# Tema 6: Diseño de bases de datos relacionales.

## *6.1 Introducción.*

Las dificultades inherentes al diseño de una base de datos han de afrontarse con procedimientos ordenados y metódicos. En el proceso de diseño de una base de datos hemos de distinguir tres grandes fases:

- Diseño conceptual, cuyo objetivo es obtener una representación de la información con independencia de usuarios y aplicaciones en particular, y fuera de consideraciones sobre la eficiencia del ordenador.

- Diseño lógico, cuyo objetivo es transformar el diseño conceptual obtenido y adaptarlo al modelo de datos en el que se apoya el SGBD que se va a utilizar. En nuestro caso, el SGBD es relacional, por lo cual nos referiremos a este modelo de datos.

- Diseño físico, cuyo objetivo es conseguir una instrumentación lo más eficiente posible del diseño lógico.

En este tema nos centraremos principalmente en el diseño conceptual y el diseño lógico, pues el diseño físico depende de cada SGBD y cada computadora en particular.

Para desarrollar el diseño de una base de datos, tomaremos como ejemplo el diseño de una base de datos relacional que permita la gestión de prestamos de libros de una biblioteca.

## *6.2 Diseño conceptual.*

El diseño conceptual, brevemente expresado, consiste en extraer del trabajo de la empresa aquellas entidades y acciones que son de uso habitual en la misma y que van a formar parte de la base de datos.

Para ello, la forma habitual de diseño es mediante la consulta con los empleados de la empresa, pues a partir de la misma se ha de obtener el conjunto de entidades que van a formar parte de la base de datos, así como las acciones relevantes que pueden afectar al diseño de la base de datos.

En nuestro ejemplo de estudio, partimos de que la forma actual de trabajo de la biblioteca, la cual consiste en una serie de fichas de tres tipos:

- Fichas con las características de los libros (nombre, código, tipo, etc.).

- Fichas con las características de los lectores (nombre, apellidos, domicilio, etc.).

- Fichas con la información de los prestamos de libros que se han efectuado, incluyendo el lector a quién se le ha prestado, la fecha, etc.

Además de estas fichas, en nuestras conversaciones con los empleados, obtenemos algunas informaciones y comentarios útiles para el diseño como los siguientes:

- De cada libro pueden existir varios ejemplares.
- Sé esta interesado en tener información sobre el idioma del libro.
- Interesa reflejar los temas de los libros, pudiendo cada libro pertenecer a varios temas y/o subtemas.
- Interesa conocer el nombre de los autores.

A partir de esta información podemos obtener el siguiente diseño conceptual, donde se incluye la cardinalidad entre las entidades. En dicho diseño, los rectángulos representan entidades y los rombos representan relaciones entre entidades, constando al lado de las mismas la cardinalidad de la relación.

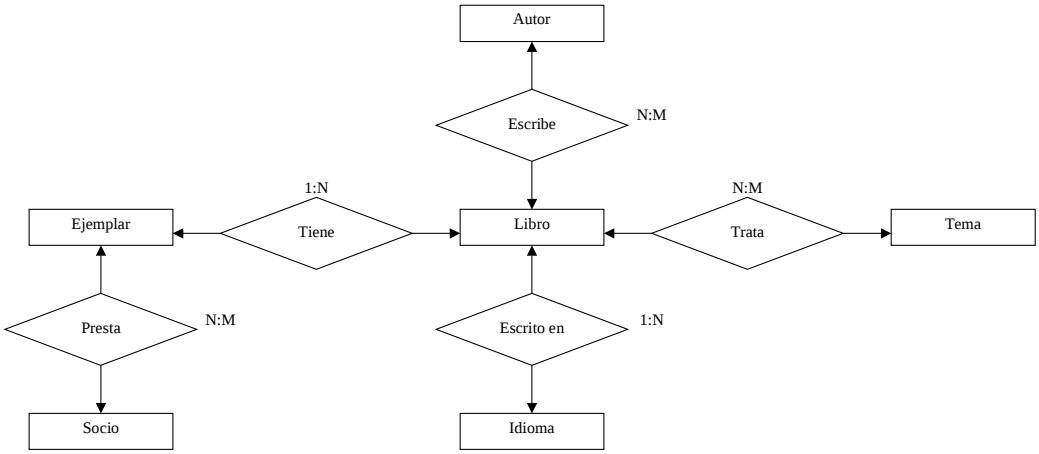


*Figura 6.2.1: Esquema del diseño conceptual de una base de datos.*

La cardinalidad es obtenida en base a las posibilidades de relación entre las entidades, existiendo tres tipos de cardinalidad:

- Cardinalidad 1:1, que es cuando una entidad A se relaciona solo con otra entidad B y viceversa. Por ejemplo, el identificador de un coche (número de bastidor) se corresponde con una matrícula y esa matrícula con ese identificador del coche.
- Cardinalidad 1:N, que es cuando una entidad A se puede relacionar con N entidades B pero no al revés. Por ejemplo un libro puede tener N ejemplares, pero un ejemplar es solo de un libro.
- Cardinalidad N:M, que es cuando una entidad A se relaciona con N entidades B y viceversa. Por ejemplo, un libro puede ser escrito por varios autores distintos y un autor puede escribir varios libros distintos.

Así, un libro puede haber sido escrito por varios autores (relación 1:N), pero además, un autor puede haber escrito varios libros (relación 1:N en sentido inverso), por lo cual, la relación resultante es N:M.

Con la obtención del esquema de la figura, puede considerarse por realizado el diseño conceptual, pues tenemos identificados los elementos y las relaciones entre ellos así como sus cardinalidades, pudiendo pasar al diseño lógico.

### *6.3 Diseño lógico.*

La conversión del diseño conceptual al diseño lógico está basada en los tres principios siguientes:

- Todo tipo de entidad del modelo conceptual se convierte en una tabla.
- Todo tipo de relación entre tablas 1:N se traduce en una propagación de la clave (se crea una clave primaria o foránea) o bien se crea una nueva tabla intermedia.
- Todo tipo de relaciones entre tablas N:M (muchos a muchos) origina la creación de una nueva tabla intermedia.

En primer lugar, se observa que de la aplicación de las reglas anteriores, en el paso del diseño conceptual al diseño lógico se pierde información semántica, pues tanto las entidades como las relaciones son convertidas en tablas, sin que exista una diferencia entre las provenientes de entidades o de relaciones.

Apliquemos las tres reglas anteriores a nuestro diseño conceptual.

Aplicando la primera regla, obtenemos que como existen seis entidades (autor, libro, ejemplar, tema, idioma y socio), hemos de crear seis tablas, una por cada entidad, obteniendo las siguientes seis tablas con su correspondiente clave primaria:

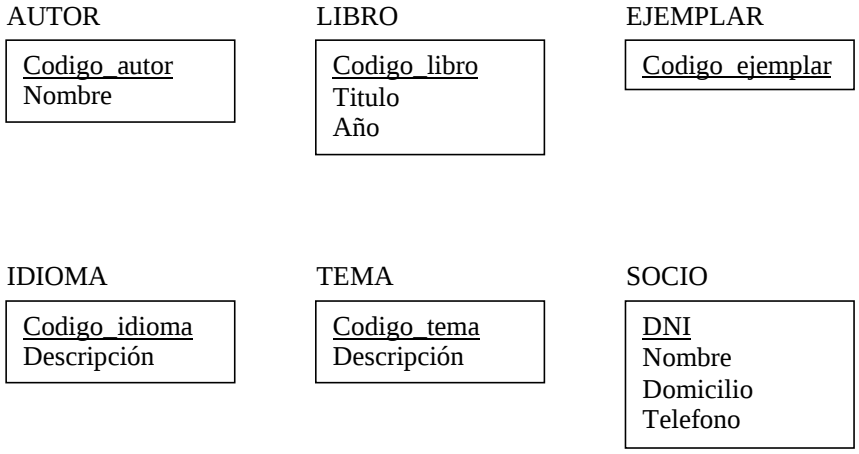


*Figura 6.3.1: Tablas de la base de datos obtenidas a partir de las entidades existentes.*

Apliquemos ahora la segunda regla. Tenemos dos relaciones 1:N, lo cual nos origina la propagación de las claves. La propagación se efectúa desde la tabla de cardinalidad 1 a la tabla de cardinalidad N. Además, la propagación es de clave primaria

si la clave primaria de la tabla a la que se propaga la clave no identifica unívocamente la entidad, en caso contrario se propaga en forma de clave foránea.

