Capítulo

1

Introducción a las bases de datos

Objetivos del capítulo:

En este capítulo aprenderá:

- Algunos usos comunes de los sistemas de bases de datos.
- Las características de los sistemas basados en archivos.
- Los problemas asociados a la técnica basada en archivo.
- El significado del término 'base de datos'.
- El significado del término 'sistema de gestión de bases de datos' (SGBD).
- Las funciones típicas de un SGBD.
- Los componentes principales del entorno SGBD.
- El personal implicado en el entorno SGBD.
- La historia del desarrollo de los SGBD.
- Las ventajas y desventajas de los SGBD.

> La historia de la investigación en sistemas de bases de datos es de una excepcional productividad y ha tenido un impacto económico extraordinario. Con apenas 20 años de antigüedad como campo de investigación científica, la investigación en bases de datos ha permitido el surgimiento de una industria de los servicios de información que sólo en los Estados Unidos factura unos 10.000 millones de dólares por año. Los diversos logros en la investigación relativa a las bases de datos han permitido realizar avances fundamentales en los sistemas de comunicaciones, transporte y logística, gestión financiera, sistemas de gestión del conocimiento, accesibilidad a la literatura técnica y una miríada de otras aplicaciones tanto civiles como militares. También han servido como base para permitir un progreso considerable en diversos campos básicos de la ciencia, desde la informática a la biología.
>
> (Silberschatz *et al*., 1990, 1996)

Esta cita está tomada de una conferencia sobre sistemas de bases de datos celebrada a principios de la década de 1990 y que tuvo su continuación en otra conferencia posterior en 1996, y nos sirve como motivación para el estudio del tema de este libro: los **sistemas de bases de datos**. Desde la celebración de estas conferencias, la importancia de los sistemas de bases de datos se ha incrementado todavía más, con significativos desarrollos en lo que respecta a la capacidad del hardware, a la funcionalidad de éste y a las comunicaciones, incluyendo la aparición de Internet, del comercio electrónico, de los sistemas de inteligencia empresarial, de las comunicaciones móviles y de la informática reticular. Los sistemas de bases de datos son, posiblemente, el desarrollo más importante en el campo de la ingeniería del software y las bases de datos forman ahora el marco de trabajo fundamental de los sistemas de formación, habiendo cambiado de forma sig-

nificativa la manera en que muchas organizaciones operan. La tecnología de bases de datos constituye un área de enorme interés en la que trabaja y, desde su aparición, ha actuado de catalizador para muchos desarrollos importantes en ingeniería del software. La conferencia a la que aludimos puso de manifiesto que los desarrollos en sistemas de bases de datos no habían terminado, al contrario de lo que algunas personas opinaban. De hecho, parafraseando una antigua frase pronunciada por un político, puede que sólo estemos *al final del principio* de los desarrollos. Las aplicaciones que será preciso soportar en el futuro tienen un grado de complejidad tanto más alto que nos veremos abocados a rehacer muchos de los algoritmos que actualmente se utilizan, como por ejemplo los algoritmos para el almacenamiento y acceso a archivos y para la optimización de consultas. El desarrollo de estos algoritmos originales ha tenido significativas ramificaciones en la ingeniería del software y el desarrollo de nuevos algoritmos tendrá, sin ninguna duda, efectos similares. En este primer capítulo vamos a proporcionar una introducción a los sistemas de bases de datos.

Estructura del capítulo

En la Sección 1.1 vamos a examinar algunas de las aplicaciones de los sistemas de bases de datos que podemos encontrar en nuestra vida cotidiana, pero de las que quizá no seamos conscientes. En las Secciones 1.2 y 1.3 compararemos la antigua técnica basada en archivos para la informatización de sistemas de archivos manuales con la técnica moderna, mucho más adecuada, que se apoya en las bases de datos. En la Sección 1.4 hablaremos de los cuatro tipos de papeles que juegan las personas en un entorno de bases de datos, es decir: administradores de datos y de bases de datos, diseñadores de bases de datos, desarrolladores de aplicaciones y usuarios finales. En la Sección 1.5 proporcionamos una breve historia de los sistemas de bases de datos, la cual está seguida, en la Sección 1.6, por un análisis de las ventajas y desventajas de los sistemas de bases de datos.

