MODELO RELACIONAL

1.- ORIGEN Y OBJETIVOS

En 1970 E. F. Codd introdujo la teoría matemática de las relaciones en el campo de los SGBD. El Modelo de datos que creó se llama ***Modelo RELACIONAL***, con una base matemática (álgebra relacional y teoría de conjuntos) muy sólida, donde los datos se agrupan en forma de relaciones (tablas) que son estructuras de datos simples y uniformes, que permiten una fácil comprensión.

En el momento en el que surgió, los modelos que se utilizaban eran el jerárquico y el de red (también denominado Codasyl, por el grupo de trabajo que lo estandarizó). Como modelo de datos los superó porque, como dijo el propio Codd, "proporciona un medio para describir las datos con su estructura únicamente, sin tener que suponer ninguna estructura adicional para representarla en la máquina" (es decir, sin punteros ni estructuras de otro tipo).

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

Los objetivos del Modelo Relacional son:

1. **Fidelidad**, para originar esquemas que representen fielmente la información (los objetos y las relaciones entre ellos) que existen en el dominio del problema.
2. **Independencia física**, para que la forma de guardar las datos no influya en su manipulación lógica, y así los usuarios que accedan a estos datos no tengan que modificar sus programas por cambios en el almacenamiento físico
3. **Independencia lógica**, para que las vistas externas no se vean afectadas por cambios en el esquema conceptual de la B.D. Añadir, eliminar o modificar cualquier elemento de la BD no debe repercutir en los programas y/o usuarios que están accediendo a subconjuntos parciales de los mismos (vistas).
4. **Flexibilidad**, para poder ofrecer a cada usuario los datos de la forma más adecuada a su aplicación.
5. **Uniformidad**, las estructuras lógicas de los datos presentan un aspecto simple y uniforme (las tablas), lo que facilita la concepción y manipulación de la BD por parte de los usuarios.
6. **Sencillez,** las características anteriores, unidas a unos lenguajes de usuario sencillos, hacen que el Modelo Relacional sea fácil de entender y de utilizar por el usuario final.

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.

La relación como elemento fundamental del modelo (de ahí su nombre), ***se puede representar en forma de tabla***.

NOMBRE	atributo 1	atributo 2	.....	atributo n	
	XXX	XXX	.....	XXX	→ tupla 1
	XXX	XXX	.....	XXX	→ tupla 2
	.....	.....	.....	.....	.....
	XXX	XXX	.....	XXX	→ tupla m

Las elementos básicos de una relación, basándonos en el ejemplo siguiente son los atributos, dominios y tuplas:

EMPLEADO

Dni	Nombre	Dirección	Teléfono	Sueldo	Fecha_n
18876543	Llopis Bernat, Jaume	C/ Artana, 3	964-213243	1.200,00	7-12-55
18900111	Garrido Vidal, Rosa	C/ Herrero,54	964-253545	1.200,00	25-1-58
18922222	Nebot Aliaga, Carme	C/ Sant Vicent, 5	694-216191	1.000,00	8-6-59
18932165	Folch Mestre, Pilar	C/Palància, 22	964-234567	1.800,00	8-6-60
18933333	Peris Andreu, Joan	C/ Balmes, 3	964-223344	1.200,00	15-3-60
18934567	Sebastià Broch, Ferran	C/ Magallanes,38	964-281706	1.300,00	14-7-62
18944444	Garcia Tomàs, Alícia	C/ Amunt,15	964-205080	1.400,00	10-5-64

Normalmente una relación tiene un ***NOMBRE*** (empleado) aunque a veces no tiene, por ejemplo una tabla que sea el resultado de una consulta ocasional, serán tablas con una existencia temporal.

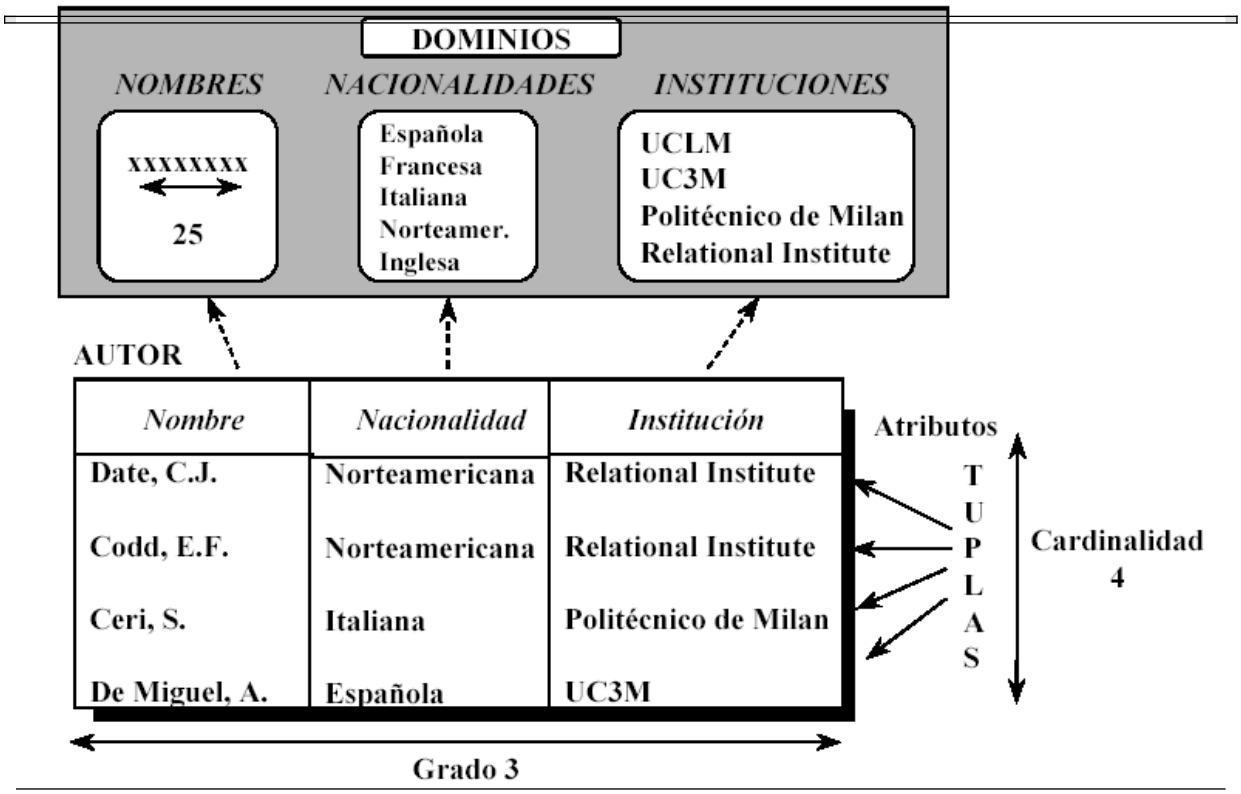
Las filas, donde tenemos la información de cada ocurrencia de la relación, *se denominan* ***TUPLAS*** (en ocasiones por similitud con ficheros, también se llaman REGISTROS).

