Capítulo 2

Modelo relacional

Introducción y objetivos

En este capítulo se presentan los principios básicos del modelo relacional, que es el modelo de datos en el que se basan la mayoría de los SGBD en uso hoy en día. En primer lugar, se presenta la estructura de datos relacional y a continuación las reglas de integridad que deben cumplirse sobre la misma.

Al finalizar este capítulo, el estudiante debe ser capaz de:

- Definir qué es un modelo de datos y describir cómo se clasifican los modelos de datos.
- Definir los distintos modelos lógicos de bases de datos.
- Definir la estructura de datos relacional y todas sus partes.
- Enumerar las propiedades de las relaciones.
- Definir los tipos de relaciones.
- Definir superclave, clave candidata, clave primaria y clave ajena.
- Definir el concepto de nulo.
- Definir la regla de integridad de entidades y la regla de integridad referencial.
- Definir qué es una regla de negocio.
- Dar un ejemplo completo de una base de datos formada por, al menos, dos relaciones con claves ajenas.

2.1. Modelos de datos

Una de las características fundamentales de los sistemas de bases de datos es que proporcionan cierto nivel de abstracción de datos, al ocultar las características sobre el almacenamiento físico que la mayoría de usuarios no necesita conocer. Los *modelos de datos* son el instrumento principal para ofrecer dicha abstracción a través de su jerarquía de niveles. Un modelo de datos es un conjunto de conceptos que sirven para describir la estructura de una base de datos, es decir, los datos, las relaciones entre los datos y las restricciones que deben cumplirse sobre los datos. Los modelos de datos contienen también un conjunto de operaciones básicas para la realización de consultas (lecturas) y actualizaciones de datos. Además, los modelos de datos más modernos incluyen mecanismos para especificar acciones compensatorias o adicionales que se deben llevar a cabo ante las acciones habituales que se realizan sobre la base de datos.

Los modelos de datos se pueden clasificar dependiendo de los tipos de conceptos que ofrecen para describir la estructura de la base de datos, formando una jerarquía de niveles. Los modelos de datos de alto nivel, o *modelos conceptuales*, disponen de conceptos muy cercanos al modo en que la mayoría de los usuarios percibe los datos, mientras que los modelos de datos de bajo nivel, o *modelos físicos*, proporcionan conceptos que describen los detalles de cómo se almacenan los datos en el ordenador. Los conceptos de los modelos físicos están dirigidos al personal informático, no a los usuarios finales. Entre estos dos extremos se encuentran los *modelos lógicos*, cuyos conceptos pueden ser entendidos por los usuarios finales, aunque no están demasiado alejados de la forma en que los datos se organizan físicamente. Los modelos lógicos ocultan algunos detalles de cómo se almacenan los datos, pero pueden implementarse de manera directa en un SGBD.

Los modelos conceptuales utilizan conceptos como entidades, atributos y relaciones. Una *entidad* representa un objeto o concepto del mundo real como, por ejemplo, un cliente de una empresa o una de sus facturas. Un *atributo* representa alguna propiedad de interés de una entidad como, por ejemplo, el nombre o el domicilio del cliente. Una *relación* describe una interacción entre dos o más entidades, por ejemplo, la relación que hay entre un cliente y las facturas que se le han realizado.

Cada SGBD soporta un modelo lógico, siendo los más comunes el *relacional*, el de *red* y el *jerárquico*. Estos modelos representan los datos valiéndose de estructuras de registros, por lo que también se denominan *modelos orientados a registros*. Hay una familia más moderna de modelos lógicos, son los *modelos orientados a objetos*, que están más próximos a los modelos conceptuales. En el modelo relacional los datos se describen como un conjunto de tablas con referencias lógicas entre ellas, mientras que en los modelos jerárquico y de red, los datos se describen como conjuntos de registros con referencias físicas entre ellos (punteros).

Los modelos físicos describen cómo se almacenan los datos en el ordenador: el formato de los registros, la estructura de los ficheros (desordenados, ordenados, agrupados) y los métodos de acceso utilizados (índices, tablas de dispersión).

A la descripción de una base de datos mediante un modelo de datos se le denomina *esquema de la base de datos*. Este esquema se especifica durante el diseño, y no es de esperar que se modifique a menudo. Sin embargo, los datos que se almacenan en la base de datos pueden cambiar con mucha frecuencia: se insertan datos, se actualizan, se borran, etc. Los datos que la base de datos contiene en un determinado momento conforman el *estado de la base de datos*, o como también se denomina: una *ocurrencia de la base de datos*.