A lo largo del libro, ilustraremos los conceptos utilizando un caso de estudio basado en una empresa ficticia de gestión inmobiliaria denominada *DreamHome*. Proporcionamos una descripción detallada de este caso de estudio en la Sección 10.4 y en el Apéndice A. En el Apéndice B se presentan otros casos de estudio que pretenden proporcionar más proyectos de carácter realista con los que pueda practica el lector. Al final de muchos capítulos se incluyen ejercicios basados en estos casos de estudio.

1.1 Introducción

Las bases de datos forman hoy en día una parte integrante de nuestra vida cotidiana, hasta tal punto que muchas veces no somos conscientes de estar usando una base de datos. Para comenzar nuestras explicaciones sobre las bases de datos, vamos a examinar en esta sección algunas explicaciones de este tipo de sistemas. En lo que respecta a las explicaciones siguientes, consideraremos que una *base de datos* es una colección de datos relacionados y que el *Sistema de Gestión de Bases de Datos* (SGBD) es el software que gestiona y controla el acceso a la base de datos. Una *aplicación de bases de datos* es simplemente un programa que interactúa con la base de datos en algún punto de su ejecución. También utilizaremos el término más inclusivo *sistema de base de datos* para referirnos a una colección de programas de aplicación que interactúan con la base de datos, junto con el SQL y la propia base de datos. En la Sección 1.3 proporcionaremos definiciones más precisas.

Las compras en el supermercado

Cuando se compra cualquier tipo de producto en el supermercado local, lo más probable es que se esté accediendo a una base de datos. El cajero utiliza un lector de códigos de barras para introducir en el sistema cada una de las compras. Este proceso de introducción de datos está enlazado con un programa de aplicación que utiliza el código de barras para averiguar el precio del elemento, que se extrae de una base de datos de productos. El programa reduce a continuación el número de elementos existentes y muestra el precio en la pantalla del cajero. Si el número de productos existentes en almacén cae por debajo de un umbral especificado, el sistema de bases de datos puede emitir automáticamente un pedido para tener más existencias de dicho producto. Si un cliente telefonea al supermercado, un asistente puede comprobar si un cierto producto está dis-

ponible en almacén ejecutando el programa de aplicación que determina la disponibilidad a partir de la base de datos.

Compras utilizando una tarjeta de crédito

Cuando se compran productos utilizando la tarjeta de crédito, el cajero comprueba normalmente si el cliente tiene disponible el crédito suficiente como para realizar la compra. Esta comprobación puede llevarse a cabo telefónicamente o realizarse automáticamente mediante un lector de tarjetas conectado a un sistema informático. En cualquiera de los dos casos, en algún sitio existe una base de datos que contiene información acerca de las compras que el cliente ha realizado con su tarjeta de crédito. Para comprobar el nivel de crédito existente, hay un programa de aplicación que utiliza el número de la tarjeta de crédito para comprobar el precio de los productos que se pretende adquirir, añadiéndolo al total de las compras que se hayan realizado este mes y luego comprobando si el nuevo total es inferior al límite de crédito predefinido. Una vez confirmada la compra, se añaden los detalles de la misma a esa base de datos. El programa de aplicación también accede a la base de datos para comprobar que la tarjeta de crédito no se encuentre incluida en la lista de tarjetas robadas o perdidas, antes de realizar la compra. Hay otros programas de aplicación que se encargan de enviar resúmenes mensuales a cada propietario de tarjeta de crédito y de cancelar las cuentas pendientes cuando se reciben los pagos.

