usuario a las bases de datos. Las organizaciones convirtieron muchas de sus interfaces telefónicas a las bases de datos en interfaces Web, y pusieron disponibles en línea muchos servicios. Por ejemplo, cuando se accede a una tienda de libros en línea y se busca un libro o una colección de música se está accediendo a datos almacenados en una base de datos. Cuando se solicita un pedido en línea, el pedido se almacena en una base de datos. Cuando se accede a un banco en un sitio Web y se consulta el estado de la cuenta y los movimientos, la información se recupera del sistema de bases de datos del banco. Cuando se accede a un sitio Web, la información personal puede ser recuperada de una base de datos para seleccionar los anuncios que se deberían mostrar. Más aún, los datos sobre los accesos Web pueden ser almacenados en una base de datos.

Así, aunque las interfaces de datos ocultan detalles del acceso a las bases de datos, y la mayoría de la gente ni siquiera es consciente de que están interactuando con una base de datos, el acceso a las bases de datos forma una parte esencial de la vida de casi todas las personas actualmente.

La importancia de los sistemas de bases de datos se puede juzgar de otra forma: actualmente, los vendedores de sistemas de bases de datos como Oracle están entre las mayores compañías software en el mundo, y los sistemas de bases de datos forman una parte importante de la línea de productos de compañías más diversificadas, como Microsoft e IBM.

1.2. SISTEMAS DE BASES DE DATOS FRENTE A SISTEMAS DE ARCHIVOS

Considérese parte de una empresa de cajas de ahorros que mantiene información acerca de todos los clientes y cuentas de ahorros. Una manera de mantener la información en un computador es almacenarla en archivos del sistema operativo. Para permitir a los usuarios manipular la información, el sistema tiene un número de programas de aplicación que manipula los archivos, incluyendo:

- Un programa para efectuar cargos o abonos en una cuenta.
- Un programa para añadir una cuenta nueva.
- Un programa para calcular el saldo de una cuenta.
- Un programa para generar las operaciones mensuales.

Estos programas de aplicación se han escrito por programadores de sistemas en respuesta a las necesidades de la organización bancaria.

Si las necesidades se incrementan, se añaden nuevos programas de aplicación al sistema. Por ejemplo, supóngase que las regulaciones de un nuevo gobierno permiten a las cajas de ahorros ofrecer cuentas corrientes. Como resultado se crean nuevos archivos permanentes que contengan información acerca de todas las cuentas corrientes mantenidas por el banco, y puede ser necesario escribir nuevos programas de aplicación para tratar situaciones que no existían en las cuentas de ahorro, tales como manejar descubiertos. Así, sobre la marcha, se añaden más archivos y programas de aplicación al sistema.

Este **sistema de procesamiento de archivos** típico que se acaba de describir se mantiene mediante un sistema operativo convencional. Los registros permanentes son almacenados en varios archivos y se escriben diferentes programas de aplicación para extraer registros y para añadir registros a los archivos adecuados. Antes de la llegada de los sistemas de gestión de bases de datos (SGBDs), las organizaciones normalmente han almacenado la información usando tales sistemas.

Mantener información de la organización en un sistema de procesamiento de archivos tiene una serie de inconvenientes importantes:

- **Redundancia e inconsistencia de datos.** Debido a que los archivos y programas de aplicación son creados por diferentes programadores en un largo período de tiempo, los diversos archivos tienen probablemente diferentes formatos y los programas pueden estar escritos en diferentes lenguajes. Más aún, la misma información puede estar duplicada en diferentes lugares (archivos). Por ejemplo, la dirección y número de teléfono de un cliente particular puede aparecer en un archivo que contenga registros de cuentas de ahorros y en un archivo que contenga registros de una cuenta corriente. Esta redundancia conduce a un almacenamiento y coste de acceso más altos. Además, puede conducir a **inconsistencia de datos;** es decir, las diversas copias de los mismos datos pueden no coincidir. Por ejemplo, un cambio en la dirección del cliente puede estar reflejado en los registros de las cuentas de ahorro pero no estarlo en el resto del sistema.
- **Dificultad en el acceso a los datos.** Supóngase que uno de los empleados del banco necesita averiguar los nombres de todos los clientes que viven en el distrito postal 28733 de la ciudad. El empleado pide al departamento de procesamiento de datos que genere dicha lista. Debido a que esta petición no fue prevista cuando el sistema original fue diseñado, no hay un programa de aplicación a mano para satisfacerla. Hay, sin embargo, un programa de aplicación que genera la lista de *todos* los clientes. El empleado del banco tiene ahora dos opciones: bien obtener la lista de todos los clientes y obtener la información que necesita manualmente, o bien pedir al departamento de procesamiento de datos que haga

que un programador de sistemas escriba el programa de aplicación necesario. Ambas alternativas son obviamente insatisfactorias. Supóngase que se escribe tal programa y que, varios días más tarde, el mismo empleado necesita arreglar esa lista para incluir sólo aquellos clientes que tienen una cuenta con saldo de 10.000 € o más. Como se puede esperar, un programa para generar tal lista no existe. De nuevo, el empleado tiene que elegir entre dos opciones, ninguna de las cuales es satisfactoria.

La cuestión aquí es que el entorno de procesamiento de archivos convencional no permite que los datos necesarios sean obtenidos de una forma práctica y eficiente. Se deben desarrollar sistemas de recuperación de datos más interesantes para un uso general.

- **Aislamiento de datos.** Debido a que los datos están dispersos en varios archivos, y los archivos pueden estar en diferentes formatos, es difícil escribir nuevos programas de aplicación para recuperar los datos apropiados.
- **Problemas de integridad.** Los valores de los datos almacenados en la base de datos deben satisfacer ciertos tipos de **restricciones de consistencia.** Por ejemplo, el saldo de una cuenta bancaria no puede nunca ser más bajo de una cantidad predeterminada (por ejemplo 25 €). Los desarrolladores hacen cumplir esas restricciones en el sistema añadiendo el código apropiado en los diversos programas de aplicación. Sin embargo, cuando se añaden nuevas restricciones, es difícil cambiar los programas para hacer que se cumplan. El problema es complicado cuando las restricciones implican diferentes elementos de datos de diferentes archivos.
- **Problemas de atomicidad.** Un sistema de un computador, como cualquier otro dispositivo mecánico o eléctrico, está sujeto a fallo. En muchas aplicaciones es crucial asegurar que, una vez que un fallo ha ocurrido y se ha detectado, los datos se restauran al estado de consistencia que existía antes del fallo. Consideremos un programa para transferir 50 € desde la cuenta A a la B. Si ocurre un fallo del sistema durante la ejecución del programa, es posible que los 50 € fueron eliminados de la cuenta A pero no abonados a la cuenta B, resultando un estado de la base de datos inconsistente. Claramente, es esencial para la consistencia de la base de datos que ambos, el abono y el cargo tengan lugar, o que ninguno tenga lugar. Es decir, la transferencia de fondos debe ser *atómica*: ésta debe ocurrir en ellos por completo o no ocurrir en absoluto. Es difícil asegurar esta propiedad en un sistema de procesamiento de archivos convencional.
- **Anomalías en el acceso concurrente.** Conforme se ha ido mejorando el conjunto de ejecución de los sistemas y ha sido posible una respuesta en tiempo más rápida, muchos sistemas han ido permitiendo a múltiples usuarios actualizar los datos simultáneamente. En tales sistemas un entorno de interacción de actualizaciones concurrentes puede dar lugar a datos inconsistentes. Considérese una cuenta bancaria A, que contiene 500 €. Si dos clientes retiran fondos (por ejemplo 50 € y 100 € respectivamente) de la cuenta A en aproximadamente el mismo tiempo, el resultado de las ejecuciones concurrentes puede dejar la cuenta en un estado incorrecto (o inconsistente). Supongamos que los programas se ejecutan para cada retirada y escriben el resultado después. Si los dos programas funcionan concurrentemente, pueden leer ambos el valor 500 €, y escribir después 450 € y 400 €, respectivamente. Dependiendo de cuál escriba el último valor, la cuenta puede contener bien 450 € o bien 400 €, en lugar del valor correcto, 350 €. Para protegerse contra esta posibilidad, el sistema debe mantener alguna forma de supervisión. Sin embargo, ya que se puede acceder a los datos desde muchos programas de aplicación diferentes que no han sido previamente coordinados, la supervisión es difícil de proporcionar.
- **Problemas de seguridad.** No todos los usuarios de un sistema de bases de datos deberían poder acceder a todos los datos. Por ejemplo, en un sistema bancario, el personal de nóminas necesita ver sólo esa parte de la base de datos que tiene información acerca de varios empleados del banco. No necesitan acceder a la información acerca de las cuentas de clientes. Como los programas de aplicación se añaden al sistema de una forma ad hoc, es difícil garantizar tales restricciones de seguridad.

Estas dificultades, entre otras, han motivado el desarrollo de los sistemas de bases de datos. En este libro se verán los conceptos y algoritmos que han sido incluidos en los sistemas de bases de datos para resolver los problemas mencionados anteriormente. En la mayor parte de este libro se usa una empresa bancaria como el ejemplo de una aplicación corriente de procesamiento de datos típica encontrada en una empresa.