En nuestro caso, la propagación de clave es:

- La clave codigo_libro de la tabla LIBRO se propaga a la tabla EJEMPLAR como clave primaria.
- La clave codigo_idioma de la tabla IDIOMA se propaga a la tabla LIBRO como clave foránea.

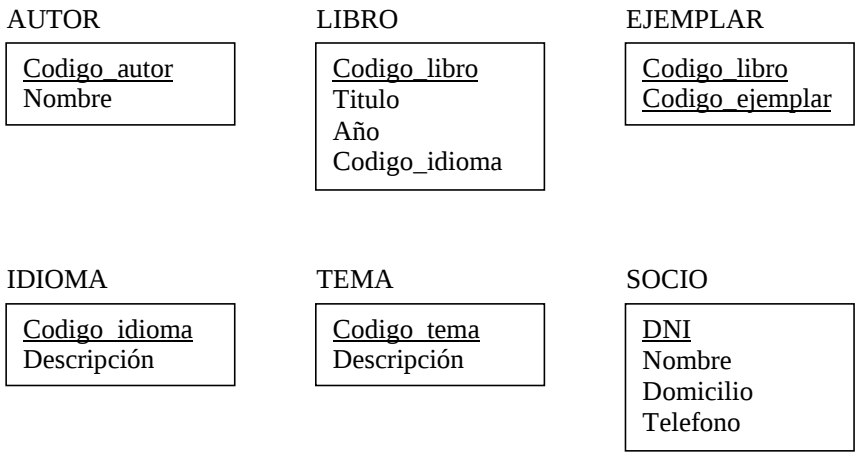


*Figura 6.3.2: Modificación de las entidades al aplicar las relaciones 1:N.*

Apliquemos ahora la tercera regla. Tenemos tres relaciones N:M, cada una de las cuales dará lugar a una nueva tabla que estará compuesta por las claves primarias de las tablas que relaciona así como por todos aquellos datos que identifican la relación entre las entidades. De esta forma obtendremos la siguiente relación final de tablas:

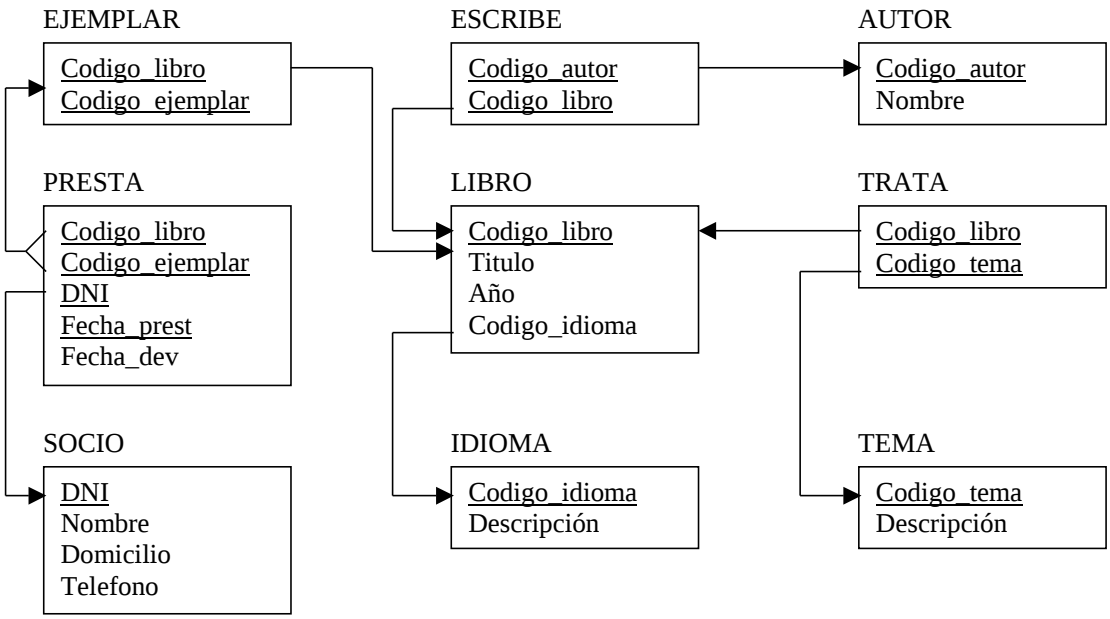


*Figura 6.3.3: Modificación de las entidades al aplicar las relaciones N:M.*

## *6.4 Teoría de la normalización.*

En el desarrollo del diseño lógico obtenemos una serie de tablas finales que son las candidatas a formar nuestra base de datos. Sin embargo, dichas tablas han sido obtenidas a partir de un diseño conceptual elaborado sin ningún tipo de reglas, por lo que podemos obtener un diseño de tablas más o menos heterogéneo.

