Teórico 9 – Del MER al MR

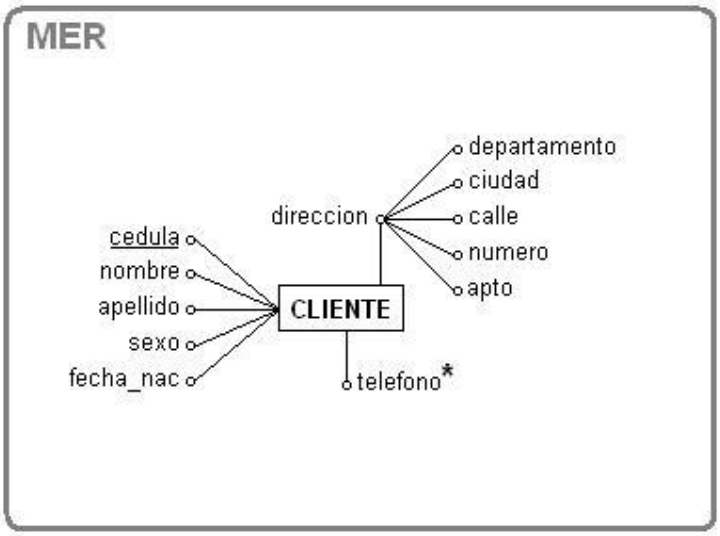
Introducción

Veremos cómo traducir un modelo conceptual, en forma de Modelo Entidad-Relación, en un modelo lógico de base de datos, en forma de Modelo Relacional. Para esto, estudiaremos cómo traducir cada uno de los conceptos del MER, en orden creciente de complejidad.

Entidades fuertes

Consideremos una empresa que quiere mantener datos de sus clientes. Se debe mantener la cédula de los clientes, que los determina. Se desea mantiener además el nombre, apellido, sexo, fecha de nacimiento, dirección completa (con departamento, ciudad, calle, número y apartamento si corresponde) y números de teléfono (que pueden ser más de uno).

El siguiente podría ser parte del MER, donde se reflejan los datos relevantes del cliente.

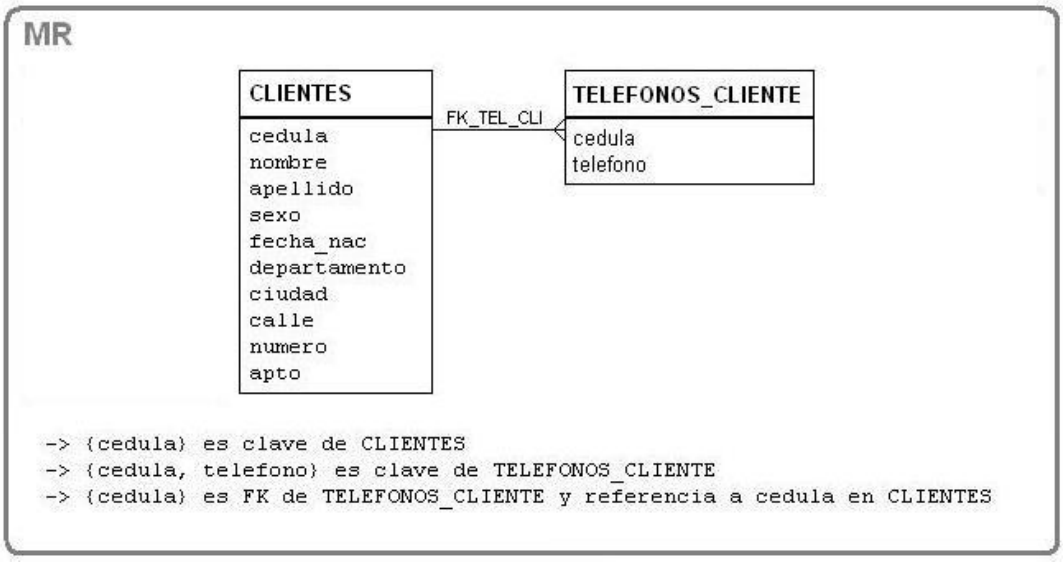


Como se ve, hay una entidad fuerte con atributos simples, atributos estructurados y atributos multivaluados.