1.3. VISIÓN DE LOS DATOS

Un sistema de bases de datos es una colección de archivos interrelacionados y un conjunto de programas que permitan a los usuarios acceder y modificar estos archivos. Uno de los propósitos principales de un sistema de bases de datos es proporcionar a los usuarios una visión *abstracta* de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos.

CAPÍTULO

2 MODELO ENTIDAD-RELACIÓN

EL modelo de datos **entidad-relación (E-R)** está basado en una percepción del mundo real consistente en objetos básicos llamados *entidades* y de *relaciones* entre estos objetos. Se desarrolló para facilitar el diseño de bases de datos permitiendo la especificación de un *esquema de la empresa* que representa la estructura lógica completa de una base de datos. El modelo de datos E-R es uno de los diferentes modelos de datos semánticos; el aspecto semántico del modelo yace en la representación del significado de los datos. El modelo E-R es extremadamente útil para hacer corresponder los significados e interacciones de las empresas del mundo real con un esquema conceptual. Debido a esta utilidad, muchas herramientas de diseño de bases de datos se basan en los conceptos del modelo E-R.

2.1. CONCEPTOS BÁSICOS

Hay tres nociones básicas que emplea el modelo de datos E-R: conjuntos de entidades, conjuntos de relaciones y atributos.

2.1.1. Conjuntos de entidades

Una **entidad** es una «cosa» u «objeto» en el mundo real que es distinguible de todos los demás objetos. Por ejemplo, cada persona en un desarrollo es una entidad. Una entidad tiene un conjunto de propiedades, y los valores para algún conjunto de propiedades pueden identificar una entidad de forma unívoca. Por ejemplo, el D.N.I. 67.789.901 identifica unívocamente una persona particular en la empresa. Análogamente, se puede pensar en los préstamos bancarios como entidades, y un número de préstamo P-15 en la sucursal de Castellana identifica unívocamente una entidad de préstamo. Una entidad puede ser concreta, como una persona o un libro, o puede ser abstracta, como un préstamo, unas vacaciones o un concepto.

Un **conjunto de entidades** es un conjunto de entidades del mismo tipo que comparten las mismas propiedades, o atributos. El conjunto de todas las personas que son clientes en un banco dado, por ejemplo, se pueden definir como el conjunto de entidades *cliente*. Análogamente, el conjunto de entidades *préstamo* podría representar el conjunto de todos los préstamos concedidos por un banco particular. Las entidades individuales que constituyen un conjunto se llaman la *extensión* del conjunto de entidades. Así, todos los clientes de un banco son la extensión del conjunto de entidades *cliente*.

Los conjuntos de entidades no son necesariamente disjuntos. Por ejemplo, es posible definir el conjunto de entidades de todos los empleados de un banco (*empleado*) y el conjunto de entidades de todos los clientes del banco (*cliente*). Una entidad *persona* puede ser una entidad *empleado*, una entidad *cliente*, ambas cosas, o ninguna.

Una entidad se representa mediante un conjunto de **atributos**. Los atributos describen propiedades que posee cada miembro de un conjunto de entidades. La designación de un atributo para un conjunto de entidades expresa que la base de datos almacena información similar concerniente a cada entidad del conjunto de entidades; sin embargo, cada entidad puede tener su propio valor para cada atributo. Posibles atributos del conjunto de entidades *cliente* son *id-cliente*, *nombre-cliente*, *calle-cliente* y *ciudad-cliente*. En la vida real, habría más atributos, tales como el número de la calle, el número del portal, la provincia, el código postal, y la comunidad autónoma, pero no se incluyen en el ejemplo simple. Posibles atributos del conjunto de entidades *préstamo* son *número-préstamo* e *importe*.

Cada entidad tiene un **valor** para cada uno de sus atributos. Por ejemplo, una entidad *cliente* en concreto puede tener el valor 32.112.312 para *id-cliente*, el valor Santos para *nombre-cliente*, el valor Mayor para *calle-cliente* y el valor Peguerinos para *ciudad-cliente*.

El atributo *id-cliente* se usa para identificar unívocamente a los clientes, dado que no hay más de un cliente con el mismo nombre, calle y ciudad. En los Estados Unidos, muchas empresas encuentran conveniente usar el número *seguridad-social* de una persona[1] como un

[1] En España se asigna a cada persona del país un número único, denominado número del documento nacional de identidad (D.N.I.) para identificarla unívocamente. Se supone que cada persona tiene un único D.N.I., y no hay dos personas con el mismo D.N.I.

atributo cuyo valor identifica unívocamente a la persona. En general la empresa tendría que crear y asignar un identificador a cada cliente.

Para cada atributo hay un conjunto de valores permitidos, llamados el **dominio**, o el **conjunto de valores**, de ese atributo. El dominio del atributo *nombre-cliente* podría ser el conjunto de todas las cadenas de texto de una cierta longitud. Análogamente, el dominio del atributo *número-préstamo* podría ser el conjunto de todas las cadenas de la forma «P-*n*», donde *n* es un entero positivo.

Una base de datos incluye así una colección de conjuntos de entidades, cada una de las cuales contiene un número de entidades del mismo tipo. En la Figura 2.1 se muestra parte de una base de datos de un banco que consta de dos conjuntos de entidades, *cliente* y *préstamo*.

Formalmente, un atributo de un conjunto de entidades es una función que asigna al conjunto de entidades un dominio. Como un conjunto de entidades puede tener diferentes atributos, cada entidad se puede describir como un conjunto de pares (atributo,valor), un par para cada atributo del conjunto de entidades. Por ejemplo, una entidad concreta *cliente* se puede describir mediante el conjunto {(*id-cliente*, 67.789.901), (*nombre-cliente*, López), (*calle-cliente*, Mayor), (*ciudad-cliente*, Peguerinos)}, queriendo decir que la entidad describe una persona llamada López que tiene D.N.I. número 67.789.901, y reside en la calle Mayor en Peguerinos. Se puede ver, en este punto, que existe una integración del esquema abstracto con el desarrollo real de la empresa que se está modelando. Los valores de los atributos que describen una entidad constituirán una porción significante de los datos almacenados en la base de datos.

Un atributo, como se usa en el modelo E-R, se puede caracterizar por los siguientes tipos de atributo.

- Atributos **simples** y **compuestos**. En los ejemplos considerados hasta ahora, los atributos han sido simples; es decir, no están divididos en subpartes. Los atributos compuestos, en cambio, se pueden dividir en subpartes (es decir, en otros atributos). Por ejemplo, *nombre-cliente* podría estar estructurado como un atributo compuesto consistente en *nombre*, *primer-apellido* y *segundo-apellido*. Usar atributos compuestos en un esquema de diseño es una buena elección si el usuario desea referirse a un atributo completo en algunas ocasiones y, en otras, a algún componente del atributo. Se podrían haber sustituido los atributos del conjunto de entidades *cliente*, *calle-cliente* y *ciudad-cliente*, por el atributo compuesto *dirección-cliente*, con los atributos *calle, ciudad, provincia,* y *código-postal*[2]. Los atributos compuestos ayudan a agrupar los atributos relacionados, haciendo los modelos más claros.

  Nótese también que un atributo compuesto puede aparecer como una jerarquía. Volviendo al ejemplo del atributo compuesto *dirección-cliente*, su componente *calle* puede ser a su vez dividido en *número-calle*, *nombre-calle* y *piso*. Estos ejemplos de atributos compuestos para el conjunto de entidades *cliente* se representa en la Figura 2.2.
- Atributos **monovalorados** y **multivalorados**. Los atributos que se han especificado en los ejemplos tienen todos un valor sólo para una entidad concreta. Por ejemplo, el atributo *número-préstamo* para una entidad préstamo específico, referencia a un único número de préstamo. Tales atributos se llaman **monovalorados**. Puede haber ocasiones en las que un atributo tiene un conjunto de valores para una entidad específica. Considérese un conjunto de entidades *empleado* con el atributo *número-teléfono*. Cualquier empleado particular puede tener cero, uno o más números de teléfono. Este tipo de atributo se llama *multivalorado*. En ellos, se pueden colocar apropiadamente límites inferior y superior en el número de valores en el atributo **multivalorado**. Como otro ejemplo, un atributo *nombre-subordinado* del conjunto de entidades *empleado*

[2] Se asume el formato de *calle-cliente* y *dirección* usado en España, que incluye un código postal numérico llamado «código postal».

Santos	32.112.312	Mayor	Peguerinos
Gómez	01.928.374	Carretas	Cerceda
López	67.789.901	Mayor	Peguerinos
Sotoca	55.555.555	Real	Cádiz
Pérez	24.466.880	Carretas	Cerceda
Valdivieso	96.396.396	Goya	Vigo
Fernández	33.557.799	Jazmín	León

cliente

P-17	1.000
P-23	2.000
P-15	1.500
P-14	1.500
P-19	500
P-11	900
P-16	1.300

préstamo

FIGURA 2.1. Conjunto de entidades *cliente* y *préstamo*.