La teoría de la normalización consiste en un conjunto de reglas formales que nos permiten asegurar que un diseño lógico cumple una serie de propiedades, corrigiendo la estructura de los datos de las tablas y evitando una serie de problemas como:

- Incapacidad de almacenar ciertos hechos.
- Redundancias y, por tanto, posibilidad de inconsistencias.
- Ambigüedades.
- Pérdida de información.
- Aparición en la base de datos de estados no válidos en el mundo real, es lo que se llama anomalías de inserción, borrado y modificación.

Las reglas formales de la teoría de la normalización son conocidas con el nombre de formas normales. Existen seis formas normales, de forma que cuando la base de datos cumple las reglas de la primera forma normal se considera que está en primera forma normal (1FN), cuando pasan la segunda, que está en segunda forma normal (2FN), etc. Además, una base de datos de la que se afirme que está en 2FN, está también en 1FN, pues las formas normales se aplican de forma sucesiva.

De las seis formas normales, generalmente solo se aplican sobre las bases de datos las tres primeras, considerando que una base de datos que está en 3FN es una base de datos correctamente diseñada. Por ello, expondremos a continuación estás tres primeras formas normales.

Para desarrollar la teoría de normalización de una base de datos tomaremos como ejemplo el diseño de la gestión de una empresa considerando solo la parte de facturación a los clientes.

### 6.4.1 Primera forma normal (1FN).

Una base de datos se considera que está en 1FN si cada atributo (campo) de una tabla contiene un solo valor atómico (simple). Un atributo que contiene varios valores puede derivar en una perdida de datos.

Tomando el ejemplo propuesto, hemos realizado un análisis y hemos obtenido que para identificar la factura hemos creado como clave primaria el código de la factura y hemos establecido además la necesidad de que una factura posea los siguientes campos:

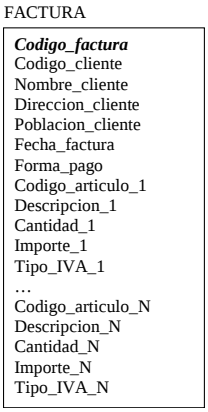


*Figura 6.4.1.1: Diseño inicial de las facturas.*

Analizando el diseño inicial de la tabla FACTURA, observamos la existencia de múltiples valores para los atributos (campos) siguientes: Codigo_articulo, descripcion, cantidad, importe e IVA. Analizando la tabla, observamos que no cumple la condición de 1FN. La solución consiste en crear una nueva tabla, que podemos llamar DETALLE_FACTURA a la cual se trasladan los datos repetitivos, en nuestro caso los datos referentes a los artículos (codigo_articulo, descripción, cantidad, importe e IVA).

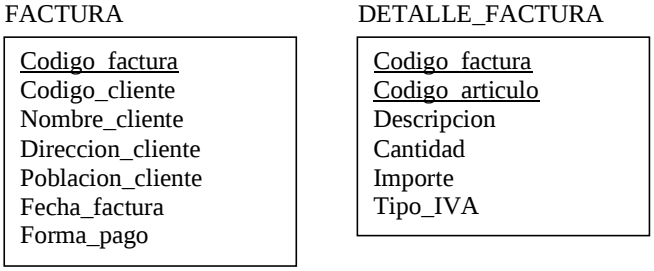


*Figura 6.4.1.2: Diseño de las facturas aplicando la 1FN.*

Cuando se produce la separación de datos de la tabla original en una nueva tabla, ésta además de los atributos necesarios, traslada la clave primaria de la tabla original como parte de su nueva clave primaria, que estará formada, generalmente, por dos atributos.

## 6.4.2 Segunda forma normal (2FN).

La segunda forma normal, como la tercera que veremos a continuación, se relaciona con el concepto de dependencia funcional.

Entendemos como dependencia funcional a la relación que tienen los atributos (campos) de una tabla con otros atributos de la propia tabla. Un campo tiene dependencia funcional si necesita información de otro/s campo/s para poder contener un valor.

