces podríamos traducirlo como en el primer punto, con una clave externa (no nula) en una de las dos tablas, y deberíamos elegir la tabla que *menos* se utilice.

- Si las dos entidades participan de forma parcial, como que si ponemos una clave externa en una de las dos, muchas veces tendrá valor nulo, podemos traducir como una nueva tabla que marque la relación, donde habrá una tupla por cada relación entre dos ocurrencias. Incluiríamos en la nueva mesa los posibles atributos de la relación.

En el ejemplo podría ser que las colecciones pertenecen a una persona particular o una institución (no tienen propietario):



quedaría:

```

PERSONA(dni, ... (dades personals) ... )
COL·LECCIÓ(cod col, ... (dades de la col·lecció) ... )
PROPIETARI(dni, cod col)
  
```

Pero como comentábamos al principio, tendremos que aplicar el sentido común, ya que tal vez una de las dos podría participar de forma "casi" total (por ejemplo, casi todas las colecciones son de una persona). Entonces podría ser mejor traducirlo como en el primer caso, poniendo la clave externa en la que participa de forma "casi" total, ya que ésta tendrá relativamente pocos valores nulos, y sería más costoso mantener otra tabla. Evidentemente la clave externa sí podría ser nula, en este caso.

Resumiendo, **una relación 1: 1 casi siempre la traduciremos como una clave externa en la tabla que participa en la relación de forma total o casi total** (o la que previsiblemente tiene más ocurrencias en la relación)

## 4.5 Entidades débiles

---

Las entidades débiles, como que al menos dependen de otra, podremos ser más restrictivos que las otras. Una entidad siempre es débil a través, como mínimo, de una relación que la comunica con la entidad principal. Además participa de forma total (como que no puede existir sin la otra, toda ocurrencia está en la relación).

Por lo tanto, toda entidad débil tendrá una clave externa, que apunta a la principal y será no nula. Pero aún podemos ir más allá:

- **Dependencia en existencia** : haremos que la clave externa **borre y actualice en cascada** , ya que si deja de existir la principal no tiene sentido la débil.
- **Dependencia en identificación** : además de borrar y actualizar en cascada, la clave externa **formará parte de la clave principal** .

Si la relación por la que depende en identificación es 1: N, hará falta otro campo en la clave principal.

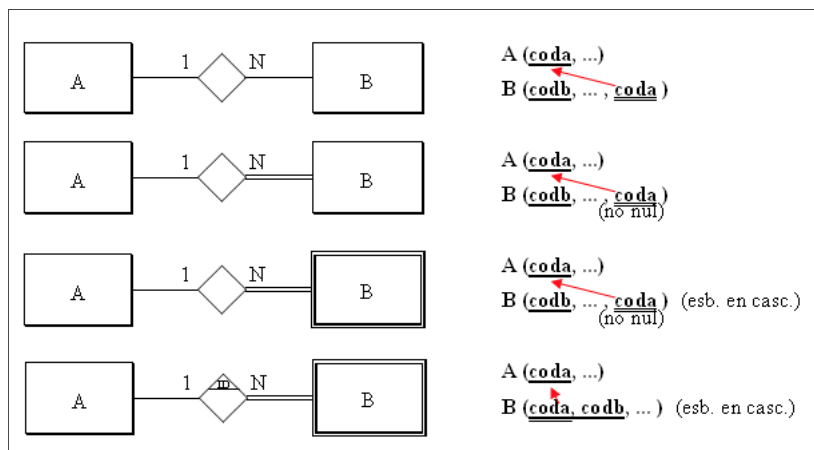
Si es 1: 1, con la clave externa es suficiente como clave principal

En nuestro ejemplo quedará así:

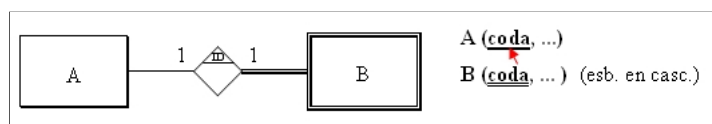
<b>FAMILIAR</b> ( <u><b>dni_e, nom_f</b></u> , data_n, parentesc) (esborrar en cascada)
---