La distinción entre el esquema y el estado de la base de datos es muy importante. Cuando definimos una nueva base de datos, sólo especificamos su esquema al SGBD. En ese momento, el estado de la base de datos es el *estado vacío*, sin datos. Cuando se cargan datos por primera vez, la base datos pasa al *estado inicial*. De ahí en adelante, siempre que se realice una operación de actualización de la base de datos, se tendrá un nuevo estado. El SGBD se encarga, en parte, de garantizar que todos los estados de la base de datos sean estados válidos que satisfagan la estructura y las restricciones especificadas en el esquema. Por lo tanto, es muy importante que el esquema que se especifique al SGBD sea correcto y se debe tener gran cuidado al diseñarlo. El SGBD almacena el esquema en su *catálogo* o *diccionario de datos*, de modo que se pueda consultar siempre que sea necesario.

En 1970, el modo en que se veían las bases de datos cambió por completo cuando E. F. Codd introdujo el modelo relacional. En aquellos momentos, el enfoque existente para la estructura de las bases de datos utilizaba punteros físicos (direcciones de disco) para relacionar registros de distintos ficheros. Si, por ejemplo, se quería relacionar un registro A con un registro B, se debía añadir al registro A un campo conteniendo la dirección en disco (un puntero físico) del registro B. Codd demostró que estas bases de datos limitaban en gran medida los tipos de operaciones que los usuarios podían realizar sobre los datos. Además, estas bases de datos eran muy vulnerables a cambios en el entorno físico. Si se añadían los controladores de un nuevo disco al sistema y los datos se movían de una localización física a otra, se requería una conversión de los ficheros de datos. Estos sistemas se basaban en el modelo de red y el modelo jerárquico, los dos modelos lógicos que constituyeron la primera generación de los SGBD.

El modelo relacional representa la segunda generación de los SGBD. En él, todos los datos están estructurados a nivel lógico como tablas formadas por filas y columnas, aunque a nivel físico pueden tener una estructura completamente distinta. Un punto fuerte del modelo relacional es la sencillez de su estructura lógica. Pero detrás de esa simple estructura hay un fundamento teórico importante del que carecen los SGBD de la primera generación, lo que constituye otro punto a su favor.

Dada la popularidad del modelo relacional, muchos sistemas de la primera generación se han modificado para proporcionar una interfaz de usuario relacional, con independencia del modelo lógico que soportan (de red o jerárquico).

En los últimos años, se han propuesto algunas extensiones al modelo relacional para capturar mejor el significado de los datos, para disponer de los conceptos de la orientación a objetos y para disponer de capacidad deductiva.

El modelo relacional, como todo modelo de datos, tiene que ver con tres aspectos de los datos, que son los que se presentan en los siguientes apartados de este capítulo: qué características tiene la estructura de datos, cómo mantener la integridad de los datos y cómo realizar el manejo de los mismos.

2.2. Estructura de datos relacional

La estructura de datos del modelo relacional es la *relación*. En este apartado se presenta esta estructura de datos, sus propiedades, los tipos de relaciones y qué es una clave de una relación. Para facilitar la comprensión de las definiciones formales de todos estos conceptos, se dan antes unas definiciones informales que permiten asimilar dichos conceptos con otros que resulten familiares.

2.2.1. Relaciones

Definiciones informales

El modelo relacional se basa en el concepto matemático de *relación*, que gráficamente se representa mediante una tabla. Codd, que era un experto matemático, utilizó una terminología perteneciente a las matemáticas, en concreto de la teoría de conjuntos y de la lógica de predicados.

Una relación es una tabla con columnas y filas. Un SGBD sólo necesita que el usuario pueda percibir la base de datos como un conjunto de tablas. Esta percepción sólo se aplica a la estructura lógica de la base de datos, no se aplica a la estructura física de la base de datos, que se puede implementar con distintas estructuras de almacenamiento.

Un atributo es el nombre de una columna de una relación. En el modelo relacional, las relaciones se utilizan para almacenar información sobre los objetos que se representan en la base de datos. Una relación se representa gráficamente como una tabla bidimensional en la que las filas corresponden a registros individuales y las columnas corresponden a los campos o atributos de esos registros. Los atributos pueden aparecer en la relación en cualquier orden.

Por ejemplo, la información de los clientes de una empresa determinada se representa mediante la relación `CLIENTES` de la figura 2.1, que tiene columnas para los atributos `codcli` (código del cliente), `nombre` (nombre y apellidos del cliente), `dirección` (calle y número donde se ubica el cliente), `codpostal` (código postal correspondiente a la dirección del cliente) y `codpue` (código de la población del cliente). La información sobre las poblaciones se representa

mediante la relación PUEBLOS de la misma figura, que tiene columnas para los atributos codpue (código de la población), nombre (nombre de la población) y codpro (código de la provincia en que se encuentra la población).