Una tabla se dice que esta en segunda forma normal (2FN) si sucede que:

- Está en 1FN

- Cada atributo (campo) no clave depende de la clave completa, no de parte de ella.

Por supuesto, una base de datos estará en 2FN si todas sus tablas lo están.

La idea intuitiva de la 2FN es identificar todas las tablas con una clave compuesta, pues todas las tablas con clave simple están por defecto en 2FN si están en 1FN, y comprobar que cada uno de los campos de esta tabla depende de la clave completa.

En nuestro ejemplo, la tabla FACTURA se encuentra en 2FN pues está en 1FN y su clave es simple. Sin embargo la tabla DETALLE_FACTURA ha de ser analizada pues su clave es compuesta (esta formada por dos atributos).

Analizando la tabla DETALLE_FACTURA, observamos que el atributo descripcion depende únicamente del atributo codigo_articulo (la descripción de un articulo depende únicamente de que articulo se trate y es completamente independiente de la factura), por lo cual la descripcion ha de ser llevada a una nueva tabla junto con el atributo clave codigo_articulo.

Supongamos además, que el cliente nos indica que en la factura, aunque cada articulo posee calculado su IVA, el tipo de IVA que aplica es común a toda la factura y no depende en cada factura de los artículos, por lo cual, vemos que el atributo Tipo_IVA solo depende funcionalmente del codigo_factura y no depende de codigo_articulo, por lo cual ha de ser devuelta a la tabla FACTURA como un único atributo Tipo_IVA que depende solo de la clave de FACTURA (codigo_factura).

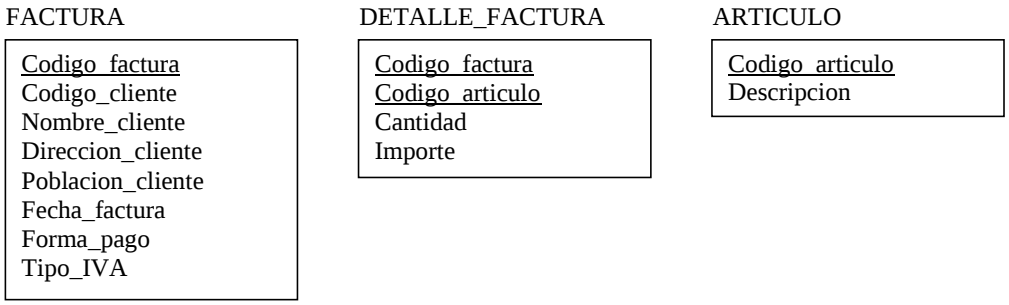


*Figura 6.4.2.1: Diseño de las facturas aplicando la 2FN.*

### 6.4.3 Tercera forma normal (3FN).

Una tabla se dice que está en tercera forma normal (3FN) si:

- Está en 2FN.

- Todos los atributos que no son claves deben ser mutuamente independientes, es decir, un atributo no debe depender de otro atributo no clave de su tabla.

Si un atributo que no es clave depende de otro atributo que no es clave, la tabla posiblemente contiene datos acerca de mas de una entidad, contradiciendo el principio de que cada tabla almacene información de una entidad.

En nuestro ejemplo, podemos observar que las tablas ARTICULO y DETALLE_FACTURA se encuentran en 3FN. Sin embargo, la tabla FACTURA no está en 3FN, pues los atributos Nombre_cliente, Direccion_cliente y Poblacion_cliente dependen funcionalmente del atributo Codigo_cliente, campo que no es clave. Por ello, debemos extraer estos atributos de la tabla FACTURA e incluirlos en una nueva tabla que haga referencia al cliente, tabla que llamaremos CLIENTE y que contendrá como clave primaria el Codigo_cliente y como atributos el Nombre_cliente, Direccion_cliente y Poblacion_cliente. Aplicando esto, nuestro diseño de las facturas da lugar a las tablas que pueden verse en la siguiente figura.