Las columnas, que serán características que nos interesen de los individuos y que en cada tupla toma un valor, *las llamaremos* ***ATRIBUTOS*** *(o CAMPOS)*. Los atributos representan las propiedades de las relaciones.

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

- ***CARDINALIDAD*** de una relación es el número de tuplas (en el ejemplo 7 filas).
- ***GRADO*** de una relación es el número de atributos (en el ejemplo 5 columnas).

Todos estos elementos representados gráficamente son:



Pero ¡CUIDADO!, una relación no es una tabla. Existen diferencias entre ambas estructuras.

Comparación de la terminología relación, tabla, fichero

RELACIÓN	~	TABLA	~	FICHERO
TUPLA		FILA		REGISTRO
ATRIBUTO		COLUMNA		CAMPO
GRADO		Nº DE COLUMNAS		Nº DE CAMPOS
CARDINALIDAD		Nº DE FILAS		Nº DE REGISTROS

Modelo	SGBD	Sistemas
Relacional	Relacionales	de Ficheros
(teoría)	(implementación)	Clásicos

2.1.- DOMINIOS Y ATRIBUTOS

El Universo del discurso (UD) de una BD Relacional está compuesto por un conjunto de dominios y de relaciones definidas sobre los dominios.

- Un **DOMINIO** es un *conjunto nominado, finito y homogéneo de valores atómicos*, esto es:
  - El dominio se identifica por un nombre.
  - Tiene un número finito de valores.
  - Todos los valores son del mismo tipo.
  - Los valores son atómicos, es decir, no pueden ser a su vez una relación, un grupo repetitivo, etc.
- Cada dominio puede definirse de dos maneras:
  - **Extensión** (dando sus posibles valores):
    Dias de la semana = { lunes, martes, miércoles, jueves, viernes, sábado, domingo }
  - **Intensión** (mediante un tipo de datos):
    Edad = entero
  - A veces se asocia una unidad de medida (kilos, metros, etc.) y/o ciertas restricciones (como un rango de valores).
- Un ATRIBUTO (A) es la interpretación de un determinado dominio en una relación, es decir el "papel" que juega en la misma.
- Un atributo y un dominio pueden llamarse igual, pero:
  - Un atributo siempre está asociado a una relación, mientras que un dominio tiene existencia propia con independencia de las relaciones.
  - Un atributo representa una propiedad de una relación.
  - Un atributo toma valores de un dominio.
  - Varios atributos distintos (de la misma o diferentes relaciones) pueden tomar sus valores del mismo dominio.

ESQUEMA DE RELACIÓN (INTENSIÓN):

AUTOR (*Nombre*: ***Nombres***, *Nacionalidad*: ***Nacionalidades***, *Institución*: ***Instituciones***)

RELACIÓN (EXTENSIÓN, ESTADO* u *OCURRENCIA):

AUTOR

Nombre	*Nacionalidad*	*Institución*
Date, C.J.	**Norteamericana**	**Relational Institute**
De Miguel, A.	**Española**	**UC3M**
Ceri, S.	**Italiana**	**Politécnico de Milan**

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

EL ESQUEMA de la relación constituye la *definición de la relación*, es decir, los atributos que tiene, a que dominios pertenecen y las restricciones que podremos definir. Normalmente no cambiará a lo largo del tiempo.

EL ESTADO de la relación refleja 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.

2.2.- CLAVES

- **CLAVE CANDIDATA:** es un *atributo, o conjunto de atributos, que identifica unívoca y mínimamente cada tupla* (registro) de la relación.
  - De la propia definición de relación se deriva que **siempre existe, al menos, una clave candidata** (*al ser una relación un conjunto y no existir dos tuplas iguales, el conjunto de todos los atributos siempre tiene que identificar unívocamente a cada tupla*).
  - La propiedad de minimalidad implica que no se incluye ningún atributo innecesario.

- Una relación puede tener *más de una clave candidata*. En este caso se debe distinguir entre:
  - **Clave primaria** (Primary Key):
    - Es la clave candidata que el usuario escoge para identificar las tuplas de la relación.

  - Cuando sólo existe una clave candidata, ésta es la clave primaria (siempre existe clave primaria).
- **Claves alternativas** (Alternative Key):
  - Son las claves candidatas que no han sido escogidas como clave primaria.

En el ejemplo de la relación EMPLEADOS podrían ser claves candidatas Dni y Nombre (si suponemos que no se repiten los nombres).

- Se denomina **CLAVE AJENA** de una relación R2 a un conjunto no vacío de atributos cuyos valores han de coincidir con los valores de la clave candidata de una relación R1.
  - R1 y R2 pueden ser la misma relación.
  - La clave ajena y la correspondiente clave candidata han de estar definidas sobre el mismo dominio.

2.3.- CLASES DE RELACIONES

Podemos encontrarnos en un esquema relacional los siguientes tipos de relaciones:

- **Permanentes**: su definición (esquema) permanece en la base de datos, borrándose solamente mediante una acción explícita del usuario.

- **Vistas** (*View*): son relaciones derivadas que se definen dando un nombre a una expresión de consulta. Se podría decir que son relaciones virtuales, en el sentido de que no tienen datos almacenados, sino que lo único que se almacena es su definición en términos de otras relaciones con nombre.

- **Temporales**: desaparecen de la BD en un cierto momento sin necesidad de una acción de borrado específica del usuario: por ejemplo al terminar una sesión o una transacción.

3.- RESTRICCIONES

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 denominadas restricciones semánticas) en las cuales es el usuario quien prohibe, porque el Modelo se lo permite, determinadas circunstancias.

3.1.- RESTRICCIONES INHERENTES

Como hemos dicho son las que impone el propio modelo. Algunas son características que han de cumplir las relaciones. Por 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 ha de ser simple, no divisible. Por 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 tomar posteriormente el nombre de pila por un lado y los

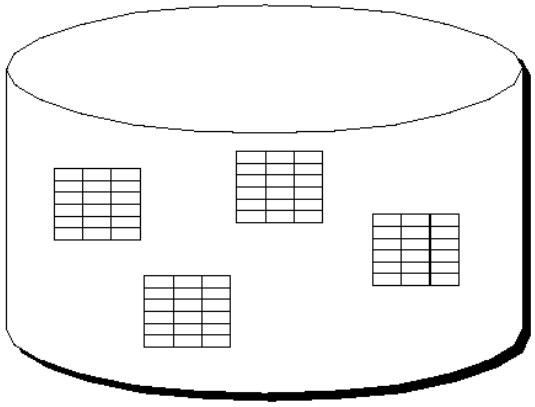
apellidos por otra; si lo quisiera hacer, se habrían de definir los atributos simples Apellido1, Apellido2 y Nombre).

- **Tuplas distintas**: no pueden haber dos tuplas iguales. Esto es una diferencia con respecto a las tablas matemáticas donde sí que se pueden duplicar filas.