## 4.6 Resumen dependencias

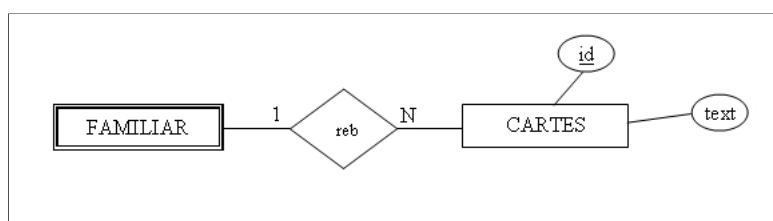
Vamos a hacer un cuadro resumen con distintos grados de dependencia entre dos relaciones y cómo se traduciría al Modelo Relacional:



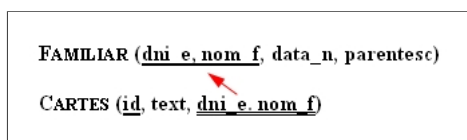
Por otra parte, si nos encontramos una entidad débil que depende en identificación a través de una relación 1: 1, es suficiente con la clave principal de A como clave principal de B



Otra cuestión que puede llevar a confusión es el caso de las claves externas formadas por 2 campos. Nos basaremos en el ejemplo de siempre, en la tabla FAMILIAR, ya que tiene una clave principal formada por 2 campos. Supongamos que hay en el Modelo Entidad-Relación una tabla que depende de ella, como podría ser comunicaciones que se le han hecho (cartas). Las entidades y la relación entre ellas podría ser esta:



Como la relación es de tipo 1: N la traduciremos por una clave externa en CARTAS. Y cuál será la clave externa? Como la tabla FAMILIAR tiene una clave principal formada por 2 campos, tendremos que poner una clave externa formada por 2 campos. Alerta! no serán 2 llaves externas, sino una clave externa formada por 2 campos. Como siempre, representaremos la clave externa con un doble subrayado, que ahora cogerá a los 2 campos.





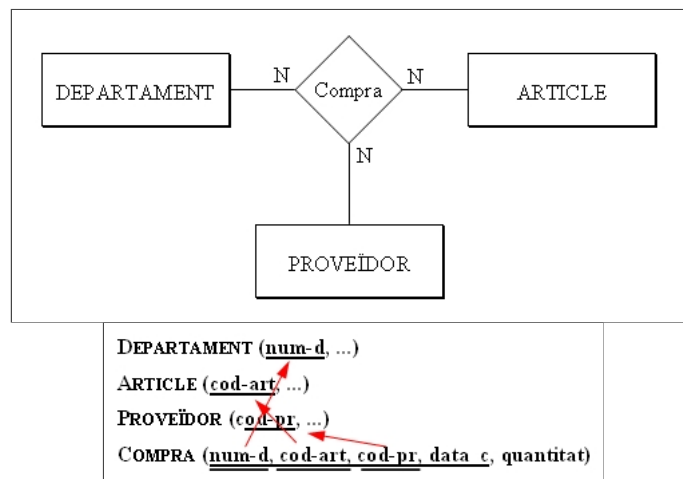
## 4.7 Relaciones ternarias

En una relación ternaria o superior construiremos una nueva tabla, donde incluiremos como claves externas las claves primarias de todas las entidades, y además los atributos de la relación.

Habitualmente, la clave principal de la nueva tabla será la combinación de todas las claves principales de las entidades. Ocasionalmente, si alguna entidad participa con cardinalidad 1, la clave principal de esta entidad no entraría a formar parte de la clave principal de la nueva tabla.

Al igual que en el caso de las relaciones M: N, habrá que preguntarse si es suficiente con la clave primaria generada o si se deberá incluir algún otro campo.

