Tema 4

DISEÑO LÓGICO: EL MODELO RELACIONAL

IES Francisco Romero Vargas
Departamento de Informática

1. El modelo E-R y el modelo relacional

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

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

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

2. Conceptos

RELACIÓN (TABLA)

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

NOMBRE

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

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

- **Atributo**

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

- **Tupla**

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

- **Dominio**

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

- **Grado**

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

- **Cardinalidad**

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

CLAVE

- **Clave candidata**

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

- **Clave primaria**

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

- **Clave alternativa**

Cualquier clave candidata que no sea primaria.

- **Clave externa, ajena o foránea**

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

RESTRICCIÓN

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

- Aquellas que son definidas por el hecho de que la base de datos sea relacional:
    - **No puede haber dos tuplas iguales**
    - **El orden de las tuplas no es significativo**
    - **El orden de los atributos no es significativo**
    - **Cada atributo sólo puede tomar un valor en el dominio en el que está inscrito**

- Aquellas que son incorporadas por los usuarios:

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

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

    - **Unicidad (*unique*)**

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

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

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

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

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

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

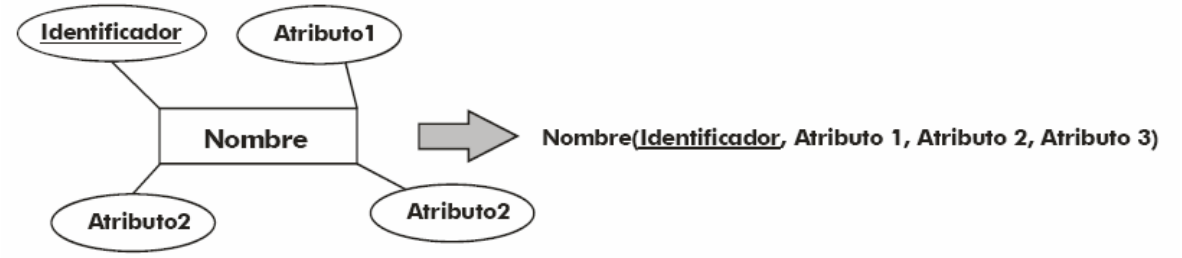
- **RESTRICT:** esta opción impide eliminar o modificar filas en la tabla referenciada si existen filas con el mismo valor de clave foránea.
- **CASCADE:** borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado de clave, implica borrar las filas con el mismo valor de clave foránea o modificar los valores de esas claves foráneas.
- **SET NULL:** borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado de clave, implica asignar el valor *NULL* a las claves foráneas con el mismo valor.
- **NO ACTION:** las claves foráneas no se modifican, ni se eliminan filas en la tabla que las contiene.
- **SET DEFAULT:** borrar o modificar una clave en una fila en la tabla referenciada con un valor determinado implica asignar el valor por defecto a las claves foráneas con el mismo valor.