CLIENTES

codcli	nombre	dirección	codpostal	codpue
333	Sos Carretero, Jesús	Mosen Compte, 14	12964	53596
336	Miguel Archilés, Ramón	Bernardo Mundina, 132-5	12652	07766
342	Pinel Huerta, Vicente	Francisco Sempere, 37-10	12112	07766
345	López Botella, Mauro	Avenida del Puerto, 20-1	12439	12309
348	Palau Martínez, Jorge	Raval de Sant Josep, 97-2	12401	12309
354	Murría Vinaiza, José	Ciudadela, 90-18	12990	12309
357	Huguet Peris, Juan Ángel	Calle Mestre Rodrigo, 7	12930	12309

PUEBLOS

codpue	nombre	codpro
07766	Burriana	12
12309	Castellón	12
17859	Enramona	12
46332	Soneja	12
53596	Vila-real	12

Figura 2.1: Relaciones que almacenan los datos de los clientes y sus poblaciones.

Un dominio es el conjunto de valores legales de uno o varios atributos. Los dominios constituyen una poderosa característica del modelo relacional. Cada atributo de una base de datos relacional se define sobre un dominio, pudiendo haber varios atributos definidos sobre el mismo dominio. La figura 2.2 muestra los dominios de los atributos de la relación CLIENTES.

Atributo	*Dominio*	*Descripción*	*Definición*
codcli	codli_dom	Posibles códigos de cliente.	Número hasta 5 dígitos.
nombre	nombre_dom	Nombres de personas: apellido1 apellido2, nombre.	50 caracteres.
dirección	dirección_dom	Domicilios de España: calle, número.	50 caracteres.
codpostal	codpostal_dom	Códigos postales de España.	5 caracteres.
codpue	codpue_dom	Códigos de las poblaciones de España.	5 caracteres.

Figura 2.2: Dominios de los atributos de la relación que almacena los datos de los clientes.

El concepto de dominio es importante porque permite que el usuario defina, en un lugar común, el significado y la fuente de los valores que los atributos pueden tomar. Esto hace que haya más información disponible para el sistema cuando éste va a ejecutar una operación relacional, de modo que las operaciones que son semánticamente incorrectas, se pueden evitar. Por ejemplo, no tiene sentido comparar el nombre de una calle con un número de teléfono, aunque los dos atributos sean cadenas de caracteres. Sin embargo, el importe mensual del alquiler de un inmueble no estará definido sobre el mismo dominio que

el número de meses que dura el alquiler, pero sí tiene sentido multiplicar los valores de ambos dominios para averiguar el importe total al que asciende el alquiler. Los SGBD relacionales no ofrecen un soporte completo de los dominios ya que su implementación es extremadamente compleja.

Una tupla es una fila de una relación. Los elementos de una relación son las tuplas o filas de la tabla. En la relación `CLIENTES`, cada tupla tiene cinco valores, uno para cada atributo. Las tuplas de una relación no siguen ningún orden.

El grado de una relación es el número de atributos que contiene. La relación `CLIENTES` es de grado cinco porque tiene cinco atributos. Esto quiere decir que cada fila de la tabla es una tupla con cinco valores. El grado de una relación no cambia con frecuencia.

La cardinalidad de una relación es el número de tuplas que contiene. Ya que en las relaciones se van insertando y borrando tuplas a menudo, la cardinalidad de las mismas varía constantemente.

Una base de datos relacional es un conjunto de relaciones normalizadas. Una relación está normalizada si en la intersección de cada fila con cada columna hay un solo valor.

Definiciones formales

Una *relación* R definida sobre un conjunto de *dominios* $D_1, D_2, \ldots, D_n$ consta de:

- *Cabecera:* conjunto fijo de pares *atributo:dominio*

  $$\{(A_1 : D_1), (A_2 : D_2), \ldots, (A_n : D_n)\}$$

  donde cada atributo A_j corresponde a un único dominio D_j y todos los A_j son distintos, es decir, no hay dos atributos que se llamen igual. El grado de la relación R es n.

- *Cuerpo:* conjunto variable de *tuplas*. Cada tupla es un conjunto de pares *atributo:valor*:

  $$\{(A_1 : v_{i1}), (A_2 : v_{i2}), \ldots, (A_n : v_{in})\}$$

  con $i = 1, 2, \ldots, m$, donde m es la cardinalidad de la relación R. En cada par $(A_j : v_{ij})$ se tiene que $v_{ij} \in D_j$.