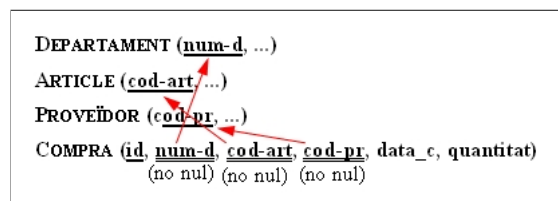
En el ejemplo que pusimos de relación ternaria, suponiendo los atributos de la relación la fecha de compra y la cantidad:



Donde hemos puesto todas las claves externas formando parte de la clave principal, ya que todas entran con cardinalidad N. Pero no teníamos bastante con esta clave principal, ya que el mismo departamento puede comprar el mismo artículo al mismo proveedor más de una vez. Como no teníamos bastante, hemos puesto otro campo.

Sería momento, seguramente, de sustituir la clave principal que está formada por 4 campos, ya que son demasiado.

Pondríamos otra clave principal, pero las claves externas continuarían siéndolo, y además serían no nulas:



Este aspecto, como en el caso de las relaciones 1: 1, también tiene múltiples soluciones. El problema de las especializaciones es que, aunque de forma teórica la solución sea correcta, en la práctica supone muchas mesas y con un cierto grado de mantenimiento entre ellas. Por tanto, y aplicando el sentido común, muchas veces se hace una simplificación, quitando bien las subclases, bien la superclase, como veremos a continuación. Estas son las posibilidades:

- Transformar las especializaciones en tablas con la clave principal heredada y los atributos específicos (como si sustituye la especialización en relaciones 1: 1, con las subclases dependiendo en identificación de la superclase). Estaría también bien añadir un atributo a la superclase para poder saber de qué tipo es. En el ejemplo nos quedaría:

<b>EMPLEAT</b> ( <u>dni</u> , nom_e, adreca, telefon, ..., tipus)
<b>CAP</b> ( <u>dni</u> , opinio)
<b>TREBALLADOR</b> ( <u>dni</u> , hores_extres)

- Simplificar las subclases. Entonces todos los atributos de estas deberían pasar a la superclase. También deberían pasar todas las relaciones que afectan a las subclases, "retocando" las participaciones totales (que ahora ya no lo serían). Ahora será obligatorio tener el campo que distingue el tipo (sino, perderíamos esta información).

<b>EMPLEAT</b> ( <u>dni</u> , nom_e, adreca, telefon, ..., tipus, opinio, hores_extres)
---

- Simplificar la superclase. Entonces todos los atributos de esta pasarían a cada una de las subclases. También pasarían las relaciones en cada una de estas, "retocando" las participaciones totales.

<b>CAP</b> ( <u>dni</u> , nom_e, adreca, telefon, ..., opinio)
<b>TREBALLADOR</b> ( <u>dni</u> , nom_e, adreca, telefon, ..., hores_extres)

## 4.9 Restricciones externas

Ya hemos comentado que restricciones externas son restricciones que no se pueden expresar por medio del modelo de datos. Entonces las expresaremos de palabra.

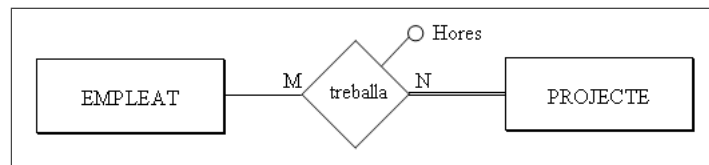
Normalmente, las restricciones externas del Modelo E / R continuarán siéndolo en el Modelo Relacional, porque tampoco se podrán expresar. En nuestro ejemplo teníamos las restricciones externas del Modelo E / R:

**Rex1** : El jefe de un departamento debe ser miembro de éste.

**Rex2** : Un empleado sólo puede trabajar en proyectos coordinados por su departamento.

Esto tampoco se puede expresar con el Modelo Relacional, por lo tanto las mantendremos.

Además, se pueden crear más restricciones externas, porque con el Modelo Relacional no se puede expresar todo lo que se podía expresar con el Modelo E / R. Por ejemplo, en la relación:



ya hemos comentado que dará lugar, además de las tablas de las entidades, a otra mesa:

<b>TREBALLA</b> ( <u>dni</u> , <u>num_p</u> , hores)
--

donde **dni** y **num\_p** son claves externas. Pero la entidad proyecto participa de forma total en la relación, es decir, en todo proyecto debe haber un empleado trabajando. ¿Cómo lo controlamos esto? Pues es una nueva restricción externa que la podríamos formular así:

**RexR3** : Todo proyecto debe tener como mínimo un empleado trabajando.

Las participaciones totales que nos supondrán una restricción externa son:

- En una relación **1: N** , una participación total de la entidad que participa con grado 1.
- En una relación **M: N** , cualquier participación total (si ambas participan de forma total, entonces habrá dos restricciones externas).
- En una relación **1: 1** , depende de la manera de traducirse.