Para traducir una entidad fuerte al Modelo Relacional, procederemos de la siguiente manera:

- Crearemos una relación por cada entidad fuerte del MER, con el mismo nombre pero en plural
- Crearemos una columna en la relación por cada atributo simple del MER, con el mismo nombre
- Crearemos una columna en la relación por cada hoja en la estructura de los atributos estructurados
- Se especificarán claves correspondientes a cada conjunto de atributos determinantes, y una de ellas se elegirá (arbitrariamente, por ahora) como clave primaria (primary key)
- Para los atributos multivaluados, se creará una nueva relación con una columna correspondiente al atributo multivaluado, se agregarán las columnas correspondientes a la clave primaria de la relación que corresponde a la entidad fuerte, que serán una clave foránea y se especificará además el conjunto de todas las columnas de la relación como una clave.

Para el caso del ejemplo, al pasar al Modelo Relacional tendremos algo como lo siguiente:



Esta notación (conocida como crow's foot) es un estándar de facto entre las herramientas que permiten trabajar con el Modelo Relacional.

Las relaciones se muestran como cajas con dos secciones, una para el nombre de la relación, y otra para el nombre de las columnas. Opcionalmente se podría especificar, para cada columna, el tipo de datos y si admite o no el valor NULL.

A veces suele representarse de una forma especial (en negritas o con subrayado) la clave primaria de cada relación.

Las relaciones de dependencia dadas por la Foreign Keys, suelen representarse como una línea entre las relaciones participantes, terminada en un símbolo como una pata de pájaro, del lado de la relación que tiene la FK. Opcionalmente, las FK pueden tener un nombre sobre la línea que las representa.

Más allá de las posibilidades de notación de los diagramas de Modelo Relacional, conviene tomar nota de las claves y claves foráneas al pie del diagrama.

Notaremos algunas cuantas cosas del ejemplo antes de seguir adelante. En primer lugar, pensemos que hubiera pasado si la entidad CLIENTE hubiera tenido otro atributo determinante, por ejemplo: credencial. Los dos hubieran constituido claves de la relación CLIENTES, de forma que hubiéramos tenido que elegir una de las dos claves para mantener en la relación TELEFONOS_CLIENTE.

Cuando haya más de una clave candidata, las distinguiremos en dos tipos, una será la PRIMARY KEY, y elegiremos para esto una de las claves que no admita valores nulos. A las claves restantes, admitan o no valores nulos, las llamaremos UNIQUE KEYS. En nuestro ejemplo, si hubiéramos tenido un atributo determiante credencial, pero admitiera valores nulos, la columna cedula seguiría siendo la PRIMARY KEY, mientras que credencial sería una UNIQUE KEY.

En ocasiones, una entidad no tiene una clave candidata claramente identificable, o bien tiene una compuesta de muchos atributos, y ya sabemos que la PK de una relación se "propaga", por ejemplo cuando hay atributos multivaluados. En estos casos entonces, nos podemos preguntar qué opción tenemos para elegir una PK adecuada, y para estos casos conviene considerar "inventar" una clave para que funcione como PK. A esto lo llamaremos surrogate key.

Los valores de una surrogate key, al no tener semántica, podrían ser generados automáticamente en la medida que sea por un método que asegure la unicidad. Los motores relacionales suelen contar con generación automática de valores únicos, que pueden utilizarse a tales efectos.

Para ilustrar el concepto de surrogate key, veremos una debilidad del Modelo Relacional que hemos generado, observando una instancia de la relación CLIENTES.

cedula	nombre	…	departamento	ciudad	…
32647389	Juan		Montevideo	Montevideo	
42536745	Alberto		Montevideo	Centro	
74658104	María		Lavalleja	Minas	
75614332	Carlos		Lavalleja	La Paloma	
10096265	Pedro		San Gregorio	de Polanco	

Queda de manifiesto que pueden existir valores para departamento que no corresponden a un departamento (San Gregorio), valores para ciudad que no corresponden a una ciudad (de Polanco), y combinaciones incorrectas de departamento y ciudad (no existe una ciudad La Paloma en Lavalleja).