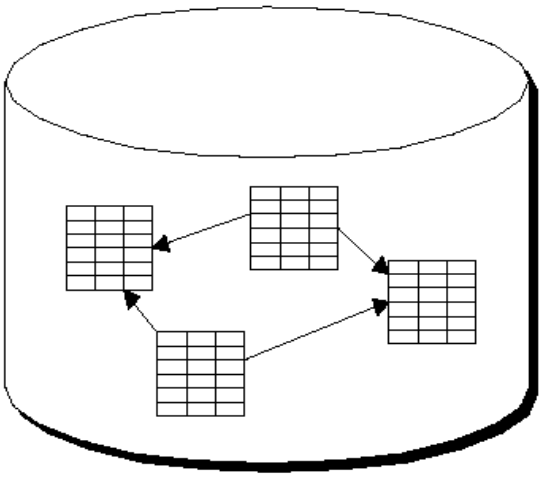
- **Clave Principal**: ha de existir una clave principal o primaria que identificará de forma unívoca las tuplas. Por tanto, la clave principal no podrá tomar valores nulos y tampoco podrá repetirse. Puede estar formada por un atributo o por más de uno.

El orden de las tuplas no es significativo.
El orden de los atributos no es significativo.

Una BD relacional es una colección de relaciones (o tablas en términos de implementación). Pero ... entonces, ¿Representa la figura una BD relacional?



- Respuesta: NO, porque una BD (relacional o de otro tipo) es una colección de datos interrelacionados.
  ⇒ Necesitamos asociar unas relaciones con otras.



Por último veamos que no se deben confundir los conceptos de tabla y de relación, puesto que:

- Una tabla es una forma de representar una relación (una estructura de datos).
- Una tabla no tiene las restricciones inherentes de una relación - como conjunto -:
  - Puede haber dos filas iguales.
  - Las filas están ordenadas en el orden de grabación física por defecto o según el valor de la clave primaria.
  - Los atributos tienen un orden según se han definido en la tabla.
  - En cada celda de una tabla puede haber uno o varios valores.

3.2.- RESTRICCIONES DE USUARIO

También llamadas **restricciones semánticas**, serán *condiciones que podremos establecer para que el esquema de la BD. refleje lo mejor posible la realidad que representa*.

- **RESTRICCIÓN DE DOMINIO**: El valor de un atributo ha de ser un valor atómico del dominio. Definiendo claramente el dominio nos aseguramos (dentro de lo posible) que el atributo no pueda tomar valores incorrectos.

  Por una parte, el dominio será de un tipo determinado, escogido de una gama suficientemente extensa: entero corto, entero, entero largo, real, doble precisión, carácter, cadena de caracteres (texto), fecha, hora, etc. Así, por ejemplo, definiendo el Dni como entero largo impediremos que por error pueda tomar el valor 18934R57, o que la fecha de nacimiento, con dominio de tipo fecha, sea 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).

  Por ejemplo, Empleado podría quedar:

  Empleado (Dni: entero(8); Nombre: carácter(30); Dirección: carácter(30); Teléfono: entero(9); Sueldo: real(5,2); Fecha_n: fecha).

- **RESTRICCIÓN DE CLAVE**: Permite declarar un atributo o un conjunto de atributos como CLAVE PRINCIPAL o PRIMARIA (*Primary Key*).

  Vimos que la obligación de declarar una clave principal era una restricción inherente. Lo que es una restricción de usuario es la elección de la clave principal, y la consecuencia de que no podrá tomar valores nulos ni repetidos.

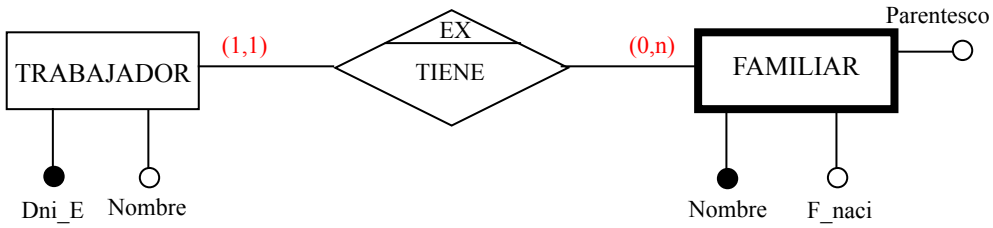
  Estas características de no permitir valores nulos ni repetirse en la relación también las podrán tener otros atributos que no sean clave de la relación:

Los tipos de *restricciones semánticas* permitidos en el modelo Relacional (incorporados a SQL92) son:
- **Clave primaria** (PRIMARY KEY),
- **Unicidad** (UNIQUE),
- **Obligatoriedad** (NOT NULL),
- **Integridad referencial** (FOREIGN KEY),

Podremos hacer que un atributo cumpla más de una. En el ejemplo de la relación EMPLEADOS el nombre podríamos hacerlo NOT NULL (y UNIQUE si no se debe permitir que haya dos empleados con el propio Nombre).

En Access:
NOT NULL ----------> Requerido
UNIQUE ----------> Indexado sin duplicados

Integridad referencial: Para poder explicarla nos apoyaremos en un ejemplo.



que se podría traducir en las siguientes tablas:

TRABAJADOR

Dni_E	**Nombre**
18876543	Lopis Bernat, Jaime
18900111	Garrido Vidal, Rosa
18922222	Ebot Aliaga, Carmen
18932165	Folch Mestre, Pilar
18933333	Peris Andreu, Juan
18934567	Sebastià Broch, Fernando
18944444	Garcia Tomás, Alicia

FAMILIAR

Dni_emp	**Nombre**	**Fecha_naci**	**Parentesco**
18876543	Jaume Llopis Doménech	01/06/85	Hijo
18900111	Felipe Gomis Pitarch	31/03/57	Cónyuge
18932165	Josep Serra González	09/05/60	Cónyuge
18932165	Xavier Serra Folch	05/04/90	Hijo
18933333	Silvia Peris Prades	17/07/95	Hija
18933333	Silvia Prades Arnau	15/03/62	Cónyuge
18944444	Andreu Garcia Torró	15/09/22	Padre
18944444	Laura Almela Garcia	05/12/92	Hija
18944444	Lluís Almela Garcia	10/05/94	Hijo
18944444	Marc Almela Tomeu	25/03/64	Cónyuge

Si en una R2 (FAMILIAR) tenemos un atributo (Dni_emp) *que es clave ajena de otra tabla* R1 (EMPLEADO), todo valor de ese atributo ha de concordar con un valor de la clave de R1 (no he de poder poner en familiar un Dni_emp que no lo tenga ningún empleado de la empresa). El atributo en R2 es, por tanto, una CLAVE EXTERNA o CLAVE AJENA.