ESQUEMA RELACIONAL

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

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

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

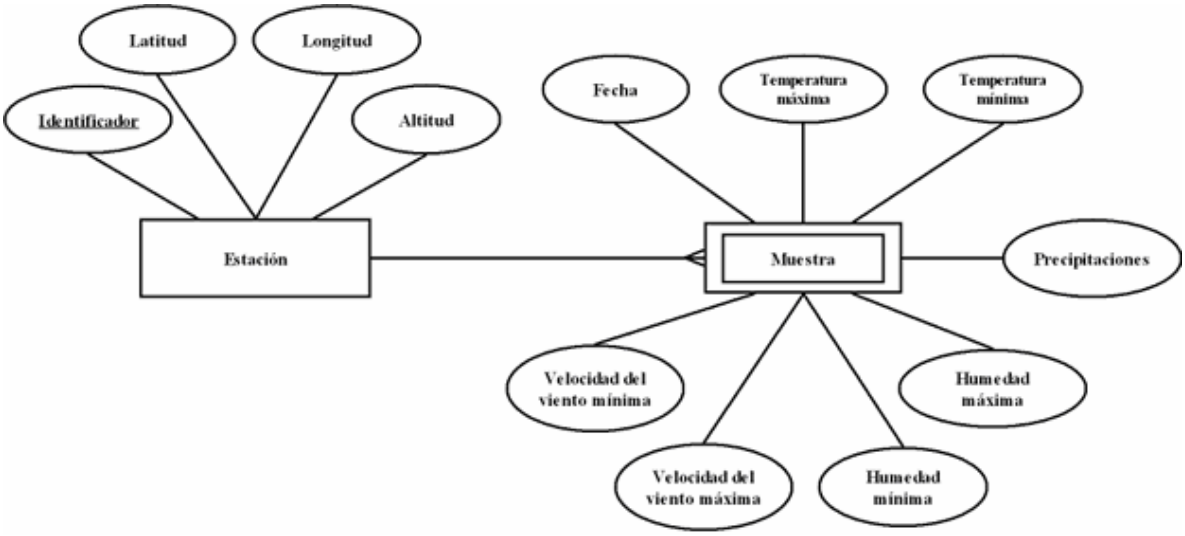


Por ejemplo, el esquema conceptual del ejercicio estaciones meteorológicas se expresa con los siguientes esquemas relacionales:

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

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

que corresponden al modelo entidad-relación:



3. Normalización

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

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

Las tablas obtenidas pueden presentar problemas:

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

- **Pérdida de restricciones de integridad.** Normalmente debido a *dependencias funcionales***.** Más adelante se explica este problema.
- **Anomalías en operaciones de modificación de datos.** El hecho de que al insertar un solo elemento haya que repetir tuplas en una tabla para variar unos pocos datos. O que eliminar un elemento suponga eliminar varias tuplas necesariamente (por ejemplo que eliminar un cliente suponga borrar seis o siete filas de la tabla de clientes, sería un error muy grave y por lo tanto un diseño terrible).

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

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

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

Por ejemplo, en el esquema

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

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

FORMAS NORMALES

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

• Primera forma normal (1FN)

Es una forma normal inherente al esquema relacional, por lo que su cumplimiento es obligatorio; es decir toda tabla realmente relacional la

cumple. Se dice que una tabla se encuentra en primera forma normal si impide que un atributo de una tupla pueda tomar más de un valor. La siguiente relación no cumple la primera forma normal:

TRABAJADOR

NOMBRE	Departamento
Elías	Informática
David	Coordinación Mantenimiento

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

TRABAJADOR

NOMBRE	Departamento
Elías	Informática
David	Coordinación
David	Mantenimiento

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

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

ALUMNO

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

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

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

ALUMNO

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

CURSOS

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

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

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

ALUMNO

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

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

ALUMNO

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

12002003	Manuela	08

PROVINCIA

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

4. Paso de Entidad-Relación al modelo relacional

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

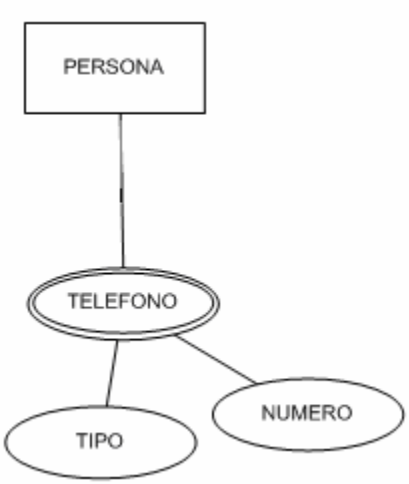
Estas reglas preparatorias se basan en la aplicación de la 1FN y su objetivo es eliminar las siguientes anomalías:

- Atributos con valores múltiples
- Atributos compuestos

ELIMINACIÓN DE ATRIBUTOS MÚLTIPLES

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

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



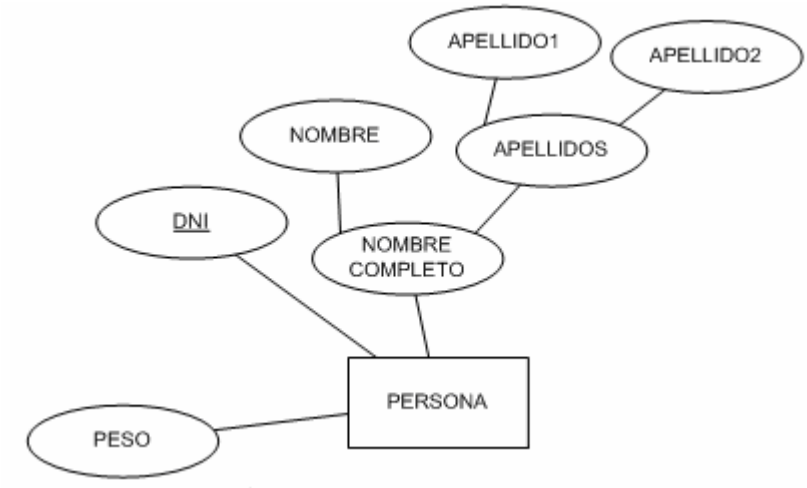
En este caso sería necesario expresar el esquema relacional de la forma:

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

ELIMINACIÓN DE ATRIBUTOS COMPUESTOS

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

El esquema entidad-relación:



Se expresaría mediante el siguiente esquema relacional:

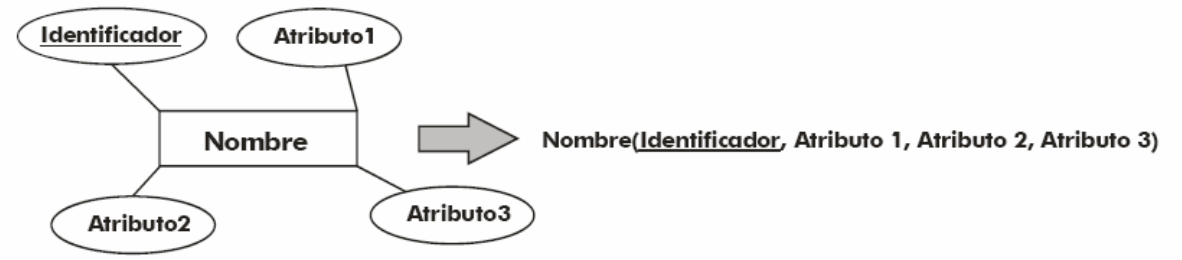
PERSONA (DNI, nombre, apellido1, apellido2, peso)

TRANSFORMACIÓN DE LAS ENTIDADES FUERTES

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

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

Esto hace que la transformación se produzca según este ejemplo:

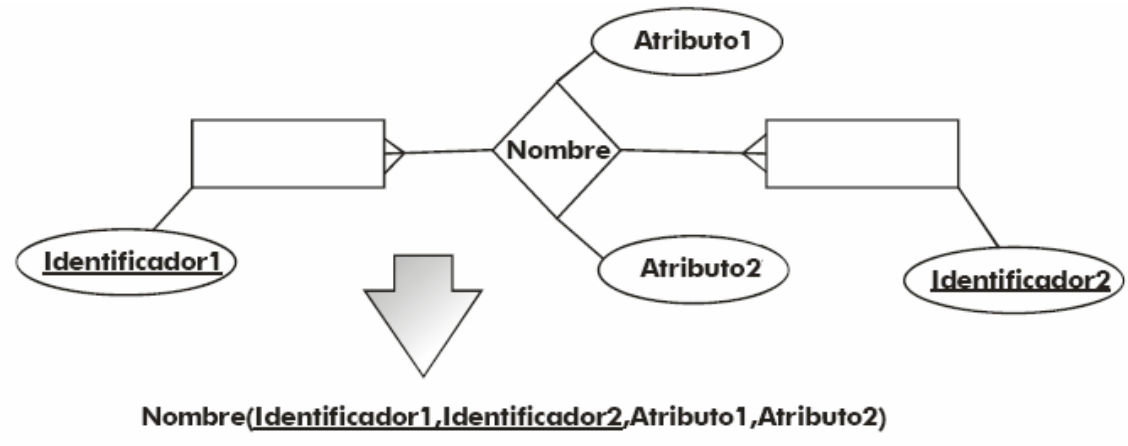


TRANSFORMACIÓN DE RELACIONES

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

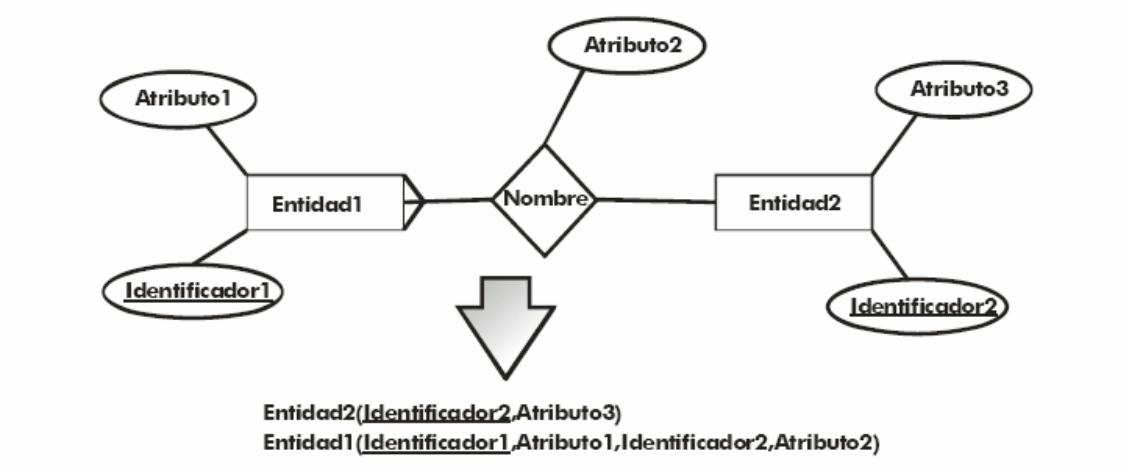
- **Relaciones varios a varios**

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



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

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

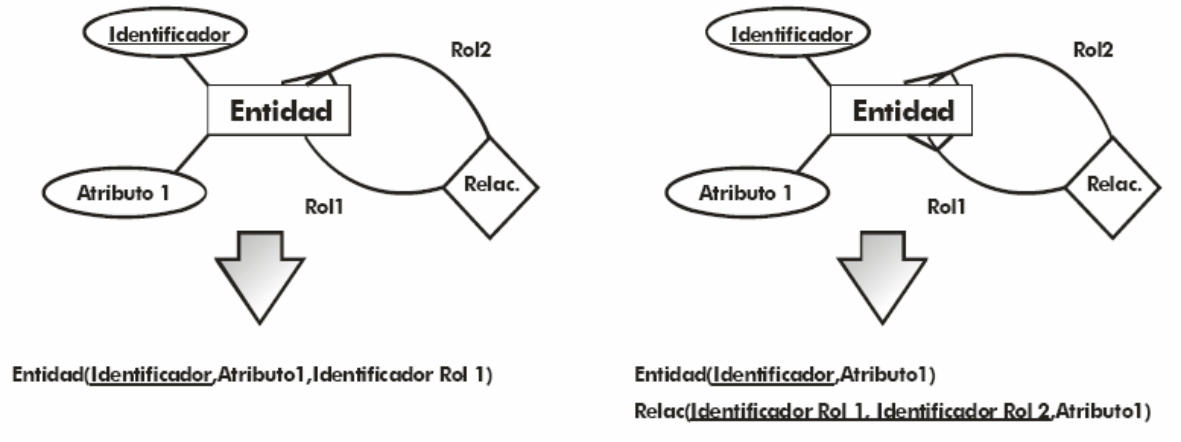


En el caso de las relaciones uno a uno, ocurre lo mismo: la relación no se convierte en tabla, sino que se coloca en una de las tablas (en principio daría igual cuál) el identificador de la entidad relacionada como clave externa. En el caso de que una entidad participe opcionalmente en la

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

- **Relaciones reflexivas**

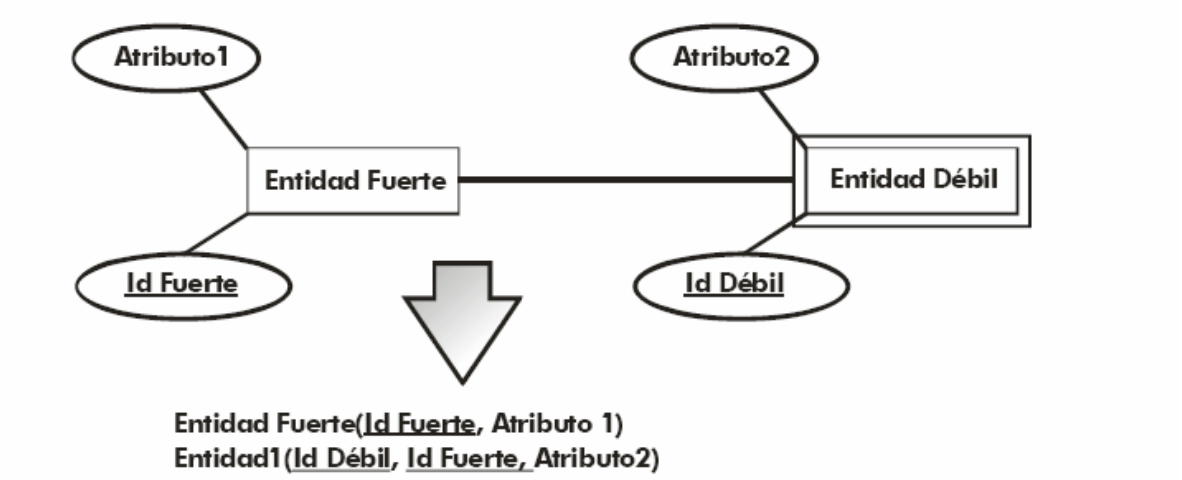
Las relaciones reflexivas o recursivas se tratan de la misma forma que las otras, sólo que un mismo atributo puede figurar dos veces en una tabla como resultado de la transformación.