A continuación se muestra la cabecera de la relación `CLIENTES` de la figura 2.1, y una de sus tuplas.

```
{ (codcli:codcli_dom), (nombre:nombre_dom),
  (dirección:dirección_dom), (codpostal:codpostal_dom),
  (codpue:codpue_dom) }
```

```
{ (codcli:333), (nombre:Sos Carretero, Jesús),
  (dirección:Mosen Compte, 14), (codpostal:12964),
  (codpue:53596) }
```

Este conjunto de pares no está ordenado, por lo que la tupla anterior y la siguiente son la misma:

```
{ (nombre:Sos Carretero, Jesús), (codpostal:12964),
  (codcli:333), (dirección:Mosen Compte, 14),
  (codpue:53596) }
```

Las relaciones se suelen representar gráficamente mediante tablas. Los nombres de las columnas corresponden a los nombres de los atributos, y las filas son cada una de las tuplas de la relación. Los valores que aparecen en cada una de las columnas pertenecen al conjunto de valores del dominio sobre el que está definido el atributo correspondiente.

2.2.2. Propiedades de las relaciones

Las relaciones tienen las siguientes características:

- Cada relación tiene un nombre, y éste es distinto del nombre de todas las demás.
- Los dominios sobre los que se definen los atributos son escalares, por lo que los valores de los atributos son atómicos. De este modo, en cada tupla, cada atributo toma un solo valor. Se dice que las relaciones están *normalizadas*.
- No hay dos atributos que se llamen igual.
- El orden de los atributos no importa: los atributos no están ordenados.
- Cada tupla es distinta de las demás: no hay tuplas duplicadas.
- El orden de las tuplas no importa: las tuplas no están ordenadas.

2.2.3. Tipos de relaciones

En un SGBD relacional hay dos tipos de relaciones:

- *Relaciones base.* Son relaciones reales que tienen nombre, y forman parte directa de la base de datos almacenada. Se dice que las relaciones base son relaciones autónomas.
- *Vistas.* También denominadas relaciones virtuales, son relaciones con nombre y derivadas (no autónomas). Que son derivadas significa que se obtienen a partir de otras relaciones; se representan mediante su definición en términos de esas otras relaciones. Las vistas no poseen datos almacenados propios, los datos que contienen corresponden a datos almacenados en relaciones base.

2.2.4. Claves

Ya que en una relación no hay tuplas repetidas, éstas se pueden distinguir unas de otras, es decir, se pueden identificar de modo único. La forma de identificarlas es mediante los valores de sus atributos. Se denomina *superclave* a un atributo o conjunto de atributos que identifican de modo único las tuplas de una relación. Se denomina *clave candidata* a una superclave en la que ninguno de sus subconjuntos es una superclave de la relación. El atributo o conjunto de atributos K de la relación R es una clave candidata para R si, y sólo si, satisface las siguientes propiedades:

- *Unicidad:* nunca hay dos tuplas en la relación R con el mismo valor de K.
- *Irreducibilidad (minimalidad):* ningún subconjunto de K tiene la propiedad de unicidad, es decir, no se pueden eliminar componentes de K sin destruir la unicidad.

Cuando una clave candidata está formada por más de un atributo, se dice que es una *clave compuesta*. Una relación puede tener varias claves candidatas. Por ejemplo, en la relación `PUEBLOS` de la figura 2.1, el atributo `nombre` no es una clave candidata ya que hay pueblos en España con el mismo nombre que se encuentran en distintas provincias. Sin embargo, se ha asignado un código único a cada población, por lo que el atributo `codpue` sí es una clave candidata de la relación `PUEBLOS`. También es una clave candidata de esta relación la pareja formada por los atributos `nombre` y `codpro`, ya que no hay dos poblaciones en la misma provincia que tengan el mismo nombre.

Para identificar las claves candidatas de una relación no hay que fijarse en un estado u ocurrencia de la base de datos. El hecho de que en un momento dado no haya duplicados para un atributo o conjunto de atributos, no garantiza que los duplicados no sean posibles. Sin embargo, la presencia de duplicados

en un estado de la base de datos sí es útil para demostrar que cierta combinación de atributos no es una clave candidata. El único modo de identificar las claves candidatas es conociendo el significado real de los atributos, ya que esto permite saber si es posible que aparezcan duplicados. Sólo usando esta información semántica se puede saber con certeza si un conjunto de atributos forman una clave candidata. Por ejemplo, viendo la ocurrencia anterior de la relación `CLIENTES` se podría pensar que el atributo `nombre` es una clave candidata. Pero ya que este atributo es el nombre de un cliente y es posible que haya dos clientes con el mismo nombre, el atributo no es una clave candidata.

Se denomina *clave primaria* de una relación a aquella clave candidata que se escoge para identificar sus tuplas de modo único. Ya que una relación no tiene tuplas duplicadas, siempre hay una clave candidata y, por lo tanto, la relación siempre tiene clave primaria. En el peor caso, la clave primaria estará formada por todos los atributos de la relación, pero normalmente habrá un pequeño subconjunto de los atributos que haga esta función.