Ha de ser imposible poner en FAMILIAR el Dni 18754321, porque no está en la relación EMPLEADO, 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, este ha de ser de la empresa:

EMPLEADO

Dni_E	**Nombre**	**Dirección**	**Teléfono**	**Sueldo**	**Fecha_n**	**Supervisor**
18876543	Lopis Bernat, Jaime	C/ Aitana, 3	964 213243	1.200	7/12/55	18933333
18900111	Garrido Vidal, Rosa	C/herrero, 54	964 253545	1.200	25/1/58	18932165
18922222	Ebot Aliaga, Carmen	C/ Oltà, 21	964 216191	1.000	8/6/59	18944444
18932165	Folch Mestre, Pilar	C/Palancia, 22	964 234567	1.600	8/6/60	0
18933333	Peris Andreu, Juan	C/ Oeste, 14	964 223344	1.200	15/3/60	18944444
18934567	Sebastià Broch, Pedro	C/ Oasis, 11	964 281706	1.500	14/7/62	0
18944444	Garcia Tomás, Alicia	C/ Pez, 3	964 245566	1.700	28/12/65	18944444

Supervisor es una clave externa, pero de la misma tabla. No todos los SGBD permiten una clave externa reflexiva (Access, por ejemplo, no permite este tipo de relaciones).

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

EMPLEADO (Dni, Nombre, Dirección, Teléfono, Sueldo, Fecha_n, Supervisor)
Caj.: Supervisor ⟶ EMPLEADO

Para el esquema E-R de TRABAJADOR y FAMILIAR con la relación TIENE quedaría:

TRABAJADOR (Dni_E, Nombre)

FAMILIAR (Dni_emp, Nombre, Fecha_naci, Parentesco)
Caj.: Dni_emp ⟶ TRABAJADOR

Esto nos impedirá que introduzcamos valores no correctos, no existentes.

¿Pero que 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:

1. No permitir el borrado o la modificación (NO ACTION).
2. Borrar también los familiares que tenga asociados o cambiar los datos en cascada (CASCADE).
3. Cambiar el valor de la clave externa al valor nulo (SET NULL) o un valor predeterminado.

En el ejemplo de los familiares se borrarían todos los familiares de un empleado cuando este deje de serlo, ya que no nos interesan los familiares de los que no son empleados 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 habríamos de eliminar los artículos. Sería suficiente con dar un valor nulo al proveedor de este artículo.

Existen SGBD que hasta permiten acciones distintas para el caso de borrado y de actualización de la clave.

- **OTRAS RESTRICCIONES**: Los SGBDR más avanzados, más potentes, permiten otras restricciones consistentes en comprobar una determinada condición después (o antes) de una actualización. Tendremos dos tipos de actuación, según la acción que se realiza en caso que la condición sea cierta:

  - **Verificación** (CHECK): Comprueba, en toda operación de actualización, si el predicado es cierto o falso y si el predicado (la condición) no se cumple, se rechaza la operación. La restricción de verificación se define sobre un único elemento (dentro de un CREATE TABLE) y puede o no tener nombre. Sirve muy bien para definir muy claramente un dominio, entre otras posibilidades.

    Por ejemplo:

    CHECK Sueldo > 0
    CHECK (Año(Fecha_n) < Año(hoy)) and ((Año(hoy)- Año(Fecha_n)) < 65)

  - **Aserción** (ASSERTION): Actúa de forma idéntica a la anterior, pero se diferencia de ella en que puede afectar a varios elementos (por ejemplo a dos tablas distintas). Por tanto, su definición no va unida a la de un determinado elemento del esquema y siempre ha de tener un nombre.

  - **Disparador** (TRIGGER): Restricción en la que el usuario puede especificar libremente la respuesta (acción) ante una determinada condición, si se cumple la condición se ejecuta un procedimiento definido por el usuario. Este concepto es muy potente, ya que da una respuesta procedimental donde se puede hacer cualquier cosa.

En resumen, podemos considerar que una B.D. Relacional es un conjunto de tablas sobre las que se han definido una serie de restricciones que garanticen la coherencia de la información almacenada. Pero uno de los objetivos de las BD. era eliminar la redundancia. Este aspecto, que aún no lo hemos tratado, lo solucionaremos mediante el proceso de NORMALIZACIÓN.

Ejemplo:

Cod_Art	**Descripción**	**Cantidad**	**Cod_prov**	**Nom_proveedor**	**Dirección**	**Telefono**
01	Disquete	5	32	Distri. Garcia	C/Ample, 32	964 224466
02	Cartucho toner	2	32	Distri. Garcia	C/Ample, 32	964 224466
03	CD-ROM	10	48	Inf. Ascencior	C/ Corrents, 34	964 203040
04	Papel	2	32	Distri. Garcia	C/Ample, 32	964 224466

En este ejemplo hay mucha redundancia, con los inconvenientes de que se ocupa mucho espacio con los datos repetidos, y si se ha de actualizar alguna información (por ejemplo la de un proveedor), se habrá de actualizar en muchos sitios.

El proceso para conseguir una BD relacional normalizada será:

1. **Realizar primero el esquema mediante el Modelo E/R.**
2. **A partir de éste construir el esquema relacional.**
3. **Finalmente normalizaremos el esquema relacional obtenido.**

4.- TRANSFORMACIÓN DEL MODELO E/R AL RELACIONAL.

A continuación veremos las reglas de transformación del esquema conceptual de datos (Modelo E/R) al Modelo Relacional.

Las tres reglas básicas para convertir un esquema en el modelo E/R al relacional son las siguientes:

1. Todo tipo de entidad se convierte en una relación.
2. Todo tipo de relación N:M se transforma en una relación (tabla).
3. Para todo tipo de relación 1:N se realiza lo que se denomina *propagación de clave* (regla general), o bien se crea una nueva relación.

Debido a que el modelo relacional no distingue entre entidades y relaciones, ambos conceptos deben representarse mediante relaciones (tablas), tendremos una pérdida de semántica con respecto al esquema E/R, ya que las relaciones N:M no se distinguen de las entidades y las 1:N se representan mediante una propagación de clave, desapareciendo incluso el nombre de la relación.

4.1.- ENTIDADES.

- **Entidades Normales**: toda entidad normal se transformará en una tabla, con todos sus atributos, que se consideraran como simples. La tabla se llamará igual que el tipo de entidad de donde proviene. En este caso la transformación es directa, y no hay pérdida de semántica.

  Cada atributo de una entidad se transforma en una columna de la relación a la que ha dado lugar la entidad.

  El (o los) atributo(s) que son identificadores principales de la entidad pasan a ser la clave primaria de la relación, y lo denotaremos subrayándolo(s).

  EMPLEADO (<u>Dni</u>, Nombre, Dirección, Teléfono, Sueldo, Fecha-n)
  DEPARTAMENTO (<u>Num-d</u>, Nom-d)
  PROYECTO (<u>Num-p</u>, Nom-p)

- **Entidades débiles**: toda entidad débil S que depende de otra T, se transformará en una tabla con todos sus atributos, y además se añade el o los atributos de la clave principal de T. este atributo será clave externa de S (ya que el valor que pueda tomar ha de coincidir con algún valor de la otra), y la denotaremos con un doble subrayado (en estas páginas aparecerá de forma explícita debajo del esquema de la relación), dado que la dependencia como mínimo es de existencia, la acción en caso de borrar una ocurrencia de T, o modificar su clave, ha de ser en cascada (si borramos un empleado, borraremos todos sus familiares). Si además la dependencia es de identificación, la clave externa formará parte de la clave principal.