TRANSFORMACIÓN DE LAS ENTIDADES DÉBILES

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

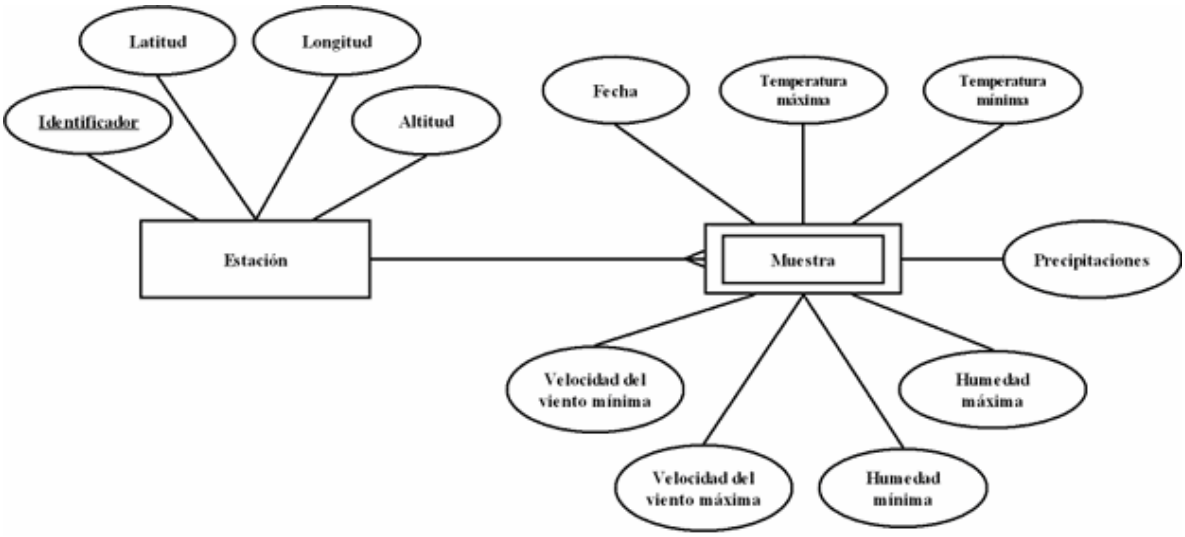
En ocasiones el identificador de la entidad débil es suficiente para identificar los ejemplares de dicha entidad, entonces ese identificador quedaría como clave principal, pero el identificador de la entidad fuerte seguiría figurando como clave externa en la entidad débil.



5. Ejemplo: diseño lógico y diseño físico partiendo de un diseño conceptual

DISEÑO CONCEPTUAL (Modelo Entidad-Relación)

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



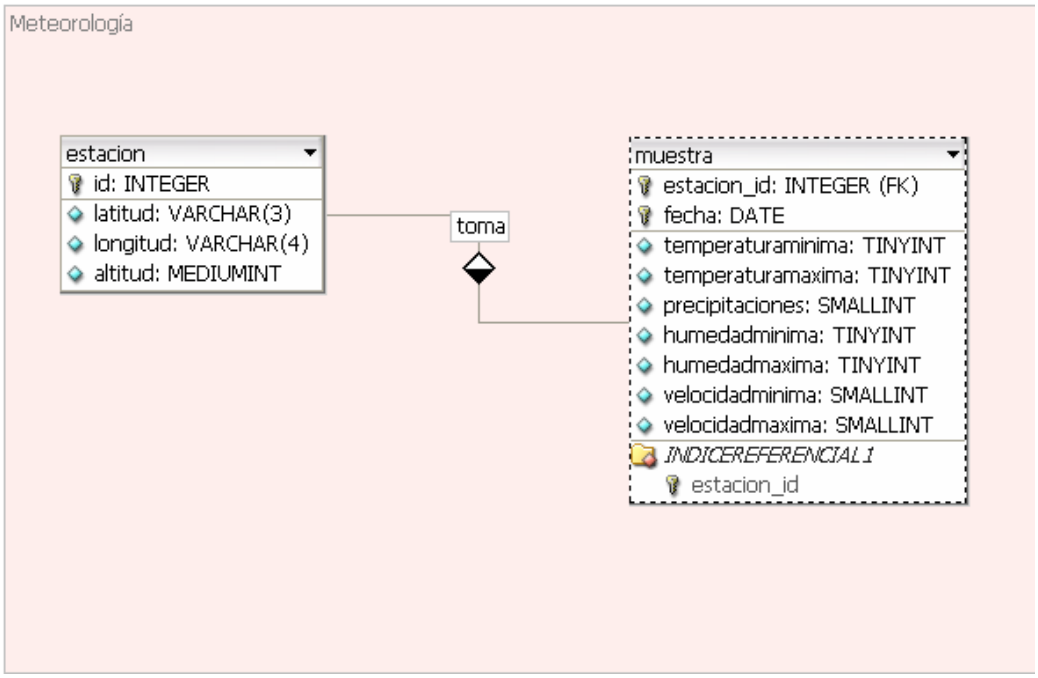
DISEÑO LÓGICO (Modelo relacional)