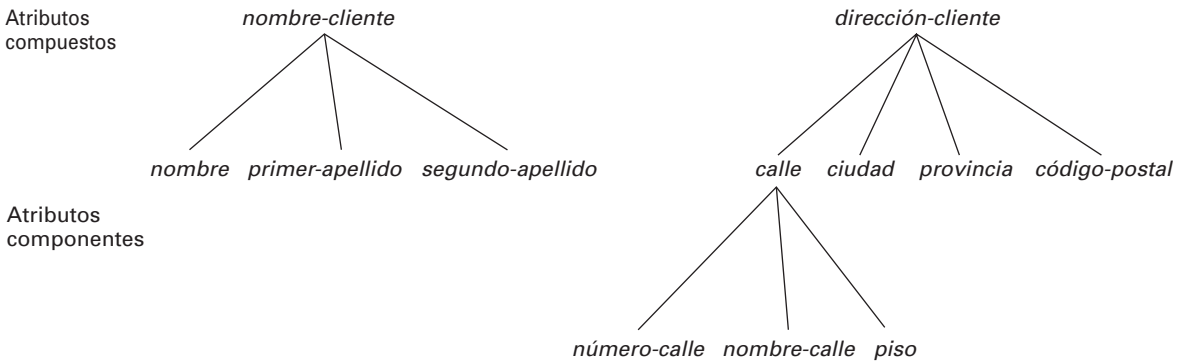

FIGURA 2.2. Atributos compuestos *nombre-cliente* y *dirección-cliente*.

sería multivalorado, ya que un empleado en concreto podría tener cero, uno o más subordinados.

Cuando sea apropiado se pueden establecer límites superior e inferior en el número de valores de un atributo multivalorado. Por ejemplo, un banco puede limitar el número de números de teléfono almacenados para un único cliente a dos. Colocando límites en este caso, se expresa que el atributo *número-teléfono* del conjunto de entidades *cliente* puede tener entre cero y dos valores.

- Atributos **derivados**. El valor para este tipo de atributo se puede derivar de los valores de otros atributos o entidades relacionados. Por ejemplo, sea el conjunto de entidades *cliente* que tiene un atributo *préstamos* que representa cuántos préstamos tiene un cliente en el banco. Ese atributo se puede derivar contando el número de entidades *préstamo* asociadas con ese *cliente*.

  Como otro ejemplo, considérese que el conjunto de entidades *empleado* tiene un atributo *edad*, que indica la edad del cliente. Si el conjunto de entidades *cliente* tiene también un atributo *fecha-de-nacimiento*, se puede calcular *edad* a partir de *fecha-de-nacimiento* y de la fecha actual. Así, *edad* es un atributo derivado. En este caso, *fecha-de-nacimiento* y *antigüedad* pueden serlo, ya que representan el primer día en que el empleado comenzó a trabajar para el banco y el tiempo total que el empleado lleva trabajando para el banco, respectivamente. El valor de *antigüedad* se puede derivar del valor de *fecha-comienzo* y de la fecha actual. En este caso, *fecha-comienzo* se puede conocer como atributo *base* o atributo *almacenado*. El valor de un atributo derivado no se almacena, sino que se calcula cuando sea necesario.

Un atributo toma un valor **nulo** cuando una entidad no tiene un valor para un atributo. El valor *nulo* también puede indicar «no aplicable», es decir, que el valor no existe para la entidad. Por ejemplo, una persona puede no tener segundo nombre de pila. *Nulo* puede también designar que el valor de un atributo es desconocido. Un valor desconocido puede ser, bien *perdido* (el valor existe pero no se tiene esa información) o *desconocido* (no se conoce si el valor existe realmente o no).

Por ejemplo, si el valor *nombre* para un *cliente* particular es *nulo*, se asume que el valor es perdido, ya que cada cliente debe tener un nombre. Un valor *nulo* para el atributo *piso* podría significar que la dirección no incluye un piso (no aplicable), que existe piso pero no se conoce cuál es (perdido), o que no se sabe si el piso forma parte o no de la dirección del cliente (desconocido).

Una base de datos para una empresa bancaria puede incluir diferentes conjuntos de entidades. Por ejemplo, además del mantenimiento de clientes y préstamos, el banco también proporciona cuentas, que se representan mediante el conjunto de entidades *cuenta* con atributos *número-cuenta* y *saldo*. También, si el banco tiene un número de sucursales diferentes, se puede mantener información acerca de todas las sucursales del banco. Cada conjunto de entidades *sucursal* se describe mediante los atributos *nombre-sucursal*, *ciudad-sucursal* y *activo*.

2.1.2. Conjuntos de relaciones

Una **relación** es una asociación entre diferentes entidades. Por ejemplo, se puede definir una relación que asocie al cliente López con el préstamo P-15. Esta relación especifica que López es un cliente con el préstamo número P-15.

Un **conjunto de relaciones** es un conjunto de relaciones del mismo tipo. Formalmente es una relación matemática con $n >= 2$ de conjuntos de entidades (posiblemente no distintos). Si $E_1, E_2, \ldots, E_n$ son conjuntos de entidades, entonces un conjunto de relaciones R es un subconjunto de:

$$\{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}$$

donde $(e_1, e_2, \ldots e_n)$ es una relación.

Considérense las dos entidades *cliente* y *préstamo* de la Figura 2.1. Se define el conjunto de relaciones *prestatario* para denotar la asociación entre clientes y préstamos bancarios que los clientes tengan. Esta asociación se describe en la Figura 2.3.

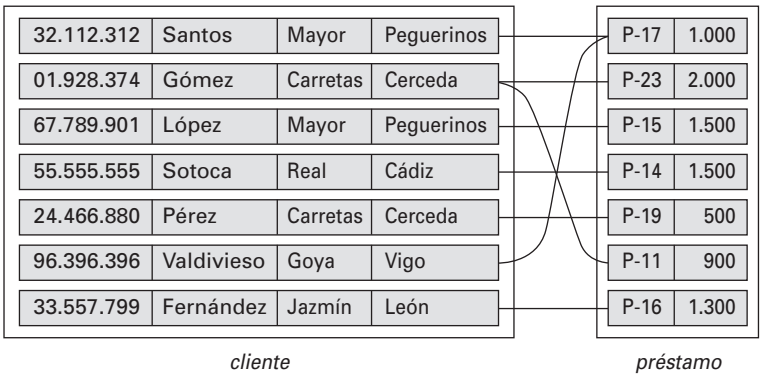

FIGURA 2.3. Conjunto de relaciones *prestatario*.

Como otro ejemplo, considérense los dos conjuntos de entidades *préstamo* y *sucursal*. Se puede definir el conjunto de relaciones *sucursal-préstamo* para denotar la asociación entre un préstamo y la sucursal en que se mantiene ese préstamo.

La asociación entre conjuntos de entidades se conoce como *participación*; es decir, los conjuntos de entidades E_1, E_2,…, E_n **participan** en el conjunto de relaciones R. Un **ejemplar de relación** en un esquema E-R representa que existe una asociación entre las entidades denominadas en la empresa del mundo real que se modela. Como ilustración, el *cliente* individual López, que tiene D.N.I. 67.789.901, y la entidad *préstamo* P-15 participan en un ejemplar de relación de *prestatario*. Este ejemplar de relación representa que, en la empresa del mundo real, la persona llamada López cuyo número de D.N.I. es 67.789.901 ha tomado un préstamo que está numerado como P-15.

La función que desempeña una entidad en una relación se llama **papel** de la entidad.

Debido a que los conjuntos de entidades que participan en un conjunto de relaciones son generalmente distintos, los papeles están implícitos y no se especifican normalmente. Sin embargo, son útiles cuando el significado de una relación necesita aclaración. Tal es el caso cuando los conjuntos de entidades de una relación no son distintos; es decir, el mismo conjunto de entidades participa en una relación más de una vez con diferentes papeles. En este tipo de conjunto de relaciones, que se llama algunas veces conjunto de relaciones **recursivo**, es necesario hacer explícitos los papeles para especificar cómo participa una entidad en un ejemplar de relación. Por ejemplo, considérese una conjunto de entidades *empleado* que almacena información acerca de todos los empleados del banco. Se puede tener un conjunto de relaciones *trabaja-para* que se modela mediante pares ordenados de entidades *empleado*. El primer empleado de un par toma el papel de *trabajador*, mientras el segundo toma el papel de *jefe*. De esta manera, todas las relaciones *trabaja-para* son pares (trabajador, jefe); los pares (jefe, trabajador) están excluidos.

Una relación puede también tener **atributos descriptivos**. Considérese un conjunto de relaciones *impositor* con conjuntos de entidades *cliente* y *cuenta*. Se podría asociar el atributo *fecha-acceso* a esta relación para especificar la fecha más reciente en que un cliente accedió a una cuenta. La relación *impositor* entre las entidades correspondientes al cliente García y la cuenta C-217 se describen mediante {(*fecha-acceso*, 23 mayo 2002)}, lo que significa que la última vez que García accedió a la cuenta C-217 fue el 23 de mayo de 2002.