  FAMILIAR (Dni-E, <u>Nom-F</u>, Fecha-n, Parentesco)
  clave ajena (Caj.): Dni-e ⟶ EMPLEADO

4.2.- RELACIONES

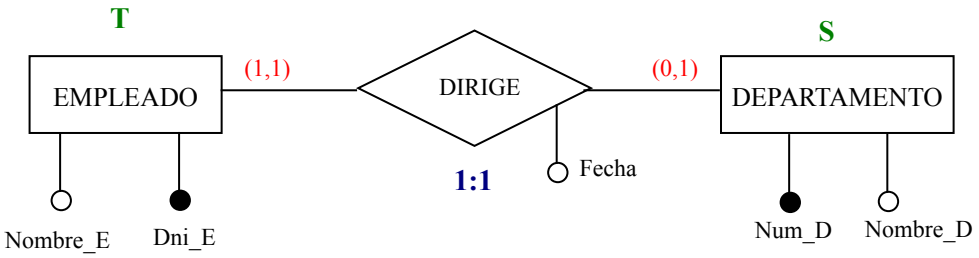
4.2.1.- RELACIONES 1:1

Una relación 1:1 es un caso particular de una M:N o, también, de una 1:N, por lo que no hay regla fija para la transformación de este tipo de relación al modelo relacional estándar.

Nos podemos encontrar con tres situaciones distintas:

A) La entidad T participa con cardinalidad (0,1) y la entidad S participa con (1,1)

Si una de las entidades que participa en la relación posee cardinalidades (0,1), mientras que en la otra entidad son (1,1), conviene propagar la clave de la entidad con cardinalidades (1,1) a la tabla resultante de la entidad con cardinalidades (0,1). Es recomendable escoger la entidad que participe de forma total, es decir S, ya que todas sus ocurrencias participan en la relación, y por tanto la cardinalidad mínima y máxima es (1,1) (por cada ocurrencia de S hay como mínimo y como máximo una de T).



Por tanto escogeremos DEPARTAMENTO:

DEPARTAMENTO (Num-d, Nombre_D, Director, Fecha)
Caj.: Director ⟶ EMPLEADO

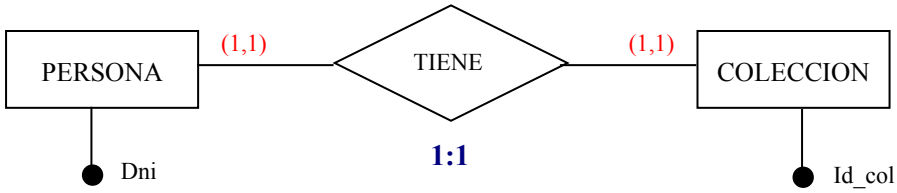
EMPLEADO (Dni_E, Nombre_E)

Observemos que el atributo Director de la tabla DEPARTAMENTO *debe tener el mismo dominio* que el atributo Dni_E para poder ser clave ajena a la tabla EMPLEADO.

Es mejor escoger la de participación total porque todos los departamentos tienen director, pero no todos los empleados son directores. Si incluyéramos en Empleado (Dni, ..., Dep-que-dirige) existirían muchos empleados con el atributo Dep-que-dirige vacío (valor nulo).

B) Las entidades T y S participan con cardinalidad (1,1)

En el caso de que ambas entidades presenten cardinalidades (1,1), se puede propagar la clave de cualquiera de ellas a la tabla resultante de la otra, teniendo en cuenta para esta elección, los accesos más frecuentes y prioritarios a los datos de las tablas.



El esquema resultante, si los accesos a los datos de personas van a ser más frecuentes, sería:

PERSONA (Dni, ..., Id_col)
Caj.: Id_col ⟶ COLECCIÓN

COLECCION (Id_col, ...)

Hay ocasiones en que se podría considerar el conjunto (las dos entidades y la relación) como una sola tabla, pero sólo cuando las dos entidades participen de forma total. En la práctica eso no será frecuente porque, en general, ya lo habríamos considerado una sola entidad. Por ejemplo, en la figura anterior, consideremos un conjunto de personas interesadas en el estudio de las mariposas y supongamos que todas ellas tienen una colección.

En ese caso podríamos considerar una única tabla, que contenga las datos de la persona y de su colección:

PERSONA (Dni, ...(datos personales)... , ...(atributos de la colección)...)

C) Las entidades T y S que se asocian participan con cardinalidad (0,1)

En estas situaciones, cuando las dos participen de forma parcial, puede ser conveniente transformar la relación 1:1 en una relación (tabla).

La relación MATRIMONIO entre los tipos de entidad HOMBRE y MUJER, se transformará en una relación (tabla), evitando así los valores nulos que aparecerían en caso de propagar la clave de MUJER a la tabla HOMBRE o viceversa, ya que como reflejan las cardinalidades no todos los hombre ni todas las mujeres se encuentran casados.

Quedaría:

HOMBRE (Dni_H, ...)

MUJER (Dni_M, ...)