Las claves candidatas que no son escogidas como clave primaria son denominadas *claves alternativas*. Por ejemplo, la clave primaria de la relación `PUEBLOS` es el atributo `codpue`, siendo la pareja formada por `nombre` y `codpro` una clave alternativa. En la relación `CLIENTES` sólo hay una clave candidata que es el atributo `codcli`, por lo que esta clave candidata es la clave primaria.

Una *clave ajena* es un atributo o un conjunto de atributos de una relación cuyos valores coinciden con los valores de la clave primaria de alguna otra relación (puede ser la misma). Las claves ajenas representan *relaciones entre datos*. Por ejemplo, el atributo `codpue` de `CLIENTES` relaciona a cada cliente con su población. Este atributo en `CLIENTES` es una clave ajena cuyos valores hacen referencia al atributo `codpue` de `PUEBLOS` (su clave primaria). Se dice que un valor de clave ajena representa una *referencia* a la tupla que contiene el mismo valor en su clave primaria (*tupla referenciada*).

Si nos fijamos en los datos de la figura 2.1, para conocer el nombre de la población del cliente con `codcli` = `333`, debemos seguir la clave ajena `codpue` que aparece en la tupla de dicho cliente y que tiene el valor `53596`. Seguir la referencia que implica la clave ajena conlleva visitar la relación `PUEBLOS` y localizar la fila que tiene el valor `53596` en su clave primaria. Nótese que, en este ejemplo, la clave ajena tiene el mismo nombre que la clave primaria a la que hace referencia. Esto no es un requisito, las claves ajenas no precisan tener el mismo nombre que la clave primaria a la que referencian; sin embargo, si se utilizan los mismos nombres (o nombres compuestos derivados de los mismos) es más fácil reconocer las claves ajenas.

Al hablar de claves primarias y de claves ajenas es importante darse cuenta de que los valores de una clave primaria no se pueden repetir, mientras que no sucede lo mismo con las claves ajenas que le hacen referencia. Así, en las tablas de la figura 2.1 no es posible encontrar dos tuplas con el mismo valor en `PUEBLOS.codpue` (cada población debe aparecer en la relación una sola vez), pero sí es posible encontrar varias tuplas con el mismo valor en

CLIENTES.codpue, ya que es posible que haya varios clientes que se ubiquen en la misma población.

2.3. Esquema de una base de datos relacional

Una base de datos relacional es un conjunto de relaciones. Para representar el esquema de una base de datos relacional se debe dar el nombre de sus relaciones, los atributos de éstas, los dominios sobre los que se definen estos atributos, las claves primarias y las claves ajenas.

El esquema de la base de datos de la empresa con la que trabajaremos en este libro es el siguiente:

```
  CLIENTES(codcli, nombre, dirección, codpostal, codpue)
VENDEDORES(codven, nombre, dirección, codpostal, codpue, codjefe)
   PUEBLOS(codpue, nombre, codpro)
PROVINCIAS(codpro, nombre)
 ARTÍCULOS(codart, descrip, precio, stock, stock_min, dto)
  FACTURAS(codfac, fecha, codcli, codven, iva, dto)
LÍNEAS_FAC(codfac, línea, cant, codart, precio, dto)
```

En el esquema anterior, los nombres de las relaciones aparecen seguidos de los nombres de los atributos encerrados entre paréntesis. Las claves primarias son los atributos subrayados. Las claves ajenas se representan mediante los siguientes *diagramas referenciales:*

CLIENTES	$\xrightarrow{\text{codpue}}$	PUEBLOS	:	Población del cliente.
VENDEDORES	$\xrightarrow{\text{codpue}}$	PUEBLOS	:	Población del vendedor.
VENDEDORES	$\xrightarrow{\text{codjefe}}$	VENDEDORES	:	Jefe del vendedor.
PUEBLOS	$\xrightarrow{\text{codpro}}$	PROVINCIAS	:	Provincia en la que se encuentra la población.
FACTURAS	$\xrightarrow{\text{codcli}}$	CLIENTES	:	Cliente al que pertenece la factura.
FACTURAS	$\xrightarrow{\text{codven}}$	VENDEDORES	:	Vendedor que ha realizado la venta.
LÍNEAS_FAC	$\xrightarrow{\text{codfac}}$	FACTURAS	:	Factura en la que se encuentra la línea.
LÍNEAS_FAC	$\xrightarrow{\text{codart}}$	ARTÍCULOS	:	Artículo que se compra en la línea de factura.