Como otro ejemplo de atributos descriptivos para relaciones, supóngase que se tienen los conjuntos de entidades *estudiante* y *asignatura* que participan en una relación *matriculado*. Se podría desear almacenar un atributo descriptivo para *créditos* con la relación, para registrar si el estudiante se ha matriculado de la asignatura para obtener créditos o sólo como oyente.

Un ejemplar de relación en un conjunto de relaciones determinado debe ser identificado unívocamente a partir de sus entidades participantes, sin usar los atributos descriptivos. Para comprender este punto supóngase que deseemos modelar todas las fechas en las que un cliente ha accedido a una cuenta. El atributo monovalorado *fecha-acceso* puede almacenar sólo una única fecha de acceso. No se pueden representar varias fechas de acceso por varios ejemplares de relación entre el mismo cliente y cuenta, ya que los ejemplares de relación no estarían identificados unívocamente por las entidades participantes. La forma correcta de manejar este caso es crear un atributo multivalorado *fechas-acceso* que pueda almacenar todas las fechas de acceso.

Sin embargo, puede haber más de un conjunto de relaciones que involucren los mismos conjuntos de entidades. En nuestro ejemplo los conjuntos de entidades *cliente* y *préstamo* participan en el conjunto de relaciones *prestatario*. Además, supóngase que cada préstamo deba tener otro cliente que sirva como avalista para el préstamo. Entonces los conjuntos de entidades *cliente* y *préstamo* pueden participar en otro conjunto de relaciones: *avalista*.

Los conjuntos de relaciones *prestatario* y *sucursal-préstamo* proporcionan un ejemplo de un conjunto de relaciones **binario**, es decir, uno que implica dos conjuntos de entidades. La mayoría de los conjuntos de relaciones en un sistema de bases de datos son binarios.

Ocasionalmente, sin embargo, los conjuntos de relaciones implican más de dos conjuntos de entidades.

Por ejemplo, considérense los conjuntos de entidades *empleado*, *sucursal* y *trabajo*. Ejemplos de las entidades *trabajo* podrían ser director, cajero, auditor y otros. Las entidades trabajo pueden tener los atributos *puesto* y *nivel*. El conjunto de relaciones *trabaja-en* entre *empleado*, *sucursal* y *trabajo* es un ejemplo de una relación ternaria. Una relación ternaria entre Santos, Navacerrada y director indica que Santos actúa de director de la sucursal Navacerrada. Santos también podría actuar como auditor de la sucursal Centro, que estaría representado por otra relación. Podría haber otra relación entre Gómez, Centro y cajero, indicando que Gómez actúa como cajero en la sucursal Centro.

El número de conjuntos de entidades que participan en un conjunto de relaciones es también el **grado** del conjunto de relaciones. Un conjunto de relaciones binario tiene grado 2; un conjunto de relaciones ternario tiene grado 3.

2.2. RESTRICCIONES

Un esquema de desarrollo E-R puede definir ciertas restricciones a las que los contenidos de la base de datos se deben adaptar. En este apartado se examina la correspondencia de cardinalidades y las restricciones de participación, que son dos de los tipos más importantes de restricciones.

2.2.1. Correspondencia de cardinalidades

La **correspondencia de cardinalidades**, o razón de cardinalidad, expresa el número de entidades a las que otra entidad puede estar asociada vía un conjunto de relaciones.

La correspondencia de cardinalidades es la más útil describiendo conjuntos de relaciones binarias, aunque ocasionalmente contribuye a la descripción de conjuntos de relaciones que implican más de dos conjuntos de entidades. Este apartado se centrará en conjuntos de relaciones binarias únicamente.

Para un conjunto de relaciones binarias R entre los conjuntos de entidades A y B, la correspondencia de cardinalidades debe ser una de las siguientes:

- **Uno a uno**. Una entidad en A se asocia con *a lo sumo* una entidad en B, y una entidad en B se asocia con *a lo sumo* una entidad en A (véase la Figura 2.4a).

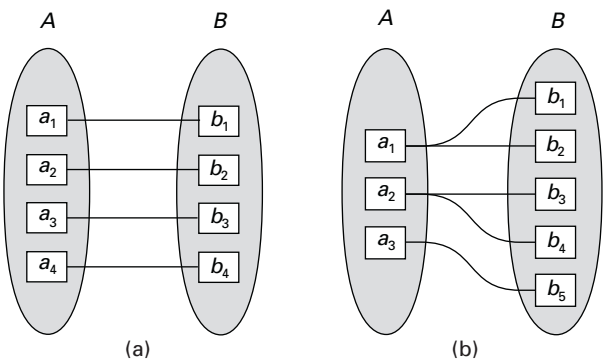

FIGURA 2.4. Correspondencia de cardinalidades. (a) Uno a uno. (b) Uno a varios.

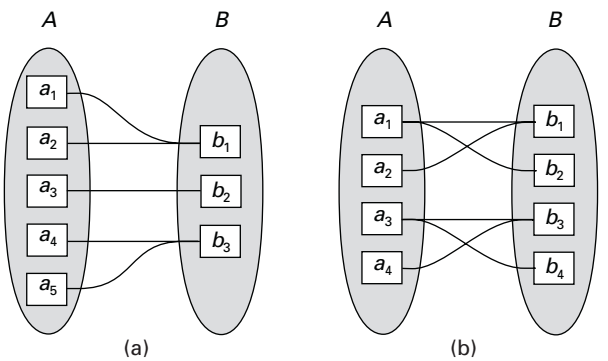

FIGURA 2.5. Correspondencia de cardinalidades. (a) Varios a uno. (b) Varios a varios.

- **Varios a uno**. Una entidad en A se asocia con *a lo sumo* una entidad en B. Una entidad en B, sin embargo, se puede asociar con cualquier número de entidades (ninguna o varias) en A (véase la Figura 2.5a).
- **Uno a varios**. Una entidad en A se asocia con cualquier número de entidades en B (ninguna o varias). Una entidad en B, sin embargo, se puede asociar con *a lo sumo* una entidad en A (véase la Figura 2.4b).
- **Varios a varios**. Una entidad en A se asocia con cualquier número de entidades (ninguna o varias) en B, y una entidad en B se asocia con cualquier número de entidades (ninguna o varias) en A (véase la Figura 2.5b).

La correspondencia de cardinalidades apropiada para un conjunto de relaciones particular depende obviamente de la situación del mundo real que el conjunto de relaciones modela.

Como ilustración considérese el conjunto de relaciones *prestatario*. Si en un banco particular un préstamo puede pertenecer únicamente a un cliente y un cliente puede tener varios préstamos, entonces el conjunto de relaciones de *cliente* a *préstamo* es uno a varios. Si un préstamo puede pertenecer a varios clientes (como préstamos que se toman en conjunto por varios socios de un negocio) el conjunto de relaciones es varios a varios. Este tipo de relación se describe en la Figura 2.3.

2.2.2. Restricciones de participación

La participación de un conjunto de entidades *E* en un conjunto de relaciones *R* se dice que es **total** si cada entidad en *E* participa al menos en una relación en R. Si sólo algunas entidades en *E* participan en relaciones en *R*, la participación del conjunto de entidades *E* en la relación *R* se llama **parcial**. Por ejemplo, se puede esperar que cada entidad *préstamo* esté relacionada con al menos un cliente mediante la relación *prestatario*. Por lo tanto, la participación de *préstamo* en el conjunto de relaciones *prestatario* es total. En cambio, un individuo puede ser cliente de un banco tenga o no tenga un préstamo en el banco. Así, es posible que sólo algunas de las entidades *cliente* estén relacionadas con el conjunto de entidades *préstamo* mediante la relación *prestatario*, y la participación de *cliente* en el conjunto de relaciones *prestatario* es por lo tanto parcial.

2.3. CLAVES

Es necesario tener una forma de especificar cómo las entidades dentro de un conjunto de entidades dado y las relaciones dentro de un conjunto de relaciones dado son distinguibles. Conceptualmente las entidades y relaciones individuales son distintas; desde una perspectiva de bases de datos, sin embargo, la diferencia entre ellas se debe expresar en término de sus atributos.

Por lo tanto, los valores de los atributos de una entidad deben ser tales que permitan *identificar unívocamente* a la entidad. En otras palabras, no se permite que ningún par de entidades tengan exactamente los mismos valores de sus atributos.

Una *clave* permite identificar un conjunto de atributos suficiente para distinguir las entidades entre sí. Las claves también ayudan a identificar unívocamente a las relaciones y así a distinguir las relaciones entre sí.

2.3.1. Conjuntos de entidades

Una **superclave** es un conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma única una entidad en el conjunto de entidades. Por ejemplo, el atributo *id-cliente* del conjunto de entidades *cliente* es suficiente para distinguir una entidad *cliente* de las otras. Así, *id-cliente* es una superclave. Análogamente, la combinación de *nombre-cliente* e *id-cliente* es una superclave del conjunto de entidades *cliente*. El atributo *nombre-cliente* de *cliente* no es una superclave, porque varias personas podrían tener el mismo nombre.