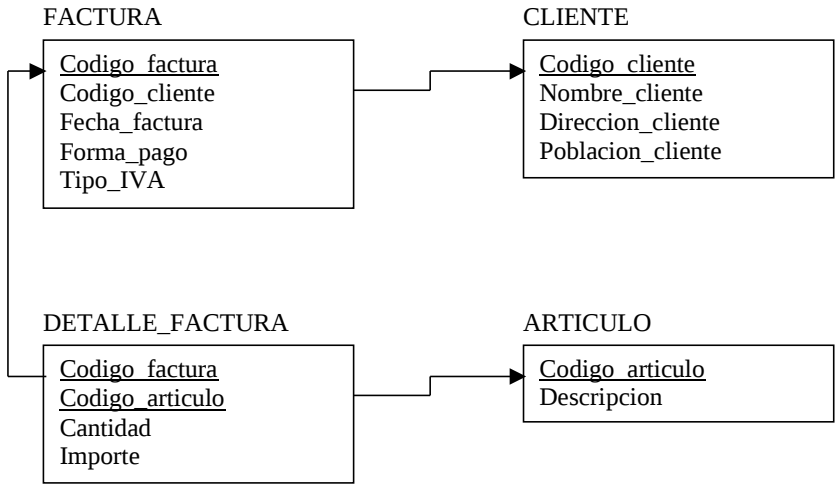


*Figura 6.4.3.1: Diseño de las facturas aplicando la 3FN.*

### 6.4.4 Consideraciones finales y problemas de la normalización.

La teoría de la normalización nos ayuda a estructurar mejor las tablas de la base de datos, evitando posibles redundancias.

Por otra parte, si seguimos la metodología de diseño expuesta en los puntos 6.2 y 6.3 del tema, obteniendo un diseño conceptual que después es convertido en diseño lógico, este diseño lógico resultante estará en 3FN siempre que todo el proceso se haya realizado de forma correcta, sirviendo en este caso la teoría de la normalización para comprobar que el diseño ha sido realizado correctamente. Si no lo fuese, podremos aplicar las formas normales para corregir los errores que hubieran podido producirse.

Mientras la normalización resuelve los problemas relacionados con la estructuración de los datos en tablas, crea problemas añadidos a su propio concepto, como son la duplicación de datos y la ineficacia en la recuperación de información.

Así, el proceso de normalización envuelve la descomposición de una tabla en tablas más pequeñas, lo cual requiere que la clave primaria de la tabla original se incluya, como una clave foránea, en la tabla/s que se forman. Esto significa que a medida que se van creando estas claves foráneas se va incrementando las probabilidades de poner en peligro la integridad de la base de datos.

Otro efecto adicional del número creciente de tablas en la base de datos, es que se ve disminuido el rendimiento del sistema en la recuperación de la información contenida. Esta disminución del rendimiento puede ser particularmente importante en

sistemas basados en microordenadores. Por tanto, en ciertas ocasiones es necesario llegar a un compromiso entre el nivel de normalización de la base de datos y el rendimiento del sistema.

## *6.5 Ejercicios de diseño de bases de datos.*

### 6.5.1 Ejercicio.

Una empresa pretende desarrollar una base de datos de empleados y proyectos. La empresa esta estructurada en departamentos, cada uno de los cuales posee uno o varios proyectos, de forma que un proyecto solo depende de un departamento. Por otro lado cada departamento consta de uno o varios empleados, que trabajan de forma exclusiva para ese departamento, pero pueden trabajar simultáneamente en varios proyectos. Cada empleado tiene un jefe encargado de supervisar su trabajo, pudiendo cada jefe supervisar el trabajo de varios empleados. Dada la descripción anterior, desarrollar la base de datos normalizada hasta 3FN.

### 6.5.2 Ejercicio.

Dada el siguiente diseño de una tabla de una base de datos, aplicar las tres primeras formas normales y llevar el diseño a 3FN.

ENVIO

| |
|---|
| ***Codigo_envio*** |
| Matricula_camion |
| Modelo_camion |
| Capacidad_camion |
| Cliente_1 |
| Direccion_cliente_1 |
| Pedido_cliente_1 |
| Articulo_1_pedido_cliente_1 |
| Volumen_articulo_1_pedido_cliente_1 |
| … |
| Articulo_I_pedido_cliente_1 |
| Volumen_articulo_I_pedido_cliente_1 |
| |
| … |
| Cliente_N |
| Direccion_cliente_N |
| Pedido_cliente_N |
| Articulo_1_pedido_cliente_N |
| Volumen_articulo_1_pedido_cliente_N |
| |
| … |
| Articulo_J_pedido_cliente_N |
| Volumen_articulo_J_pedido_cliente_N |