Para solucionar el problema de los datos inexistentes, observamos que el problema radica en que los nombres de departamentos y ciudades deben pertenecer a un dominio mucho más reducido que el conjunto VARCHAR(30), por ejemplo.

Podríamos pensar en mantener el conjunto de los nombres válidos de departamento, en una relación con una única columna; y lo mismo para ciudades. En cada caso, la única columna de la relación constituirá la Primary Key de la misma.

DEPARTAMENTOS

nombre
Montevideo
Canelones
…

{nombre} es PK de DEPARTAMENTOS

CIUDADES

nombre
Montevideo
Canelones
San Gregorio de Polanco
…

{nombre} es PK de CIUDADES

Teniendo definidas estas relaciones y especificando Foreign Keys como las siguientes en la relación CLIENTES, solucionamos el problema de los departamentos y ciudades inexistentes:

- {departamento} es FK en CLIENTES, y referencia a nombre en DEPARTAMENTOS
- {ciudad} es FK en CLIENTES, y referencia a nombre en CIUDADES

Parecería que la elección de la clave primaria en DEPARTAMENTOS y CIUDADES era la única posible ya que estas relaciones tienen una única columna, y conceptualmente están especificando un dominio. En estos casos, suelen utilizarse surrogate keys, "inventando" una columna con un código para cada uno de los valores del dominio.

DEPARTAMENTOS

codigo	nombre
1	Montevideo
2	Canelones
…	…

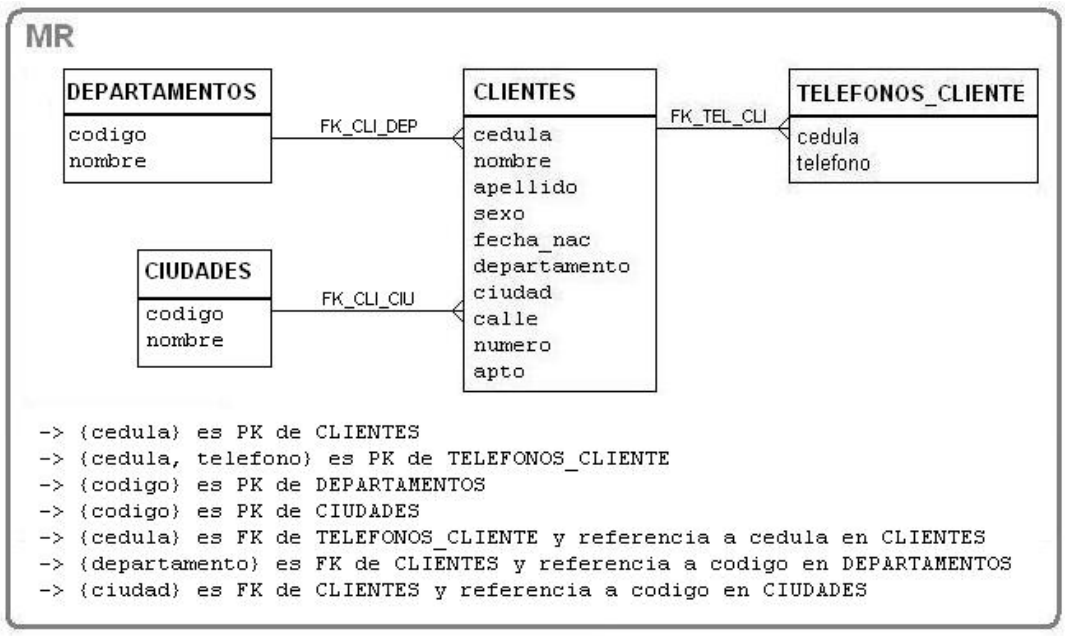
{codigo} es PK de DEPARTAMENTOS
{nombre} es UK de DEPARTAMENTOS

CIUDADES

codigo	nombre
1	Montevideo
2	Canelones
17	San Gregorio de Polanco
…	…

{codigo} es PK de CIUDADES
{nombre} es UK de CIUDADES

El diagrama del Modelo Relacional, quedaría de la siguiente forma:



Note que el MER no teníamos una entidad DEPARTAMENTO ni una entidad CIUDAD, sino que agregamos estas relaciones al Modelo Relacional para solucionar el problema de restringir los dominios de algunas columnas. Estas relaciones especiales con códigos y valores formando un dominio, se conocen muchas veces como “codigueras”.