Reserva de un programa de vacaciones en una agencia de viajes

Cuando se pide información acerca de un programa de vacaciones, la agencia de viajes puede acceder a diversas bases de datos que contienen los detalles sobre dichos programas y sobre los vuelos existentes. Cuando se reserva un cierto programa, el sistema de base de datos tiene que realizar todas las reservas necesarias. En este caso, el sistema debe garantizar que no haya dos agentes distintos que reserven el mismo programa de vacaciones y asegurarse de que no se vendan más billetes de avión que los asientos disponibles en un vuelo. Por ejemplo, si sólo queda un asiento en un vuelo de Londres a Nueva York y dos agentes tratan de reservar ese último asiento al mismo tiempo, el sistema debe detectar dicha situación, permitiendo que una de las reservas se confirme e informando al otro agente de que ya no queda ningún asiento disponible. La agencia de viajes puede disponer también de otra base de datos, usualmente separada, para el tema de facturación.

Utilización de la biblioteca local

Cualquier biblioteca local dispone, probablemente, de una base de datos que contiene los detalles de los libros disponibles en la biblioteca, la información sobre los lectores, sobre las reservas, etc. Existirá un índice informatizado que permita a los lectores encontrar un libro a partir de su título, de sus autores o de su tema. El sistema de bases de datos se encarga de gestionar las reservas para permitir que un lector reserve un libro y de informarle por correo cuando éste esté disponible. El sistema también envía recordatorios a las personas que tienen un libro prestado cuando éstas no lo devuelven en la fecha prevista. Normalmente, el sistema dispondrá de un lector de códigos de barras similar al que se utiliza en los supermercados, códigos de barras que se emplean para controlar los libros que entran y salen de la biblioteca.

Contratación de un seguro

Cada vez que se quiere contratar un seguro, ya sea un seguro personal, un seguro doméstico o un seguro para automóviles, el agente puede acceder a diversas bases de datos que contienen la información de precios correspondiente a diversas instituciones aseguradoras. Los detalles personales que el cliente proporciona, como su nombre, su dirección, edad y si es o no fumador o bebedor, son utilizados por el sistema de bases de datos para determinar el coste del seguro. El agente puede tratar de buscar en diversas bases de datos para hallar la empresa aseguradora que proporcione el mejor precio.

Alquiler de un vídeo

Cuando se desea alquilar una película en un videoclub, lo más probable es que la empresa propietaria del videoclub mantenga una base de datos compuesta por los títulos de las películas que tiene en almacén, por deta-

lles de las copias existentes de cada película y por información de si la copia está disponible para su alquiler o ya ha sido alquilada. La base de datos incluirá también detalles de los miembros del videoclub (las personas que alquilan las películas) y de los títulos que tienen prestados y la fecha en que deben devolverlos. La base de datos puede incluso almacenar información más detallada sobre cada película, como por ejemplo su director o sus actores y la empresa puede usar esta información para monitorizar el uso que se hace de los productos existentes en almacén y para predecir las tendencias futuras de alquiler basándose en los datos históricos.

Utilización de Internet

Muchos de los sitios de Internet funcionan utilizando aplicaciones de bases de datos. Por ejemplo, podemos visitar una librería en línea que permita consultar y comprar libros, como por ejemplo Amazon.com. Dicha librería nos permiten consultar los libros disponibles en diferentes categorías, como por ejemplo informática o gestión empresarial, o los libros escritos por un determinado autor. En cualquier caso, existe una base de datos en el servidor web de la empresa que está compuesta por los detalles correspondientes a cada libro, por la información de disponibilidad, por la información de envíos, por los niveles de existencias y por la información de pedidos. Los detalles de los libros que se almacenan en la base de datos incluyen el título del libro, el ISBN, el autor, el precio, el historial de ventas, el editor, las reseñas y una descripción detallada. La base de datos permite establecer referencias cruzadas entre los libros. Por ejemplo, un libro puede aparecer en varias categorías distintas, como por ejemplo informática, lenguajes de programación, libros más vendidos y títulos recomendados. Las referencias cruzadas también permiten a Amazon proporcionar información sobre otros libros que se suelen pedir junto con el título en el que el cliente está interesado en ese momento.

Al igual que sucedía en un ejemplo anterior, podemos proporcionar los detalles de nuestra tarjeta de crédito para comprar uno o más libros en línea. Amazon.com personaliza su servicio para los clientes que acuden de forma repetitiva a su sitio web, manteniendo un registro de todas las transacciones, incluyendo los elementos adquiridos y los detalles de envío y de la tarjeta de crédito. Cuando el cliente vuelve al sitio web, el sistema puede saludarle utilizando su nombre y presentarle una lista de títulos recomendados que está basada en las compras anteriores.