El concepto de una superclave no es suficiente para lo que aquí se propone, ya que, como se ha visto, una superclave puede contener atributos innecesarios. Si *K* es una superclave, entonces también lo es cualquier superconjunto de *K*. A menudo interesan las superclaves tales que los subconjuntos propios de ellas no son superclave. Tales superclaves mínimas se llaman **claves candidatas**.

Es posible que conjuntos distintos de atributos pudieran servir como clave candidata. Supóngase que una combinación de *nombre-cliente* y *calle-cliente* es suficiente para distinguir entre los miembros del conjunto de entidades *cliente*. Entonces, los conjuntos {*id-cliente*} y {*nombre-cliente*, *calle-cliente*} son claves candidatas. Aunque los atributos *id-cliente* y *nombre-cliente* juntos puedan distinguir entidades *cliente*, su combinación no forma una clave candidata, ya que el atributo *id-cliente* por sí solo es una clave candidata.

Se usará el término **clave primaria** para denotar una clave candidata que es elegida por el diseñador de la base de datos como elemento principal para identificar las entidades dentro de un conjunto de entidades. Una clave (primaria, candidata y superclave) es una propiedad del conjunto de entidades, más que de las entidades individuales. Cualesquiera dos entidades individuales en el conjunto no pueden tener el mismo valor en sus atributos clave al mismo tiempo. La designación de una clave representa una restricción en el desarrollo del mundo real que se modela.

Las claves candidatas se deben designar con cuidado. Como se puede comprender, el nombre de una persona es obviamente insuficiente, ya que hay mucha gente con el mismo nombre. En España, el D.N.I. puede ser una clave candidata. Como los no residentes en España normalmente no tienen D.N.I., las empresas internacionales pueden generar sus propios identificadores únicos. Una alternativa es usar alguna combinación única de otros atributos como clave.

La clave primaria se debería elegir de manera que sus atributos nunca, o muy raramente, cambien. Por ejemplo, el campo dirección de una persona no debería formar parte de una clave primaria, porque probablemente cambiará. Los números de D.N.I., por otra parte, es seguro que no cambiarán. Los identificadores únicos generados por empresas generalmente no cambian, excepto si se fusionan dos empresas; en tal caso el mismo identificador puede haber sido emitido por ambas empresas y es necesario la reasignación de identificadores para asegurarse de que sean únicos.

2.3.2. Conjuntos de relaciones

La clave primaria de un conjunto de entidades permite distinguir entre las diferentes entidades del conjunto. Se necesita un mecanismo similar para distinguir entre las diferentes relaciones de un conjunto de relaciones.

Sea *R* un conjunto de relaciones que involucra los conjuntos de entidades E_1, E_2,…, E_n. Sea *clave-prima-*

ria(E_i) el conjunto de atributos que forma la clave primaria para el conjunto de entidades E_i.

Asúmase por el momento que los nombres de los atributos de todas las claves primarias son únicos y que cada conjunto de entidades participa sólo una vez en la relación. La composición de la clave primaria para un conjunto de relaciones depende de la estructura de los atributos asociados al conjunto de relaciones *R*.

Si el conjunto de relaciones *R* no tiene atributos asociados, entonces el conjunto de atributos:

$$clave\text{-}primaria(E_1) \cup clave\text{-}primaria(E_2) \cup \ldots \cup clave\text{-}primaria(E_n)$$

describe una relación individual en el conjunto *R*.

Si el conjunto de relaciones *R* tiene atributos a_1, $a_2, \ldots, a_m$ asociados a él, entonces el conjunto de atributos

$$clave\text{-}primaria(E_1) \cup clave\text{-}primaria(E_2) \cup \ldots \cup clave\text{-}primaria(E_n) \cup \{a_1, a_2, \ldots, a_m\}$$

describe una relación individual en el conjunto *R*.

En ambos casos, el conjunto de atributos

$$clave\text{-}primaria(E_1) \cup clave\text{-}primaria(E_2) \cup \ldots \cup clave\text{-}primaria(E_n)$$

forma una superclave para el conjunto de relaciones.

En el caso de que los nombres de atributos de las claves primarias no sean únicos en todos los conjuntos de entidades, los atributos se renombran para distinguirlos; el nombre del conjunto de entidades combinado con el atributo formaría un nombre único. En el caso de que un conjunto de entidades participe más de una vez en un conjunto de relaciones (como en la relación *trabaja-para* del Apartado 2.1.2) el nombre del papel se usa en lugar del nombre del conjunto de entidades para formar un nombre único de atributo.

La estructura de la clave primaria para el conjunto de relaciones depende de la correspondencia de cardinalidades asociada al conjunto de relaciones. Como ilustración, considérese el conjunto de entidades *cliente* y *cuenta*, y un conjunto de relaciones *impositor*, con el atributo *fecha-acceso* del Apartado 2.1.2. Supóngase que el conjunto de relaciones es varios a varios. Entonces la clave primaria de *impositor* consiste en la unión de las claves primarias de *cliente* y *cuenta*. Sin embargo, si un cliente puede tener sólo una cuenta —es decir, si la relación *impositor* es varios a uno de *cliente* a *cuenta*— entonces la clave primaria de *impositor* es simplemente la clave primaria de *cliente*. Análogamente, si la relación es varios a uno de *cuenta* a *cliente* —es decir, cada cuenta pertenece a lo sumo a un cliente— entonces la clave primaria de *impositor* es simplemente la clave primaria de *cuenta*. Para relaciones uno a uno se puede usar cualquier clave primaria.

Para las relaciones no binarias, si no hay restricciones de cardinalidad, entonces la superclave formada como se describió anteriormente en este apartado es la única clave candidata, y se elige como clave primaria. La elección de la clave primaria es más complicada si aparecen restricciones de cardinalidad. Ya que no se ha discutido cómo especificar restricciones de cardinalidad en relaciones no binarias, no se discutirá este aspecto en este capítulo. Se considerará este aspecto con más detalle en el apartado 7.3.

2.4. CUESTIONES DE DISEÑO

Las nociones de conjunto de entidades y conjunto de relaciones no son precisas, y es posible definir un conjunto de entidades y las relaciones entre ellas de diferentes formas. En este apartado se examinan cuestiones básicas de diseño de un esquema de bases de datos E-R. El proceso de diseño se trata con más detalle en el Apartado 2.7.4.

2.4.1. Uso de conjuntos de entidades o atributos

Considérese el conjunto de entidades *empleado* con los atributos *nombre-empleado* y *número-teléfono*. Se puede argumentar fácilmente que un *teléfono* es una entidad por sí misma con atributos *número-teléfono* y *ubicación* (la oficina donde está ubicado el teléfono). Si se toma este punto de vista, el conjunto de entidades *empleado* debe ser redefinido como sigue:

- El conjunto de entidades *empleado* con el atributo *nombre-empleado*
- El conjunto de entidades *teléfono* con atributos *número-teléfono* y *ubicación*
- La relación *empleado-teléfono*, que denota la asociación entre empleados y los teléfonos que tienen.

¿Cuál es, entonces, la diferencia principal entre esas dos definiciones de un empleado? Al tratar un teléfono como un atributo *número-teléfono* implica que cada empleado tiene precisamente un número de teléfono. Al tratar un teléfono como una entidad *teléfono* permite que los empleados puedan tener varios números de teléfono (incluido ninguno) asociados a ellos. Sin embargo, se podría definir fácilmente *número-teléfono* como un atributo multivalorado para permitir varios teléfonos por empleado.

CAPÍTULO

3 EL MODELO RELACIONAL

El modelo relacional se ha establecido actualmente como el principal modelo de datos para las aplicaciones de procesamiento de datos. Ha conseguido la posición principal debido a su simplicidad, que facilita el trabajo del programador en comparación con otros modelos anteriores como el de red y el jerárquico.

En este capítulo se estudia en primer lugar los fundamentos del modelo relacional, que proporciona una forma muy simple y potente de representar datos. A continuación se describen tres lenguajes formales de consulta; los lenguajes de consulta se usan para especificar las solicitudes de información. Los tres que se estudian en este capítulo no son cómodos de usar, pero a cambio sirven como base formal para lenguajes de consulta que sí lo son y que se estudiarán más adelante. El primer lenguaje de consulta, el álgebra relacional, se estudia en detalle. El álgebra relacional forma la base del lenguaje de consulta SQL ampliamente usado. A continuación se proporcionan visiones generales de otros dos lenguajes formales: el cálculo relacional de tuplas y el cálculo relacional de dominios, que son lenguajes declarativos de consulta basados en la lógica matemática. El cálculo relacional de dominios es la base del lenguaje QBE.

Existe una amplia base teórica de las bases de datos relacionales. En este capítulo se estudia la base teórica referida a las consultas. En el Capítulo 7 se examinarán aspectos de la teoría de bases de datos relacionales que ayudan en el diseño de esquemas de bases de datos relacionales, mientras que en los Capítulos 13 y 14 se estudian aspectos de la teoría que se refieren al procesamiento eficiente de consultas.