Ahora bien, ¿qué ganamos al agregar la surrogate key codigo en las nuevas relaciones? Por lo pronto, espacio. Si bien no entraremos en detalles de diseño físico, las codigueras son tan comunes que merecen una mención.

Imagine que para almacenar el nombre de una ciudad como “San Gregorio de Polanco” se necesitan 25 bytes (uno por cada carácter más dos para delimitar el fin de la cadena). Si tenemos 1.000 clientes de San Gregorio de Polanco tendremos 25.000 bytes en la tabla CLIENTES dedicados a mantener esta información. Si podemos tener un código entero de 4 bytes para cada ciudad, ocuparemos 4.000 bytes en la tabla CLIENTES en lugar de 25.000 para mantener la misma información.

La introducción de las codigueras lo vemos como un elemento agregado en el Modelo Relacional, porque es difícil que el MER se haya considerado una restricción no estructural del tipo “El atributo departamento de la entidad CLIENTE debe ser uno de los siguientes: Montevideo, Canelones, …”; aunque es válido que existan este tipo de restricciones.

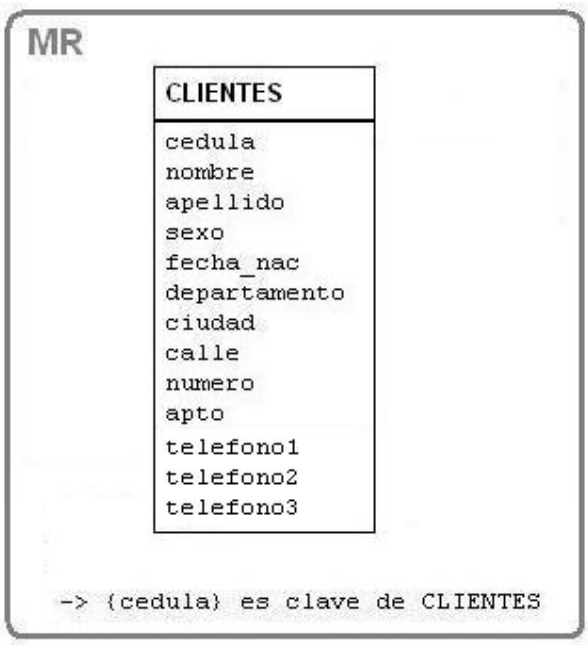
Algo más que mencionaremos en este punto antes de seguir avanzando, está relacionado a los atributos multivaluados. Alguien podría haber supuesto que existe un número máximo de teléfonos para un cliente (digamos 3), y haber construido un Modelo Relacional con una sola relación CLIENTES, con columnas telefono1, telefono2 y telefono3.

Este enfoque presenta varios problemas. El primero es que una regla del diseño relacional es que las columnas deberían separar entidades conceptualmente diferentes, pero el telefono1 no es conceptualmente diferente del telefono2 de un cliente: ambos son teléfonos de un cliente, y por lo tanto deberían estar en una misma columna de una relación. Esta es una de las acepciones de “no repeating groups” que es el requisito para asegurar que una tabla está en primera forma normal (hablaremos de normalización más adelante).

Un segundo problema es que cuando un cliente tenga menos de tres teléfonos declarados deberíamos mantener un valor NULL para mantener la semántica de “dato desconocido”. Pero

ya sabemos los problemas que acarrean los NULL, así que sería mejor que un cliente con sólo un teléfono tuviera una sola tupla en la relación TELEFONOS_CLIENTE, evitando tener que utilizar el valor NULL.

Un tercer problema de tener una tabla desnormalizada de esta manera, es operacional. Ciertas operaciones, como contar los teléfonos de cada cliente se podrían realizar más fácilmente con el Modelo Relacional en primera forma normal.



Aún así, existen casos en la realidad en que se mantienen tablas desnormalizadas a propósito, aunque no estudiaremos los motivos que llevan a esto por ahora.