MATRIMONIO (Dni_H, Dni_M)
Caj.: Dni_H ⟶ HOMBRE
Caj.: Dni_M ⟶ MUJER

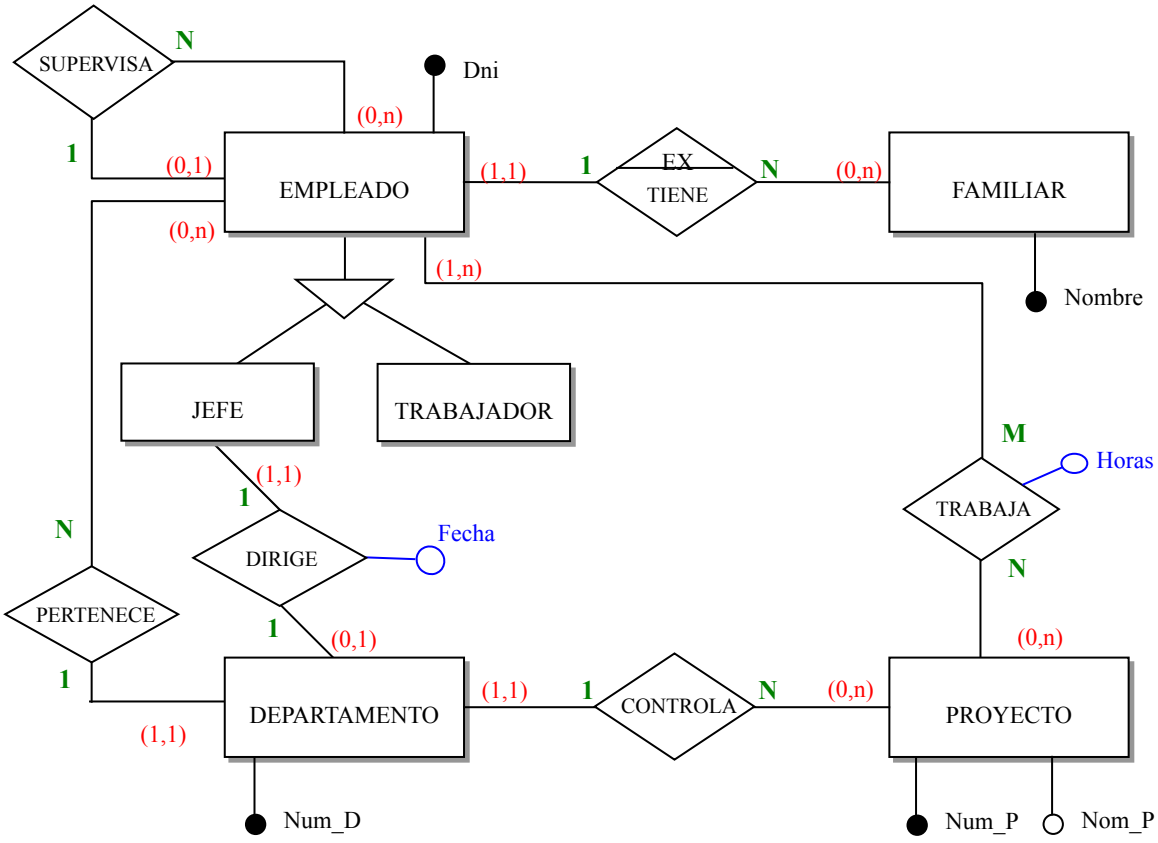
En MATRIMONIO sólo figurarían los Dni_H de los hombres que se encuentran casados y los Dni_M de las mujeres casadas.

4.2.2.- RELACIONES 1:N

Existen dos soluciones para la transformación de una relación 1:N, donde no intervengan entidades débiles.

A. **Propagar los atributos identificadores principales** del tipo de entidad que tiene de cardinalidad máxima 1 a la que tiene N, es decir en el sentido de la flecha, desapareciendo el nombre de la relación, con lo cual se pierde semántica (ésta es la regla habitual).

En el ejemplo de los trabajadores:



- Por la relación PERTENECE incluiremos el atributo Departamento en EMPLEADO.
- Por la relación CONTROLA incluiremos el atributo Departamento en PROYECTO.
- Por la relación SUPERVISA incluiremos el atributo Supervisor en EMPLEADO (es reflexiva).

EMPLEADO (Dni, Nom_E, Domicilio, Teléfono, Sueldo, Fecha_Naci, Departamento, Supervisor)
Caj.: Departamento ⟶ DEPARTAMENTO
Caj.: Supervisor ⟶ EMPLEADO

PROYECTO (Num_P, Nom_P, Departamento)
Caj.: Departamento ⟶ DEPARTAMENTO

El atributo Departamento debe tener el mismo Dominio que el atributo Num_D de la relación DEPARTAMENTO a la que es clave ajena

B. **Transformarla en una relación**, cuyos atributos serán los atributos identificadores principales de las entidades relacionadas, mas los atributos que tenga la relación. La clave primaria de la relación creada estará formada SOLO por los atributos procedentes de la entidad con cardinalidad N. El (los) atributo(s) que procedan de la entidad con cardinalidad 1 serán clave ajena a la relación que

se derive de la entidad con cardinalidad 1. Los casos en los que puede ser apropiado transformar la relación en una tabla son los siguientes:

I. Cuando el número de ejemplares relacionados de la entidad que propaga su clave es muy pequeño y, por tanto, existirían muchos valores nulos en la clave propagada.
II. Cuando la relación tiene atributos propios y no deseamos propagarlos (a fin de conservar la semántica).

4.2.3.- RELACIONES M:N

Por cada relación M:N construiremos una nueva tabla donde la clave primaria será la concatenación de los atributos identificadores principales de las entidades que asocia, que además serán clave ajena a cada entidad asociada. Los atributos de la relación serán atributos normales de la relación (tabla) creada.

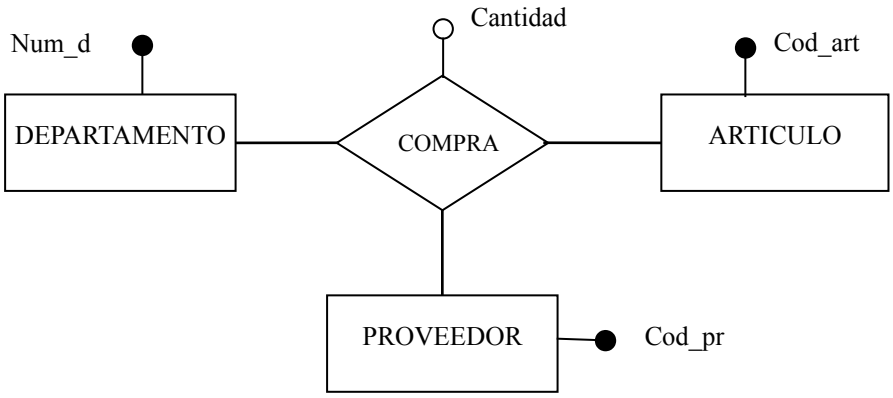
TRABAJA (Dni, Num-p, Horas)
Caj.: Dni ⟶ EMPLEADO
Num-p ⟶ PROYECTO

4.2.4.- 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 (1,1), la clave principal podría ser la clave principal de esta entidad.

El ejemplo de relación ternaria discutido en el tema anterior, origina:



DEPARTAMENTO (Num-d, ...)
ARTÍCULO (Cod-art, ...)
PROVEEDOR (Cod-pr, ...)
COMPRA (Num-d, Cod-Art, Cod-pr, cantidad)
Caj.: Num-d ⟶ DEPARTAMENTO
Caj.: Cod-Art ⟶ ARTICULO
Caj.: Cod-pr ⟶ PROVEEDOR

Aplicando estos criterios al esquema relacional resultante del ejemplo objeto de estudio, obtenemos las siguientes relaciones:

EMPLEADO (Dni, Nom_E, Dirección, Teléfono, Sueldo, Fecha_Naci, Departamento, Supervisor)
Caj.: Departamento ⟶ DEPARTAMENTO
Caj.: Supervisor ⟶ EMPLEADO