La tabla PROVINCIAS almacena información sobre las provincias de España. De cada provincia se almacena su nombre (nombre) y un código que la identifica (codpro). La tabla PUEBLOS contiene los nombres (nombre) de los pueblos de España. Cada pueblo se identifica por un código que es único (codpue) y tiene una referencia a la provincia a la que pertenece (codpro). La tabla CLIENTES contiene los datos de los clientes: código que identifica a

cada uno (codcli), nombre y apellidos (nombre), calle y número (dirección), código postal (codpostal) y una referencia a su población (codpue). La tabla VENDEDORES contiene los datos de los vendedores de la empresa: código que identifica a cada uno (codven), nombre y apellidos (nombre), calle y número (dirección), código postal (codpostal), una referencia a su población (codpue) y una referencia al vendedor del que depende (codjefe), si es el caso. En la tabla ARTÍCULOS se tiene el código que identifica a cada artículo (codart), su descripción (descrip), el precio de venta actual (precio), el número de unidades del artículo que hay en el almacén (stock), la cantidad mínima que se desea mantener almacenada (stock_min) y, si el artículo está en oferta, el descuento (dto) que se debe aplicar cuando se venda. La tabla FACTURAS contiene las cabeceras de las facturas correspondientes a las compras realizadas por los clientes. Cada factura tiene un código único (codfac), la fecha en que se ha realizado (fecha), así como el IVA (iva) y el descuento que se le ha aplicado (dto). Cada factura hace referencia al cliente al que pertenece (codcli) y al vendedor que la ha realizado (codven). Las líneas de cada factura se encuentran en la tabla LÍNEAS_FAC, identificándose cada una por el número de línea que ocupa dentro de la factura (codfac, línea). En cada una de ellas se especifica la cantidad de unidades (cant) del artículo que se compra (codart), el precio de venta por unidad (precio) y el descuento que se aplica sobre dicho precio (dto), si es que el artículo estaba en oferta cuando se vendió.

A continuación se muestra un estado de la base de datos cuyo esquema se acaba de definir.

CLIENTES

codcli	nombre	dirección	codpostal	codpue
333	Sos Carretero, Jesús	Mosen Compte, 14	12964	53596
336	Miguel Archilés, Ramón	Bernardo Mundina, 132-5	12652	07766
342	Pinel Huerta, Vicente	Francisco Sempere, 37-10	12112	07766
345	López Botella, Mauro	Avenida del Puerto, 20-1	12010	12309
348	Palau Martínez, Jorge	Raval de Sant Josep, 97-2	12003	12309
354	Murría Vinaiza, José	Ciudadela, 90-18	12003	12309
357	Huguet Peris, Juan Ángel	Calle Mestre Rodrigo, 7	12100	12309

VENDEDORES

codven	nombre	dirección	codpostal	codpue	codjefe
5	Guillén Vilar, Natalia	Sant Josep, 110	12597	53596	105
105	Poy Omella, Paloma	Sanchis Tarazona, 103-1	12257	46332	
155	Rubert Cano, Diego	Benicarló Residencial, 154	12425	17859	5
455	Agost Tirado, Jorge	Pasaje Peñagolosa, 21-19	12914	53596	5

PUEBLOS

codpue	nombre	codpro
07766	Burriana	12
12309	Castellón	12
17859	Enramona	12
46332	Soneja	12
53596	Vila-real	12

PROVINCIAS

codpro	nombre
03	Alicante
12	Castellón
46	Valencia

ARTÍCULOS

codart	descrip	precio	stock	stock_min	dto
IM3P32V	Interruptor magnetotérmico 4p, 2	27.01	1	1	
im4P10L	Interruptor magnetotérmico 4p, 4	32.60	1	1	15
L14340	Bases de fusibles cuchillas T0	0.51	3	3	
L17055	Bases de fusible cuchillas T3	7.99	3	3	
L76424	Placa 2 E. legrand serie mosaic	2.90	5	2	
L85459	Tecla legrand marfil	2.80	0	4	
L85546	Tecla difusores legrand bronce	1.05	13	5	5
L92119	Portalámparas 14 curvo	5.98	2	1	
ME200	Marco Bjc Ibiza 2 elementos	13.52	1	1	
N5072	Pulsador luz piloto Niessen trazo	1.33	11	2	
N8017BA	Reloj Orbis con reserva de cuerda	3.40	7	4	
P605	Caja 1 elem. plastimetal	1.65	16	9	
P695	Interruptor rotura brusca 100 A M	13.22	1	1	
P924	Interruptor marrón dec. con visor	2.39	8	3	
REF1X20	Regleta fluorescente 1x36 bajo F	8.71	1	1	
S3165136	Bloque emergencia Satf 150 L	4.81	6	3	
T4501	Tubo empotrar 100	2.98	0	5	
TE7200	Doble conmutador Bjc Ibiza blanco	13.22	1	1	
TFM16	Curva tubo hierro 11	0.33	23	13	
TH11	Curva tubo hierro 29	1.42	20	3	
THC21	Placa mural Felmax	1.56	1	1	
ZNCL	Base T,t lateral Ticino S, Tekne	41.71	1	1	10