- Esquema relacional

estacion (id, latitutd, longitud, altitud)

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

- Representación gráfica (DBDesigner)



DISEÑO FÍSICO (Sentencias DDL)

```
CREATE TABLE estacion (
  id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  latitud VARCHAR(3) NOT NULL,
  longitud VARCHAR(4) NOT NULL,
  altitud MEDIUMINT UNSIGNED NOT NULL,
  PRIMARY KEY(id)
) ENGINE=InnoDB;
```

```
CREATE TABLE muestra (
  estacion_id INTEGER UNSIGNED NOT NULL,
  fecha DATE NOT NULL,
  temperaturaminima TINYINT,
  temperaturamaxima TINYINT,
  precipitaciones SMALLINT UNSIGNED,
  humedadminima TINYINT UNSIGNED,
  humedadmaxima TINYINT UNSIGNED,
  velocidadminima SMALLINT UNSIGNED,
  velocidadmaxima SMALLINT UNSIGNED,
  PRIMARY KEY (estacion_id, fecha),
  INDEX (estacion_id),
  FOREIGN KEY(estacion_id)
    REFERENCES estacion (id)
      ON DELETE CASCADE
      ON UPDATE NO ACTION
) ENGINE=InnoDB;
```

ANEXO. **Las 12 reglas de Codd**

Preocupado por los productos que decían ser sistemas gestores de bases de datos relacionales (RDBMS) sin serlo, Codd publica las 12 reglas que debe cumplir todo DBMS para ser considerado relacional. Estas reglas en la práctica las cumplen pocos sistemas relacionales. Las reglas son:

1. **Información**. Toda la información de la base de datos debe estar representada explícitamente en el esquema lógico. Es decir, todos los datos están en las tablas.
2. **Acceso garantizado**. Todo dato es accesible sabiendo el valor de su clave y el nombre de la columna o atributo que contiene el dato.
3. **Tratamiento sistemático de los valores nulos**. El DBMS debe permitir el tratamiento adecuado de estos valores.
4. **Catálogo en línea basado en el modelo relacional.** Los metadatos deben de ser accesibles usando un esquema relacional.
5. **Sublenguaje de datos completo**. Al menos debe de existir un lenguaje que permita el manejo completo de la base de datos. Este lenguaje, por lo tanto, debe permitir realizar cualquier operación.
6. **Actualización de vistas.** El DBMS debe encargarse de que las vistas muestren la última información
7. **Inserciones, modificaciones y eliminaciones de dato nivel.** Cualquier operación de modificación debe actuar sobre conjuntos de filas, nunca deben actuar registro a registro.
8. **Independencia física.** Los datos deben de ser accesibles desde la lógica de la base de datos aún cuando se modifique el almacenamiento.
9. **Independencia lógica.** Los programas no deben verse afectados por cambios en las tablas
10. **Independencia de integridad.** Las reglas de integridad deben almacenarse en la base de datos (en el diccionario de datos), no en los programas de aplicación.
11. **Independencia de la distribución.** El sublenguaje de datos debe permitir que sus instrucciones funciones igualmente en una base de datos distribuida que en una que no lo es.
12. **No subversión**. Si el DBMS posee un lenguaje que permite el recorrido registro a registro, éste no puede utilizarse para incumplir las reglas relacionales.