La manera de implementar las restricciones externas será por medio de un TRIGGER, que se active cuando haya una actualización (inserción, modificación o borrado) que afecte a la restricción externa. Por ejemplo, en la restricción **RexR3** los momentos importantes son después de insertar un nuevo proyecto, y antes de eliminar o modificar en la tabla **Trabaja** (por si un proyecto se queda sin gente trabajando en él).

Las acciones a desarrollar podrían ser sacar un aviso, u obligar a insertar al menos una tupla en la tabla **Trabaja** , en el caso de la inserción de un nuevo proyecto; en el caso de modificación o eliminación en **Trabaja** podría impedirse esta actualización.

En nuestro ejemplo tendremos las restricciones externas al Modelo Relacional:

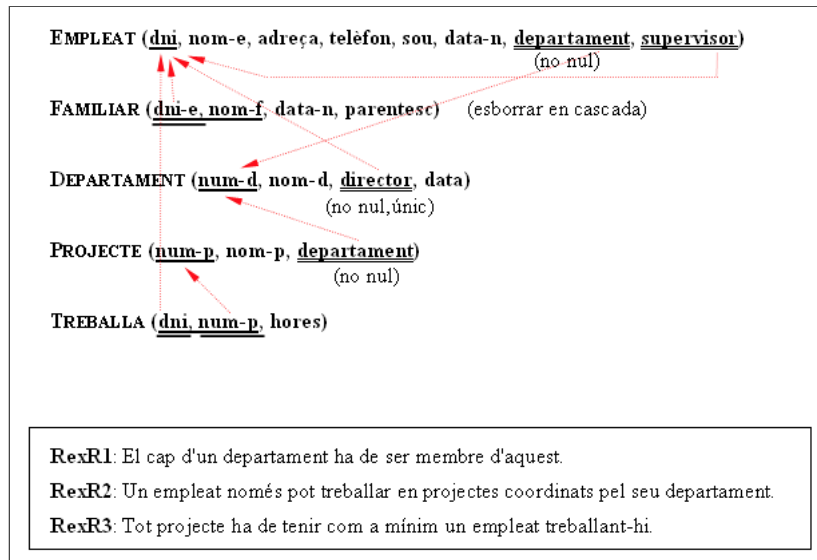
**RexR1** : El jefe de un departamento debe ser miembro de éste.

**RexR2** : Un empleado sólo puede trabajar en proyectos coordinados por su departamento.

**RexR3** : Todo proyecto debe tener como mínimo un empleado trabajando.

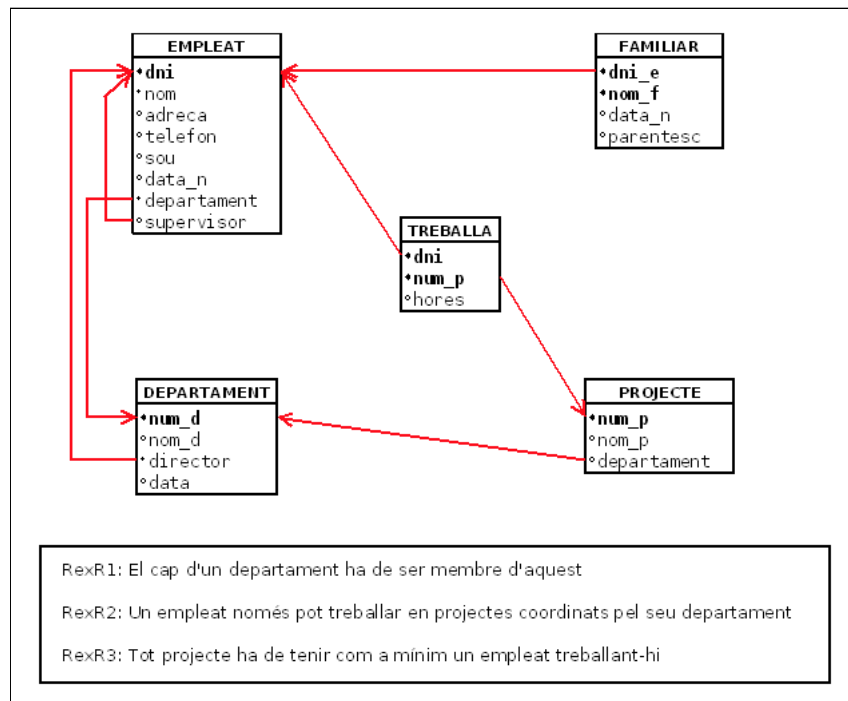
Vamos a ver cómo quedará definitivamente la traducción del ejemplo que estamos arrastrando desde el Tema 2. Daremos 2 versiones, teniendo en cuenta o no la especialización de que el trabajador puede ser jefe o trabajador normal.