FACTURAS

codfac	fecha	codcli	codven	iva	dto
6643	16/07/2010	333	105	18	10
6645	16/07/2010	336	105	0	20
6654	31/07/2010	357	155	8	0
6659	08/08/2010	342	5	0	0
6680	10/09/2010	348	455	8	0
6723	06/11/2010	342	5	18	0
6742	17/12/2010	333	105	8	20

LÍNEAS_FAC

codfac	linea	cant	codart	precio	dto
6643	1	6	L14340	0.51	20
6643	2	1	N5072	1.33	0
6643	3	2	P695	13.22	0
6645	1	10	ZNCL	41.71	0
6645	2	6	N8017BA	3.40	0
6645	3	3	TE7200	13.22	0
6645	4	4	L92119	5.98	0
6654	1	6	REF1X20	8.71	50
6659	1	8	THC21	1.56	0
6659	2	12	L17055	7.99	25
6659	3	9	L76424	2.90	0
6680	1	12	T4501	2.98	0
6680	2	11	im4P10L	32.60	0
6723	1	5	L85459	2.80	5
6742	1	9	ME200	13.52	0
6742	2	8	S3165136	4.81	5

2.4. Reglas de integridad

Una vez definida la estructura de datos del modelo relacional, pasamos a estudiar las reglas de integridad que los datos almacenados en dicha estructura deben cumplir para garantizar que son correctos.

Al definir cada atributo sobre un dominio se impone una restricción sobre el conjunto de valores permitidos para cada atributo. A este tipo de restricciones se les denomina *restricciones de dominios*. Hay además dos reglas de integridad muy importantes que son restricciones que se deben cumplir en todas las bases de datos relacionales y en todos sus estados (las reglas se deben cumplir todo el tiempo). Estas reglas son la *regla de integridad de entidades* y la *regla de integridad referencial*. Antes de definirlas, es preciso conocer el concepto de *nulo*.

2.4.1. Nulos

Cuando en una tupla un atributo es desconocido, se dice que es *nulo*. Un nulo no representa el valor cero ni la cadena vacía ya que éstos son valores que tienen significado. El nulo implica ausencia de información, bien porque al insertar la tupla se desconocía el valor del atributo, o bien porque para dicha tupla el atributo no tiene sentido.

Ya que los nulos no son valores, deben tratarse de modo diferente, lo que causa problemas de implementación. De hecho, no todos los SGBD relacionales soportan los nulos.

2.4.2. Regla de integridad de entidades

La primera regla de integridad se aplica a las claves primarias de las relaciones base: *ninguno de los atributos que componen la clave primaria puede ser nulo.*

Por definición, una clave primaria es una clave irreducible que se utiliza para identificar de modo único las tuplas. Que es irreducible significa que ningún subconjunto de la clave primaria sirve para identificar las tuplas de modo único. Si se permitiera que parte de la clave primaria fuera nula, se estaría diciendo que no todos sus atributos son necesarios para distinguir las tuplas, con lo que se estaría contradiciendo la irreducibilidad.

Nótese que esta regla sólo se aplica a las relaciones base y a las claves primarias, no a las claves alternativas.

2.4.3. Regla de integridad referencial

La segunda regla de integridad se aplica a las claves ajenas: *si en una relación hay alguna clave ajena, sus valores deben coincidir con valores de la clave primaria a la que hace referencia, o bien, deben ser completamente nulos.*

En la base de datos presentada en el apartado anterior hay ocho claves ajenas que mantienen relacionada la información almacenada en las tablas. Así, a través de la clave ajena `FACTURAS.codcli` se puede conocer los datos personales del cliente al que pertenece una determinada factura buscando en la relación `CLIENTES` la tupla en cuya clave primaria aparece el valor de `codcli` al que se hace referencia en la factura. El nombre de la población del cliente se podrá conocer siguiendo la clave ajena `CLIENTES.codpue` y, una vez localizada la población con dicho `codpue` en `PUEBLOS`, se podrá acceder al nombre de su provincia siguiendo la clave ajena `PUEBLOS.codpro`.

Pues bien, la regla de integridad referencial exige que los valores que aparecen en la clave ajena `FACTURAS.codcli` aparezcan como clave primaria en `CLIENTES`. De ese modo, todas las facturas corresponderán a clientes cuyos datos se encuentran en la base de datos. Del mismo modo, la regla exige que los valores de `CLIENTES.codpue` aparezcan en la clave primaria de `PUEBLOS` y que los valores de `PUEBLOS.codpro` aparezcan en la clave primaria de `PROVINCIAS`.