3.1. LA ESTRUCTURA DE LAS BASES DE DATOS RELACIONALES

Una base de datos relacional consiste en un conjunto de **tablas**, a cada una de las cuales se le asigna un nombre exclusivo. Cada tabla tiene una estructura parecida a la presentada en el Capítulo 2, donde se representaron las bases de datos E-R mediante tablas. Cada fila de la tabla representa una *relación* entre un conjunto de valores. Dado que cada tabla es un conjunto de dichas relaciones, hay una fuerte correspondencia entre el concepto de *tabla* y el concepto matemático de *relación*, del que toma su nombre el modelo de datos relacional. A continuación se introduce el concepto de relación.

En este capítulo se utilizarán varias relaciones diferentes para ilustrar los conceptos subyacentes al modelo de datos relacional. Estas relaciones representan parte de una entidad bancaria. Se diferencian ligeramente de las tablas que se utilizaron en el Capítulo 2, por lo que se puede simplificar la representación. En el Capítulo 7 se estudiarán los criterios sobre la adecuación de las estructuras relacionales.

3.1.1. Estructura básica

Considérese la tabla *cuenta* de la Figura 3.1. Tiene tres cabeceras de columna: *número-cuenta*, *nombre-sucursal* y *saldo*. Siguiendo la terminología del modelo relacional se puede hacer referencia a estas cabeceras como **atributos** (igual que se hizo en el modelo E-R en el Capítulo 2). Para cada atributo hay un conjunto de valores permitidos, llamado **dominio** de ese atributo. Para el atributo *nombre-sucursal*, por ejemplo, el dominio es el conjunto de los nombres de las sucursales. Supongamos que D_1 denota el conjunto de todos los números de cuenta, D_2 el conjunto de todos los nombres de sucursal y D_3 el conjunto de los saldos. Como se vio en el Capítulo 2 todas las filas de *cuenta* deben consistir en una tupla (v_1, v_2, v_3), donde v_1 es un número de cuenta (es decir, v_1 está en el dominio D_1), v_2 es un nombre de sucursal (es decir, v_2 está en el dominio D_2) y v_3 es un saldo (es decir, v_3 está en el dominio D_3). En general,

número-cuenta	*nombre-sucursal*	*saldo*
C-101	Centro	500
C-102	Navacerrada	400
C-201	Galapagar	900
C-215	Becerril	700
C-217	Galapagar	750
C-222	Moralzarzal	700
C-305	Collado Mediano	350

FIGURA 3.1. La relación *cuenta*.

cuenta sólo contendrá un subconjunto del conjunto de todas las filas posibles. Por tanto, *cuenta* es un subconjunto de

$$D_1 \times D_2 \times D_3$$

En general, una **tabla** de *n* atributos debe ser un subconjunto de

$$D_1 \times D_2 \times \ldots \times D_{n-1} \times D_n$$

Los matemáticos definen las **relaciones** como subconjuntos del producto cartesiano de la lista de dominios. Esta definición se corresponde de manera casi exacta con la definición de tabla dada anteriormente. La única diferencia es que aquí se han asignado nombres a los atributos, mientras que los matemáticos sólo utilizan «nombres» numéricos, utilizando el entero 1 para denotar el atributo cuyo dominio aparece en primer lugar en la lista de dominios, 2 para el atributo cuyo dominio aparece en segundo lugar, etcétera. Como las tablas son esencialmente relaciones, se utilizarán los términos matemáticos **relación** y **tupla** en lugar de los términos **tabla** y **fila**. Una **variable tupla** es una variable que representa a una tupla; en otras palabras, una tupla que representa al conjunto de todas las tuplas.

En la relación *cuenta* de la Figura 3.1 hay siete tuplas. Supóngase que la variable tupla *t* hace referencia a la primera tupla de la relación. Se utiliza la notación *t*[*número-cuenta*] para denotar el valor de *t* en el atributo *número-cuenta*. Por tanto, *t*[*número-cuenta*] = «C-101» y *t*[*nombre-sucursal*] = «Centro». De manera alternativa, se puede escribir *t*[1] para denotar el valor de la tupla *t* en el primer atributo (*número-cuenta*), *t*[2] para denotar *nombre-sucursal*, etcétera. Dado que las relaciones son conjuntos se utiliza la notación matemática $t \in r$ para denotar que la tupla *t* está en la relación *r*.

El orden en que aparecen las tuplas es irrelevante, dado que una relación es un *conjunto* de tuplas. Así, si las tuplas de una relación se muestran ordenadas como en la Figura 3.1, o desordenadas, como en la Figura 3.2, no importa; las relaciones de estas figuras son las mismas, ya que ambas contienen el mismo conjunto de tuplas.

Se exigirá que, para todas las relaciones *r*, los dominios de todos los atributos de *r* sean atómicos. Un dominio es **atómico** si los elementos del dominio se consideran unidades indivisibles. Por ejemplo, el conjunto de los enteros es un dominio atómico, pero el conjunto de todos los conjuntos de enteros es un dominio no atómico. La diferencia es que no se suele considerar que los enteros tengan subpartes, pero sí se considera que los conjuntos de enteros las tienen; por ejemplo, los enteros que forman cada conjunto. Lo importante no es lo que sea el propio dominio, sino la manera en que se utilizan los elementos del dominio en la base de datos. El dominio de todos los enteros sería no atómico si se considerase que cada entero fuera una lista ordenada de cifras. En todos los ejemplos se supondrá que los dominios son atómicos. En el Capítulo 9 se estudiarán extensiones al modelo de datos relacional para permitir dominios no atómicos.

Es posible que varios atributos tengan el mismo dominio. Por ejemplo, supóngase que se tiene una relación *cliente* que tiene los tres atributos *nombre-cliente*, *calle-cliente* y *ciudad-cliente* y una relación *empleado* que incluye el atributo *nombre-empleado*. Es posible que los atributos *nombre-cliente* y *nombre-empleado* tengan el mismo dominio, el conjunto de todos los nombres de personas, que en el nivel físico son cadenas de caracteres. Los dominios de *saldo* y *nombre-sucursal*, por otra parte, deberían ser distintos. Quizás es menos claro si *nombre-cliente* y *nombre-sucursal* deberían tener el mismo dominio. En el nivel físico, tanto los nombres de clientes como los nombres de sucursales son cadenas de caracteres. Sin embargo, en el nivel lógico puede que se desee que *nombre-cliente* y *nombre-sucursal* tengan dominios diferentes.

Un valor de dominio que es miembro de todos los dominios posibles es el valor **nulo**, que indica que el valor es desconocido o no existe. Por ejemplo, supóngase que se incluye el atributo *número-teléfono* en la relación *cliente*. Puede ocurrir que un cliente no tenga número de teléfono, o que su número de teléfono no figure en la guía. Entonces habrá que recurrir a los valores nulos para indicar que el valor es desconocido o que no existe. Más adelante se verá que los valores nulos crean algunas dificultades cuando se tiene acceso a la base de datos o cuando se actualiza y que, por tanto, deben eliminarse si es posible. Se asumirá inicialmente que no hay valores nulos y en el Apartado 3.3.4 se describirá el efecto de los valores nulos en las diferentes operaciones.

número-cuenta	*nombre-sucursal*	*saldo*
C-101	Centro	500
C-215	Becerril	700
C-102	Navacerrada	400
C-305	Collado Mediano	350
C-201	Galapagar	900
C-222	Moralzarzal	700
C-217	Galapagar	750

FIGURA 3.2. La relación *cuenta* con las tuplas desordenadas.

3.1.2. Esquema de la base de datos

Cuando se habla de bases de datos se debe diferenciar entre el **esquema de la base de datos**, o diseño lógico de la misma, y el **ejemplar de la base de datos**, que es una instantánea de los datos de la misma en un momento dado.

El concepto de relación se corresponde con el concepto de variable de los lenguajes de programación. El concepto de **esquema de la relación** se corresponde con el concepto de definición de tipos de los lenguajes de programación.

Resulta conveniente dar un nombre a los esquemas de las relaciones, igual que se dan nombres a las definiciones de tipos en los lenguajes de programación. Se adopta el convenio de utilizar nombres en minúsculas para las relaciones y nombres que comiencen por una letra mayúscula para los esquemas de las relaciones. Siguiendo esta notación se utilizará *Esquema-cuenta* para denotar el esquema de la relación de la relación *cuenta*. Por tanto,

Esquema-cuenta = (*número-cuenta, nombre-sucursal, saldo*)

Se denota el hecho de que *cuenta* es una relación de *Esquema-cuenta* mediante

cuenta (*Esquema-cuenta*)

En general, los esquemas de las relaciones incluyen una lista de los atributos y de sus dominios correspondientes. La definición exacta del dominio de cada atributo no será relevante hasta que se discuta el lenguaje SQL en el Capítulo 4.

El concepto de **ejemplar de relación** se corresponde con el concepto de valor de una variable en los lenguajes de programación. El valor de una variable dada puede cambiar con el tiempo; de manera parecida, el contenido del ejemplar de una relación puede cambiar con el tiempo cuando la relación se actualiza. Sin embargo, se suele decir simplemente «relación» cuando realmente se quiere decir «ejemplar de la relación».