FAMILIAR (Nombre, Fecha_N, Parentesco, Dni_E)
Caj.: Dni_E ⟶ EMPLEADO

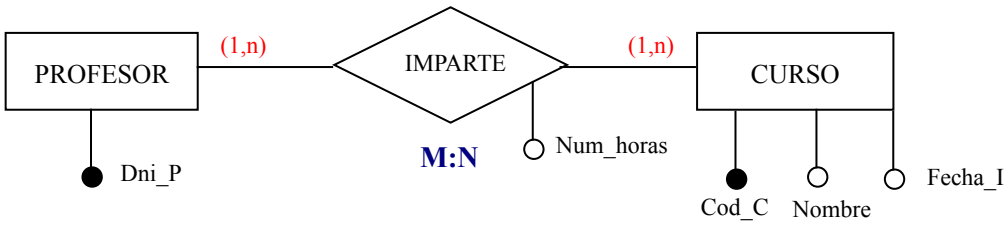
DEPARTAMENTO (Num_D, Nom_D, Director, Fecha)
Caj.: Director ⟶ EMPLEADO

PROYECTO (Num_P, Nom_P, Departamento)
Caj.: Departamento ⟶ DEPARTAMENTO

TRABAJA (Dni, Num_P, Horas)
Caj.: Dni ⟶ EMPLEADO
Caj.: Num_P ⟶ PROYECTO

4.2.5.- Atributos de las relaciones

Si la relación (E-R) se transforma en una relación (Relacional), todos sus atributos pasan a ser columnas de la relación (Relacional).



La relación IMPARTE entre PROFESOR y CURSO tiene un atributo Num_horas que pasa a ser una columna de la tabla que se crea a partir de ella. La transformación es directa y no hay pérdida de semántica.

PROFESOR (Dni_P, ...)

CURSO (Cod_C, Nombre, Fecha_I)

IMPARTE (Dni_P, Cod_C, Num_horas)
Caj.: Dni_P ⟶ PROFESOR
Cod_C ⟶ CURSO

En caso de que la relación se transforme mediante propagación de clave, sus atributos migran junto a la clave a la relación que corresponda, aunque ya hemos advertido que en este caso puede ser preferible crear una nueva relación para representar las relaciones que tienen atributos.

4.3.- REGLAS PARA LAS EXTENSIONES DEL MODELO E/R

4.3.1.- Especializaciones y Generalizaciones

En lo que respecta los tipos y subtipos, no son objetos que se puedan representar explícitamente en el modelo relacional. Ante un tipo de entidad y sus subtipos caben varias soluciones de transformación al modelo relacional, con la consiguiente pérdida de semántica dependiendo de la estrategia elegida. Tendremos tres opciones.

A. Englobar todos los atributos de la entidad y sus subtipos en una sola relación. En general, adoptaremos esta solución cuando los subtipos se diferencien en muy pocos atributos y las relaciones que los asocian con el resto de entidades del esquema sean las mismas para todos (o casi todos) los subtipos.
B. Crear una relación para el supertipo y tantas relaciones como subtipos haya, con sus atributos correspondientes: Ésta es la solución adecuada cuando existen muchos atributos distintos entre los subtipos y se quieren mantener de todas maneras los atributos comunes a todos ellos en una relación.
C. Considerar relaciones distintas para cada subtipo, que contengan, además de los atributos propios, los atributos comunes. Se elegiría esta opción cuando se dieran las mismas condiciones que en el caso anterior – muchos atributos distintos – y los accesos realizados sobre los datos de los distintos subtipos siempre afectan a atributos comunes.

Desde un punto de vista exclusivamente semántico, la opción b es la mejor. Por otra parte, desde el punto de vista de la eficiencia tenemos que tener en cuenta que:

Opción A: El acceso a una tupla que refleje toda la información de una determinada entidad es mucho más rápido (no hace falta combinar varias relaciones).

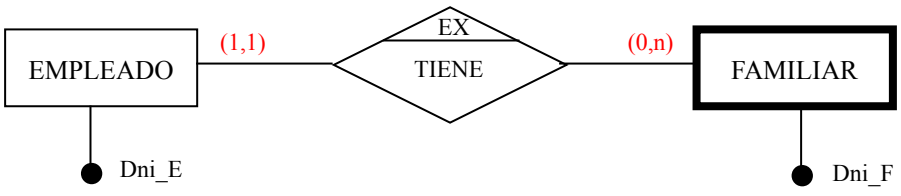
Opción B: La menos eficiente, aunque, como ya hemos señalado, es la mejor desde el punto de vista exclusivamente semántico.

Opción C: Con esta solución aumentamos la eficiencia ante determinadas consultas (las que afecten a todos los atributos, tanto comunes como propios, de un subtipo) pero la podemos disminuir ante otras. Esta solución es en la que se pierde más semántica: además si existe solapamiento se introduce redundancia que debe ser controlada si queremos evitar inconsistencias. Hay que tener en cuenta que esta solución no es válida cuando la generalización es parcial.

Elegiremos una estrategia u otra dependiendo de que sea la semántica o la eficiencia la que prime para el usuario en un momento determinado.

4.3.2.- Transformación de dependencias en identificación y en existencia.

- **Restricción de existencia**. La manera de transformar una relación con restricción de existencia es utilizar el mecanismo de propagación de clave, creando una clave ajena, con nulos no permitidos, en la relación de la entidad dependiente, con la característica de obligar a una modificación y un borrado en cascada.



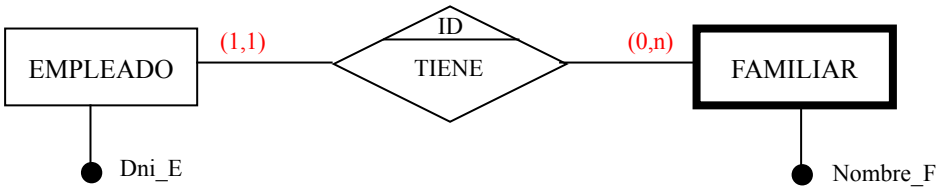
En el ejemplo se ha supuesto que el atributo Dni_F es suficiente para distinguir cada una de las ocurrencias del tipo de entidad FAMILIAR, por lo que el tipo de restricción es de existencia.

Las relaciones resultantes de la transformación serían:

EMPLEADO (Dni_E, ...)

FAMILIAR (Dni_F, ..., Dni_E, ...)
Caj.: Dni_E ⟶ EMPLEADO
NOT NULL
ON DELETE CASCADE
ON UPDATE CASCADE

- **Restricción de Identificación**. En esta caso, la clave primaria de la relación en la que se ha transformado la entidad débil debe estar formada por la concatenación de las claves de las dos entidades participantes en la relación.



En el ejemplo se ha supuesto que el atributo Nombre_F no es suficiente para distinguir cada una de las ocurrencias del tipo de entidad FAMILIAR, lo que convierte la restricción de la entidad débil en una restricción de identificación.