La regla de integridad referencial se enmarca en términos de estados de la base de datos: indica lo que es un estado ilegal, pero no dice cómo puede evitarse. Por lo tanto, una vez establecida la regla, hay que plantearse qué hacer si estando en un estado legal, llega una petición para realizar una operación que conduce a un estado ilegal. Existen dos opciones: *rechazar* o *aceptar* la operación y realizar operaciones adicionales compensatorias que conduzcan a un estado legal.

Para hacer respetar la integridad referencial se debe contestar, para cada clave ajena, a las tres preguntas que se plantean a continuación y que determinarán su comportamiento:

- *Regla de los nulos:* «¿Tiene sentido que la clave ajena acepte nulos?»
- *Regla de borrado:* «¿Qué ocurre si se intenta borrar la tupla referenciada por la clave ajena?»
    - *Restringir:* no se permite borrar la tupla referenciada.
    - *Propagar:* se borra la tupla referenciada y se propaga el borrado a las tuplas que la referencian mediante la clave ajena.
    - *Anular:* se borra la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).
    - *Valor por defecto:* se borra la tupla referenciada y las tuplas que la referenciaban ponen en la clave ajena el valor por defecto establecido para la misma.

- *Regla de modificación:* «¿Qué ocurre si se intenta modificar el valor de la clave primaria de la tupla referenciada por la clave ajena?»
  - *Restringir:* no se permite modificar el valor de la clave primaria de la tupla referenciada.
  - *Propagar:* se modifica el valor de la clave primaria de la tupla referenciada y se propaga la modificación a las tuplas que la referencian, mediante la clave ajena.
  - *Anular:* se modifica la tupla referenciada y las tuplas que la referenciaban ponen a nulo la clave ajena (sólo si acepta nulos).
  - *Valor por defecto:* se modifica la tupla referenciada y las tuplas que la referenciaban ponen en la clave ajena el valor por defecto establecido para la misma.

Así, en el caso del esquema de la base de datos presentada en el apartado anterior, deberemos determinar las reglas de comportamiento para cada clave ajena. Por ejemplo, para la clave ajena `FACTURAS.codcli` se ha escogido el siguiente comportamiento:

- *Regla de los nulos:* la clave ajena acepta nulos, por lo que es posible encontrar facturas cuyo cliente se ignore (esto se ha decidido así porque lo impone un requisito del usuario).
- *Regla de borrado:* anular. Cuando se elimine un cliente de la base de datos y se proceda a borrarlo de la relación `CLIENTES`, se deberán anular todas las referencias que hubiera desde `FACTURAS.codcli`. De este modo, todas las facturas que tenía ese cliente pasarán a tener un nulo en el código del cliente.
- *Regla de modificación:* propagar. En caso de que se modifique el código a un cliente (quizá porque el sistema de codificación se cambie por parte de la empresa), todas las facturas de dicho cliente actualizarán el valor de `FACTURAS.codcli` para continuar haciendo referencia a la misma tupla.

Del mismo modo, se deberá escoger reglas para el resto de las claves ajenas de la base de datos. Una vez establecidas todas las reglas, el sistema se comportará de manera coherente obedeciendo a todas las reglas impuestas. Por ejemplo, si la regla de borrado para `LÍNEAS_FAC.codfac` es propagar y la regla de borrado para `FACTURAS.codven` es restringir, cuando se borre una tupla en `FACTURAS` se propagará el borrado a `LÍNEAS_FAC`, y se borrarán todas las líneas de la factura referenciada. Sin embargo, cuando se intente borrar la tupla de un vendedor que aparezca en alguna factura, la regla impuesta sobre `FACTURAS.codven` rechazará el borrado del vendedor y no se procederá al borrado ni de sus facturas, ni de las líneas de factura de éstas. Lo que sí será posible es el borrado de un vendedor cuyo código no aparezca en ninguna factura.

2.4.4. Reglas de negocio

Además de las dos reglas de integridad anteriores, es posible que sea necesario imponer ciertas restricciones específicas sobre los datos que forman parte de la estrategia de funcionamiento de la empresa. A estas reglas se las denomina *reglas de negocio*.

Por ejemplo, si en cada oficina de una determinada empresa sólo puede haber hasta veinte empleados, el SGBD debe dar la posibilidad al usuario de definir una regla al respecto y debe hacerla respetar. En este caso, no debería permitir dar de alta a un empleado en una oficina que ya tiene los veinte permitidos. No todos los SGBD relacionales permiten definir este tipo de restricciones y hacerlas respetar.