Como ejemplo de ejemplar de una relación, considérese la relación *sucursal* de la Figura 3.3. El esquema de esa relación es

Esquema-relación = (*nombre-sucursal, ciudad-sucursal, activos*)

Obsérvese que el atributo *nombre de la sucursal* aparece tanto en *Esquema-sucursal* como en *Esquema-cuenta*. Esta duplicidad no es una coincidencia. Más bien, utilizar atributos comunes en los esquemas de las relaciones es una manera de relacionar las tuplas de relaciones diferentes. Por ejemplo, supóngase que se desea obtener información sobre todas las cuentas abiertas en sucursales ubicadas en Arganzuela. Primero se busca en la relación *sucursal* para encontrar los nombres de todas las sucursales sitas en Arganzuela. Luego, para cada una de ellas, se mira en la relación *cuenta* para encontrar la información sobre las cuentas abiertas en esa sucursal. Esto no es sorprendente: recuérdese que los atributos que forma la clave primaria de un conjunto de entidades fuertes aparecen en la tabla creada para representar el conjunto de entidades, así como en las tablas creadas para crear relaciones en las que participar el conjunto de entidades.

Continuemos con el ejemplo bancario. Se necesita una relación que describa la información sobre los clientes. El esquema de la relación es:

Esquema-cliente = (*nombre-cliente, calle-cliente, ciudad-cliente*)

En la Figura 3.4 se muestra un ejemplo de la relación *cliente* (*Esquema-cliente*). Obsérvese que se ha omitido el atributo *id-cliente*, que se usó en el Capítulo 2, porque no se desea tener esquemas de relación más pequeños en este ejemplo. Se asume que el nombre de cliente identifica unívocamente un cliente; obviamente, esto no es cierto en el mundo real, pero las suposiciones hechas en estos ejemplos los hacen más sencillos de entender.

En una base de datos del mundo real, *id-cliente* (que podría ser el número de la seguridad social o un identificador generado por el banco) serviría para identificar unívocamente a los clientes.

También se necesita una relación que describa la asociación entre los clientes y las cuentas. El esquema de la relación que describe esta asociación es:

Esquema-impositor = (*nombre-cliente, número-cuenta*)

En la Figura 3.5 se muestra un ejemplo de la relación *impositor* (*Esquema-impositor*).

Puede parecer que, para el presente ejemplo bancario, se podría tener sólo un esquema de relación, en vez de tener varios. Es decir, puede resultar más sencillo para el usuario pensar en términos de un esquema de

nombre de la sucursal	*ciudad de la sucursal*	*activos*
Galapagar	Arganzuela	7.500
Centro	Arganzuela	9.000.000
Becerril	Aluche	2.000
Segovia	Cerceda	3.700.000
Navacerrada	Aluche	1.700.000
Navas de la Asunción	Alcalá de Henares	1.500
Moralzarzal	La Granja	2.500
Collado Mediano	Aluche	8.000.000

FIGURA 3.3. La relación *sucursal*.

nombre-cliente	*calle-cliente*	*ciudad-cliente*
Abril	Preciados	Valsaín
Amo	Embajadores	Arganzuela
Badorrey	Delicias	Valsaín
Fernández	Jazmín	León
Gómez	Carretas	Cerceda
González	Arenal	La Granja
López	Mayor	Peguerinos
Pérez	Carretas	Cerceda
Rodríguez	Yeserías	Cádiz
Rupérez	Ramblas	León
Santos	Mayor	Peguerinos
Valdivieso	Goya	Vigo

FIGURA 3.4. La relación *cliente*.

nombre cliente	*número cuenta*
Abril	C-102
Gómez	C-101
González	C-201
González	C-217
López	C-222
Rupérez	C-215
Santos	C-305

FIGURA 3.5. La relación *impositor*.

número-préstamo	*nombre-sucursal*	*importe*
P-11	Collado Mediano	900
P-14	Centro	1.500
P-15	Navacerrada	1.500
P-16	Navacerrada	1.300
P-17	Centro	1.000
P-23	Moralzarzal	2.000
P-93	Becerril	500

FIGURA 3.6. La relación *préstamo*.

relación, en lugar de en términos de varios esquemas. Supóngase que sólo se utilizara una relación para el ejemplo, con el esquema

(*nombre-sucursal, ciudad-sucursal, activos, nombre-cliente, calle-cliente, ciudad-cliente, número-cuenta, saldo*)

Obsérvese que si un cliente tiene varias cuentas hay que repetir su dirección una vez por cada cuenta. Es decir, hay que repetir varias veces parte de la información. Esta repetición supone un gasto inútil y se evita mediante el uso de varias relaciones, como en el ejemplo presente.

Además, si una sucursal no tiene ninguna cuenta (por ejemplo, una sucursal recién creada que todavía no tiene clientes), no se puede construir una tupla completa en la relación única anterior, dado que no hay todavía ningún dato disponible referente a *cliente* ni a *cuenta*. Para representar las tuplas incompletas hay que utilizar valores *nulos* que indiquen que ese valor es desconocido o no existe. Por tanto, en el ejemplo presente, los valores de *nombre-cliente*, *calle-cliente*, etcétera, deben quedar nulos. Utilizando varias relaciones se puede representar la información de las sucursales de un banco sin clientes sin utilizar valores nulos. Sencillamente, se utiliza una tupla en *Esquema-sucursal* para representar la información de la sucursal y sólo crear tuplas en los otros esquemas cuando esté disponible la información adecuada.

En el Capítulo 7 se estudiarán los criterios para decidir cuándo un conjunto de esquemas de relaciones es más apropiado que otro en términos de repetición de la información y de la existencia de valores nulos. Por ahora se supondrá que los esquemas de las relaciones vienen dados de antemano.

Se incluyen dos relaciones más para describir los datos de los préstamos concedidos en las diferentes sucursales del banco:

Esquema-préstamo = (*número-préstamo, nombre-sucursal, importe*)
Esquema-prestatario = (*nombre-cliente, número-préstamo*)

Las relaciones de ejemplo *préstamo* (*Esquema-préstamo*) y *prestatario* (*Esquema-prestatario*) se muestran en las Figuras 3.5 y 3.6, respectivamente.

La entidad bancaria que se ha descrito se deriva del diagrama E-R mostrado en la Figura 3.8. Los esquemas de las relaciones se corresponden con el conjunto de tablas que se podrían generar utilizando el método esbozado en el Apartado 2.9. Obsérvese que las tablas para *cuenta-sucursal* y *préstamo-sucursal* se han combinado en las tablas de *cuenta* y *préstamo* respectivamente. Esta combinación es posible dado que las relaciones son de varios a uno desde *cuenta* y *préstamo*, respectivamente, a *sucursal* y, además, la participación de *cuenta* y *préstamo* en las relaciones correspondientes es total, como indican las líneas dobles en la figura. Finalmente, obsérvese que la relación *cliente* puede contener información sobre clientes que ni tengan cuenta ni un préstamo en el banco.

La entidad bancaria aquí descrita servirá como ejemplo principal en este capítulo y en los siguientes. Cuando sea necesario, habrá que introducir más esquemas de relaciones para ilustrar casos concretos.

3.1.3. Claves

Los conceptos de *superclave*, de *clave candidata* y de *clave primaria*, tal y como se discute en el Capítulo 2, también son aplicables en el modelo relacional. Por ejemplo, en *Esquema-sucursal*, tanto {*nombre-sucursal*} como {*nombre-sucursal*, *ciudad-sucursal*} son superclaves. {*nombre-sucursal*, *ciudad-sucursal*} no es una clave candidata porque {*nombre-sucursal*} es un subconjunto de {*nombre-sucursal*, *ciudad-sucursal*} y {*nombre-sucursal*} es una superclave. Sin embargo, {*nombre-sucursal*} es una clave candidata, y servirá también como clave primaria para estos fines. El atributo *ciudad-sucursal* no es una superclave, dado que dos sucursales de la misma ciudad pueden tener nombres diferentes (y diferentes volúmenes de activos).

nombre cliente	*número préstamo*
Fernández	P-16
Gómez	P-93
Gómez	P-15
López	P-14
Pérez	P-17
Santos	P-11
Sotoca	P-23
Valdivieso	P-17

FIGURA 3.7. La relación *prestatario*.