Sin tener en cuenta la especialización tendremos esta solución:

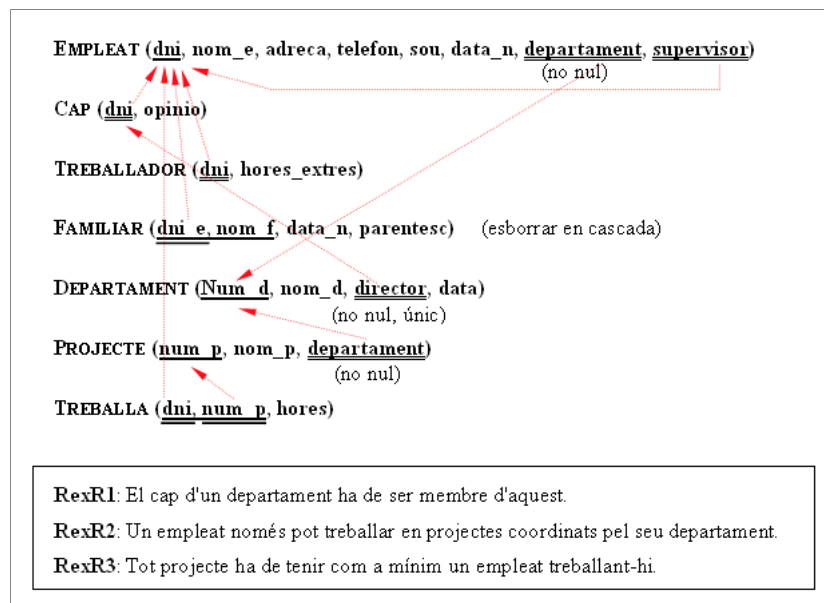


este video explica todo el proceso paso a paso:

Y esta sería la forma alternativa de representarlo:

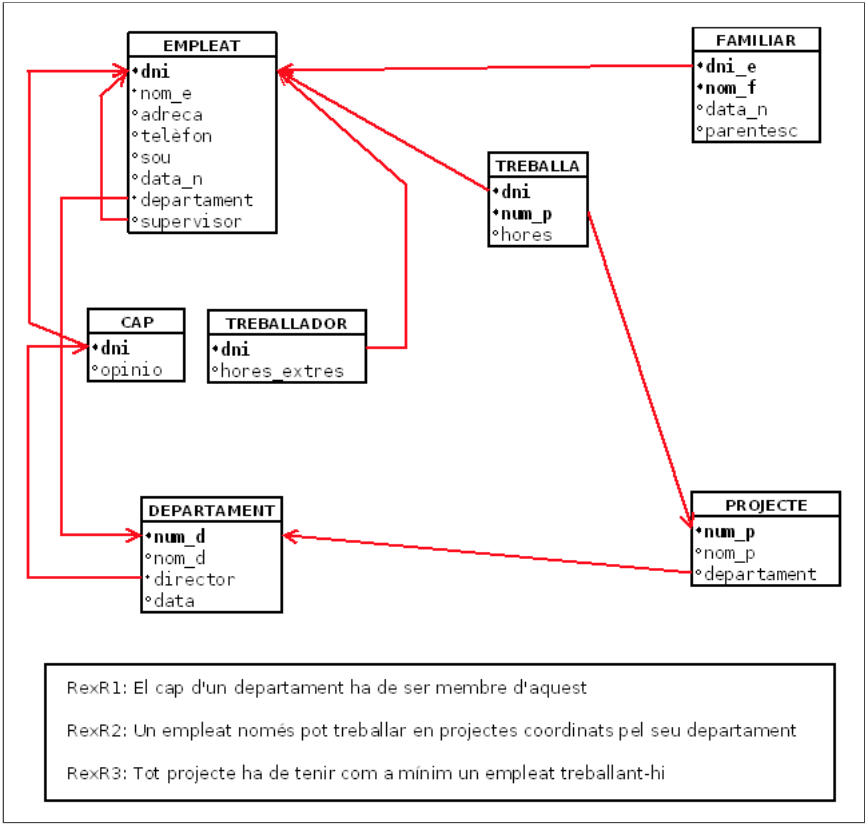


Y teniendo en cuenta la especialización:



que este vídeo explica en los puntos diferentes a la anterior solución

Y aquí tendríamos la representación alternativa:



## 5. Lenguajes relacionales

Hasta el momento hemos diseñado una BD Relacional. Pero si queremos la BD es para consultarla, sacar la información que nos interesa, manipularla. Nos hará falta, por tanto, un lenguaje de manipulación de la BD ( **DML** ). Estos lenguajes pueden estar basados en el Álgebra Relacional o en el Cálculo Relacional.

- Por medio de la **ÁLGEBRA RELACIONAL** haremos operaciones sobre las tablas, combinándolas, seleccionando lo que nos importa, ..., en definitiva manipulando la foto. El resultado será una nueva tabla, que será el resultado final o servirá para hacer otra operación. Las operaciones pueden ser **proyección** (coger algunas columnas de una tabla), **selección** (seleccionar algunas filas de una tabla, las que cumplen una determinada condición), **unión** , **intersección** , **producto cartesiano** , **reunión** ...

El lenguaje **SQL** , que es el estándar de hecho y que veremos en el tema 6, está basado en el Álgebra relacional. Las sentencias son de la forma:

```
SELECT dni, nombre, sueldo
FROM EMPLEADO
WHERE sueldo > 2000
```

donde estamos proyectando sobre los campos dni, nombre y sueldo, y seleccionando las filas que cumplen la condición del final

Vamos a comentar dos de las operaciones antes mencionadas. El **producto cartesiano** de dos tablas consiste en combinar cada una de las filas de una tabla con cada una de las filas de la otra. Así, el producto cartesiano de **Empleado** y **Departamento** sería:

```
SELECT dni, nombre, nom_d
FROM EMPLEADO, DEPARTAMENTO
```

Pero esta operación no parece tener mucho sentido en este caso ¿para qué queremos combinar un empleado con todos los departamentos de la empresa? Parece mucho más lógico combinar cada empleado únicamente con el departamento al que pertenece.

La **reunión** de dos tablas consiste en hacer un producto cartesiano y luego seleccionar las filas que tienen el mismo valor en dos campos determinados (uno de cada tabla).

```
SELECT dni, nombre, nom_d
FROM EMPLEADO, DEPARTAMENTO
WHERE EMPLEADO.departament = DEPARTAMENTO.num_d
```

Es decir, del producto cartesiano seleccionamos sólo las filas en las que coinciden los campos departamento y num\_d, combinando cada empleado con su departamento.

- En el **CÁLCULO RELACIONAL** , se definen variables de tipo **tabla** , se utilizan operadores entre las variables, y también unos cuantificadores ( **para todo** y **existe** ). Va de forma paralela al álgebra de manera que se pueden obtener las mismas cosas con el Álgebra y con el Cálculo.

El **QBE** (Query By Example) se basa en el cálculo relacional, y su particularidad es la sencillez de hacer consultas para los no expertos. Por medio de una plantilla podremos colocar los atributos que queremos visualizar, el orden, los criterios de selección, etc.