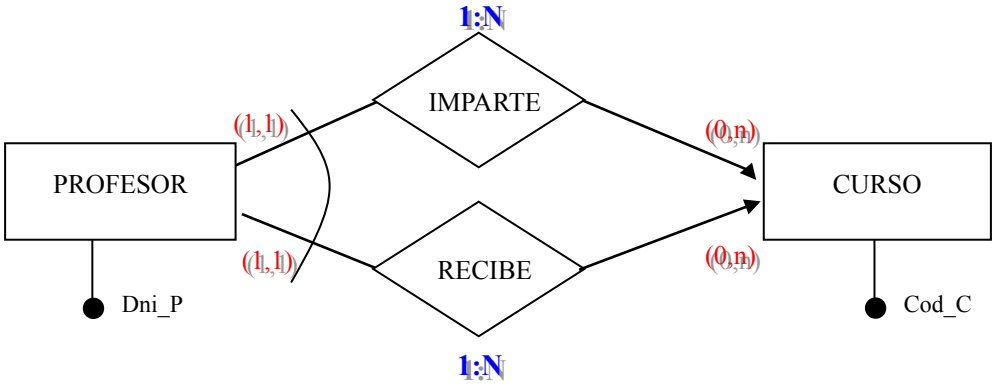
Las relaciones resultantes de la transformación serían:

EMPLEADO (Dni_E, ...)

FAMILIAR (Nombre_F, Dni_E, ...)
Caj.: Dni_E ⟶ EMPLEADO
ON DELETE CASCADE
ON UPDATE CASCADE

4.3.3.- Transformación de restricciones de relaciones.

- **Restricciones de exclusividad.** El modelo relacional no contempla este tipo de transformaciones. Para hacer que se cumpla una restricción habría que introducir las correspondientes restricciones en una cláusula CHECK en la creación de la tabla. Veámoslo con un ejemplo de restricción de exclusividad entre dos relaciones.



Las relaciones resultantes serían las siguientes:

PROFESOR (Dni_P, ...)

CURSO (Cod_C, ..., Dni_P_imparte, ..., Dni_P_recibe)
Caj.: Dni_P_imparte → PROFESOR *ON UPDATE CASCADE*
Dni_P_recibe → PROFESOR *ON UPDATE CASCADE*

Este esquema no refleja la restricción de exclusividad entre las relaciones DIRIGE y RECIBE. La restricción quedaría resuelta en la sentencia SQL de creación de la tabla CURSO a través de la cláusula **CHECK**:

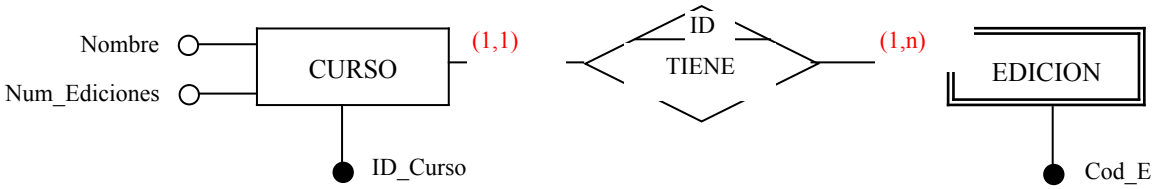
```
CREATE TABLE CURSO      ( Cod_C Dominio_codigos, ...
                          Dni_P_imparte Dominio_dni,
                          Dni_P_recibe Dominio_dni,
                          PRIMARY KEY (Cod_C)
                          FOREIGN KEY (Dni_P_imparte) REFERENCES PROFESOR
                                ON UPDATE CASCADE
                          FOREIGN KEY (Dni_P_recibe) REFERENCES PROFESOR
                                ON UPDATE CASCADE
                    CHECK (( Dni_P_imparte NOT IN (SELECT Dni_P_recibe FROM CURSO)
                    AND  Dni_P_recibe NOT IN (SELECT Dni_P_imparte FROM CURSO))));
```

El resto de restricciones entre relaciones se resolverían mediante la definición de CHECK y/o de aserciones de forma análoga al ejemplo anterior.

4.3.4.- Transformación de atributos derivados.

No existe para los atributos derivados una representación directa y concreta en el modelo relacional, por lo tanto se tratarán como atributos normales, que pasarán a ser columnas de la relación que corresponda.

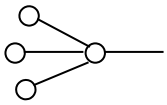
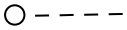
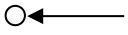
En este caso es preciso construir un disparador que calcule el valor del atributo derivado cada vez que se inserten o borren las ocurrencias de los atributos que intervienen en el cálculo de éste y añadir las restricciones correspondientes.



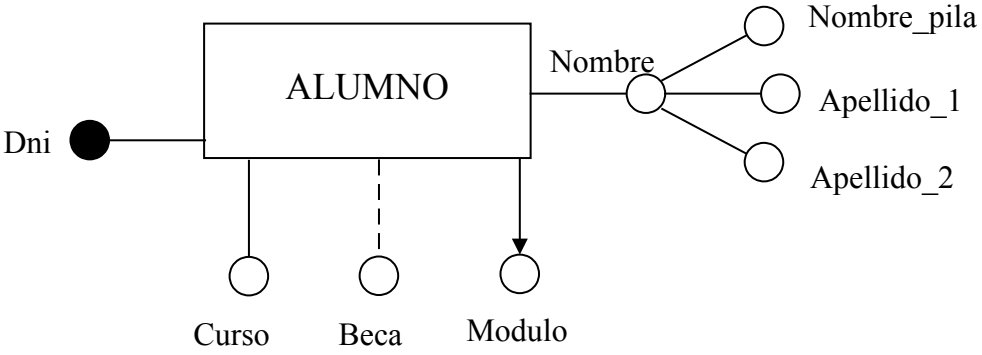
En el caso del ejemplo, cada vez que se inserte o borre una tupla de la tabla EDICION, el disparador debe actualizar el atributo *Num_ediciones* de la tabla CURSO.

Otra solución es no almacenar las columnas que provengan de atributos derivados, creando procedimientos que calculen los valores de éstos cada vez que se recuperan, lo que en ocasiones puede plantear problemas de semántica y en ocasiones de eficiencia.

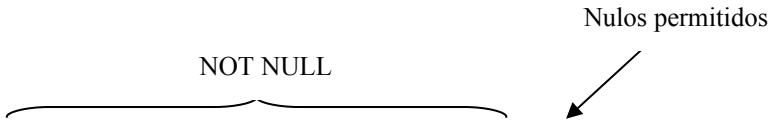
4.3.5- Transformación de atributos compuestos, multivaluados y opcionales

TIPO ATRIBUTO EN MODELO E-R	MODELO RELACIONAL	RESTRICCIONES
Compuesto obligatorio 	Un atributo por cada elemento	NOT NULL
Opcional 	Un atributo	Nulos permitidos
Multivaluado obligatorio 	Una relación con él y el atributo Identificador principal de su entidad, formando todos la clave primaria. El atributo identificador de la entidad, será clave ajena a la relación en que se transforma la entidad de procedencia.	Las de clave primaria

Ejemplo:



La transformación de la entidad ALUMNO al modelo relacional sería:



ALUMNO (Dni, Nombre_pila, Apellido_1, Apellido_2. Curso, Beca)

MODULO (Dni, Nombre_modulo)
Caj.: Dni ⟶ ALUMNO