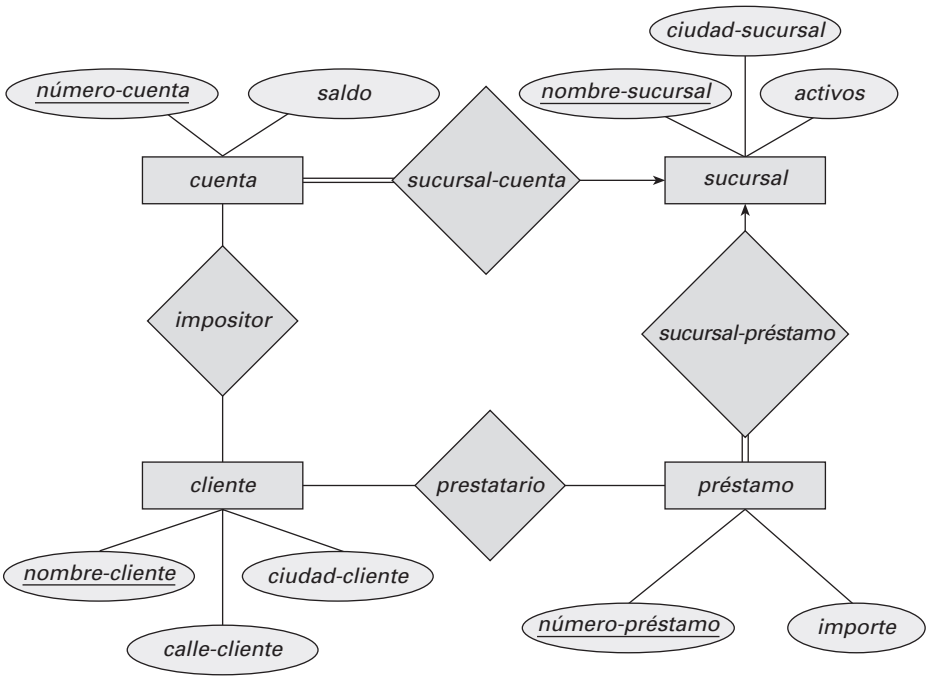

FIGURA 3.8. Diagrama E-R de la entidad bancaria.

Sea R el esquema de una relación. Si se dice que un subconjunto K de R es una *superclave* de R para las relaciones $r(R)$ en las que no hay dos tuplas diferentes que tengan los mismos valores en todos los atributos de K. Es decir, si t_1 y t_2 están en R y $t_1 \neq t_2$, entonces $t_1[K] \neq t_2[K]$.

Si el esquema de una base de datos relacional se basa en las tablas derivadas de un esquema E-R es posible determinar la clave primaria del esquema de una relación a partir de las claves primarias de los conjuntos de entidades o de relaciones de los que se deriva el esquema:

- **Conjunto de entidades fuertes.** La clave primaria del conjunto de entidades se convierte en la clave primaria de la relación.
- **Conjunto de entidades débiles.** La tabla y, por tanto, la relación correspondientes a un conjunto de entidades débiles incluyen
  - — Los atributos del conjunto de entidades débiles.
  - — La clave primaria del conjunto de entidades fuertes del que depende el conjunto de entidades débiles.

  La clave primaria de la relación consiste en la unión de la clave primaria del conjunto de entidades fuertes y el discriminante del conjunto de entidades débil.
- **Conjunto de relaciones.** La unión de las claves primarias de los conjuntos de entidades relacionados se transforma en una superclave de la relación. Si la relación es de varios a varios, esta superclave es también la clave primaria. En el Apartado 2.4.2 se describe la manera de determinar las claves primarias en otros casos. Recuérdese del Apartado 2.9.3 que no se genera ninguna tabla para los conjuntos de relaciones que vinculan un conjunto de entidades débiles con el conjunto de entidades fuertes correspondiente.
- **Tablas combinadas.** Recuérdese del Apartado 2.9.3 que un conjunto binario de relaciones de varios a uno entre A y B puede representarse mediante una tabla que consista en los atributos de A y en los atributos (si hay alguno) del conjunto de relaciones. La clave primaria de la entidad «varios» se transforma en la clave primaria de la relación (es decir, si el conjunto de relaciones es de varios a uno entre A y B, la clave primaria de A es la clave primaria de la relación). Para los conjuntos de relaciones de uno a uno la relación se construye igual que en el conjunto de relaciones de varios a uno. Sin embargo, cualquiera de las claves primarias del conjunto de entidades puede elegirse como clave primaria de la relación, dado que ambas son claves candidatas.
- **Atributos multivalorados.** Recuérdese del Apartado 2.9.4 que un atributo multivalorado M se representa mediante una tabla consistente en la clave primaria del conjunto de entidades o de relaciones del que M es atributo y en una columna C que guarda un valor concreto de M. La clave primaria del conjunto de entidades o de relaciones

junto con el atributo C se convierte en la clave primaria de la relación.

A partir de la lista precedente se puede ver que el esquema de una relación puede incluir entre sus atributos la clave primaria de otro esquema, digamos r_2. Este atributo es una **clave externa** de r_1 que hace referencia a r_2. La relación r_1 también se denomina la **relación referenciante** de la dependencia de clave externa, y r_2 se denomina la **relación referenciada** de la clave externa. Por ejemplo, el atributo *nombre-sucursal* de *Esquema-cuenta* es una clave externa de *Esquema-cuenta* referenciando a *Esquema-sucursal*, ya que *nombre-sucursal* es la clave primaria de *Esquema-sucursal*. En cualquier ejemplar de la base de datos, dada una tupla t_a de la relación *cuenta*, debe haber alguna tupla t_b en la relación *cuenta* tal que el valor del atributo *nombre-sucursal* de t_a sea el mismo que el valor de la clave primaria, *nombre-sucursal*, de t_b.

Es obligado listar los atributos que forman clave primaria de un esquema de relación antes que el resto; por ejemplo, el atributo *nombre-sucursal* de *Esquema-sucursal* se lista en primer lugar, ya que es la clave primaria.

3.1.4. Diagramas de esquema

Un esquema de bases de datos, junto con las dependencias de clave primaria y externa, se puede mostrar gráficamente mediante **diagramas de esquema**. La Figura 3.9 muestra el diagrama de esquema del ejemplo bancario. Cada relación aparece como un cuadro con los atributos listados dentro de él y el nombre de la relación sobre él. Si hay atributos clave primaria, una línea horizontal cruza el cuadro con los atributos clave primaria listados sobre ella. Las dependencias de clave externa aparecen como flechas desde los atributos clave externa de la relación referenciante a la clave primaria de la relación referenciada.

No hay que confundir un diagrama de esquema con un diagrama E-R. En particular, los diagramas E-R no muestran explícitamente los atributos clave externa, mientras que los diagramas de esquema sí.

Muchos sistemas de bases de datos proporcionan herramientas de diseño con una interfaz gráfica de usuario para la creación de diagramas de esquema.

3.1.5. Lenguajes de consulta

Un **lenguaje de consulta** es un lenguaje en el que un usuario solicita información de la base de datos. Estos lenguajes suelen ser de un nivel superior que el de los lenguajes de programación habituales. Los lenguajes de consulta pueden clasificarse como procedimentales o no procedimentales. En los **lenguajes procedimentales** el usuario instruye al sistema para que lleve a cabo una serie de operaciones en la base de datos para calcular el resultado deseado. En los **lenguajes no procedimentales** el usuario describe la información deseada sin dar un procedimiento concreto para obtener esa información.

La mayor parte de los sistemas comerciales de bases de datos relacionales ofrecen un lenguaje de consulta que incluye elementos de los enfoques procedimental y no procedimental. Se estudiarán varios lenguajes comerciales en el Capítulo 4. El Capítulo 5 trata los lenguajes QBE y Datalog, este último parecido a Prolog.

En este capítulo se examinarán los lenguajes «puros»: el álgebra relacional es procedimental, mientras que el cálculo relacional de tuplas y el de dominios son no procedimentales. Estos lenguajes de consulta son rígidos y formales, y carecen del «azúcar sintáctico» de los lenguajes comerciales, pero ilustran las técnicas fundamentales para la extracción de datos de las bases de datos.

Aunque inicialmente sólo se estudiarán las consultas, un lenguaje de manipulación de datos completo no sólo incluye un lenguaje de consulta, sino también un lenguaje para la modificación de las bases de datos. Estos lenguajes incluyen órdenes para insertar y borrar tuplas, así como órdenes para modificar partes de las tuplas existentes. Las modificaciones de las bases de datos se examinarán después de completar la discusión sobre las consultas.

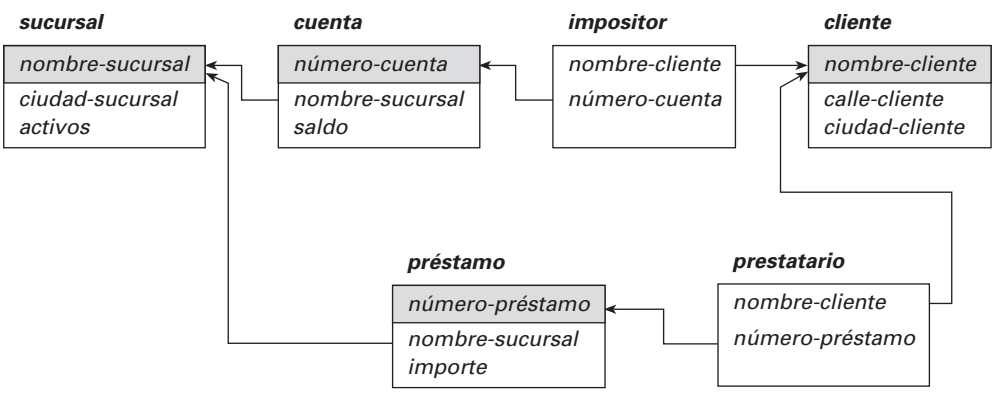

FIGURA 3.9. Diagrama de esquema para el banco.