Campo:	Dni	Nom	Sou	
Tabla:	Empleat	Empleat	Empleat	
Orden:		Ascendente		
Mostrar:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Criterios:			>2000	
o:				

## ? Llenar los espacios en blanco

Intenta llenar los espacios en blanco de los siguientes enunciados:

El elemento básico del modelo relacional es la  , Que no es más que una  bidimensional. Cada una está formada por filas, también llamadas  Y columnas, también llamadas

## ? elección múltiple

El conjunto de valores posibles que puede tomar una columna se denomina:

- ☐ universo
- ☐ dominio
- ☐ campo
- ☐ ninguna de las anteriores

Noooooooo

muy bien

Noooooooo

Noooooooo

### **solución**

1. Incorrecto ( Realimentación )
2. Opción correcta ( Realimentación )
3. Incorrecto ( Realimentación )
4. Incorrecto ( Realimentación )

## ? Verdadero / falso

Indica si las siguientes afirmaciones son correctas

Los valores de las columnas deben ser multivaluados

pista

- ☐ verdad ☐ mentira

### **mentira**

El orden de las filas no es significativo

pista

- ☐ verdad ☐ mentira

### **verdad**

El orden de las columnas sí es significativo, ya que supone cambiar la estructura

pista

- ☐ verdad ☐ mentira

### **mentira**



## ? elección múltiple

¿Qué afirmación es la correcta?

- ☐ La clave principal sólo puede estar formada por un campo, que no se repetirá ni cogerá valores nulos.
- ☐ La clave principal puede estar formada por un campo o más de un campo. En este último caso algún campo puede tomar el valor nulo, pero no todos a la vez.
- ☐ La clave principal está formada por varias claves candidatas, tomadas utilizando el "sentido común"
- ☐ Ninguna de las anteriores

incorrecto

incorrecto

incorrecto

correcto

### **solución**

1. Incorrecto ( Realimentación )
2. Incorrecto ( Realimentación )
3. Incorrecto ( Realimentación )
4. Opción correcta ( Realimentación )

El valor nulo es equivalente a:

- ☐ el cero
- ☐ el espacio en blanco
- ☐ cualquiera de las anteriores
- ☐ ninguna de las anteriores

incorrecto

incorrecto

incorrecto

Correcto !!

### **solución**

1. Incorrecto ( Realimentación )
2. Incorrecto ( Realimentación )
3. Incorrecto ( Realimentación )
4. Opción correcta ( Realimentación )

El concepto de integridad referencial se aplica a través de la ...

- ☐ clave principal
- ☐ clave candidata
- ☐ clave externa
- ☐ ninguna de las anteriores

incorrecto

incorrecto

Correcto !!

incorrecto

### **solución**

1. Incorrecto ( Realimentación )
2. Incorrecto ( Realimentación )
3. Opción correcta ( Realimentación )
4. Incorrecto ( Realimentación )

## ? Cómo traducir del Modelo E / R al Relacional

Como traducir una entidad?

- ☐ como una nueva tabla
- ☐ como un campo
- ☐ depende de la entidad
- ☐ como una clave externa

Correcto !!

incorrecto

incorrecto

incorrecto

### **solución**

1. Opción correcta ( Realimentación )
2. Incorrecto ( Realimentación )
3. Incorrecto ( Realimentación )
4. Incorrecto ( Realimentación )

Y como traducir una relación 1: N?

- ☐ como una nueva tabla
- ☐ como un campo
- ☐ depende de la relación
- ☐ como una clave externa

incorrecto

incorrecto

incorrecto

Correcto !!

### **solución**

1. Incorrecto ( Realimentación )
2. Incorrecto ( Realimentación )
3. Incorrecto ( Realimentación )
4. Opción correcta ( Realimentación )

Y como traducir una relación M: N

- ☐ como una nueva tabla
- ☐ como un campo
- ☐ depende de la relación
- ☐ como una clave externa

Correcto !!

incorrecto

incorrecto

incorrecto

### **solución**

1. Opción correcta ( Realimentación )
2. Incorrecto ( Realimentación )
3. Incorrecto ( Realimentación )
4. Incorrecto ( Realimentación )