Estudio en una universidad

Si el lector es estudiante en una universidad, existirá un sistema de bases de datos que contenga información personal sobre él, información sobre el curso en el que está matriculado, detalles sobre las posibles becas existentes, información sobre los cursos que se hayan seguido en años anteriores y detalles sobre los resultados de los exámenes. También puede haber una base de datos que contenga información sobre los estudiantes admitidos para el curso siguiente y otra con detalles sobre el personal que trabaja en la universidad, en la que se incluirán tanto información personal como información salarial, para la posterior elaboración de las nóminas.

1.2 Sistemas tradicionales basados en archivos

Resulta casi una tradición que los libros de bases de datos introduzcan los sistemas de bases de datos mediante una revisión de sus predecesores, los sistemas basados en archivos. Nosotros vamos también a ajustarnos a esta tradición porque, aunque los sistemas basados en archivos pueden considerarse obsoletos, siguen existiendo buenas razones para analizarlos:

- Si comprendemos los problemas inherentes a los sistemas basados en archivos, podemos evitar repetir esos problemas en los sistemas de bases de datos. En otras palabras, resulta conveniente que aprendamos de nuestros anteriores errores. En realidad, no resulta justo emplear la palabra ‘errores’, ya que parece que estamos quitando valor a un trabajo que sirvió a un propósito útil durante muchos años. En realidad, es mucho lo que hemos aprendido de ese trabajo y gracias a él hemos visto que hay formas mejores de gestionar los datos.
- Si se desea convertir un sistema basado en archivos en un sistema de bases de datos, resulta extremadamente útil, si no esencial, comprender cómo funcionan los sistemas basados en archivos.

1.2.1 La técnica basada en archivos

Sistema basado en archivos Una colección de programas de aplicación que realiza diversos servicios para los usuarios finales, como por ejemplo la producción de informes. Cada programa define y gestiona sus propios datos.

Los sistemas basados en archivos fueron uno de los primeros intentos para informatizar los sistemas de archivo manual con los que todos nosotros estamos familiarizados. Por ejemplo, puede crearse un archivo manual en una organización para albergar toda la correspondencia externa e interna relativa a un proyecto, a un producto, a una tarea, a un cliente o a un empleado. Normalmente, existen muchos de dichos archivos, los cuales es preciso etiquetar y almacenar en una o más caja o contenedores por cuestiones de seguridad. Los lugares donde se almacenen esos archivos pueden disponer de llave, también por cuestiones de seguridad, o pueden estar ubicados en áreas seguras del edificio. En nuestra propia casa, probablemente dispongamos de algún tipo de sistema de archivo en el que depositamos los recibos, las facturas, la información bancaria, los contratos de seguros, etc. Si necesitamos consultar algo, vamos a nuestro archivo personal y buscamos en él, de principio a fin, hasta encontrar la información deseada. Alternativamente, puede que dispongamos de un sistema de indexación que nos ayude a localizar más rápidamente lo que queremos. Por ejemplo, puede que tengamos subdivisiones en el sistema de archivo o carpetas separadas para los diferentes elementos que están *relacionadas de alguna manera lógica* entre sí.

Los sistemas de archivo manual funcionan bien cuando el número de elementos almacenados es pequeño. También puede funcionar de forma adecuada cuando hay un gran número de elementos y lo único que necesitamos es almacenarlos o extraerlos. Sin embargo, los sistemas manuales de archivo dejan de ser útiles cuando tenemos que establecer referencias cruzadas o procesar la información contenida en los documentos. Por ejemplo, una agencia inmobiliaria típica podría disponer de un archivo separado para cada inmueble que desee vender o alquilar, para cada potencial comprador o inquilino y para cada miembro de su personal. Piense en el enorme esfuerzo que se requeriría para responder a las siguientes cuestiones:

- ¿Qué viviendas de tres dormitorios hay disponibles para la venta que tengan jardín y garaje?
- ¿Qué apartamentos existen para alquilar a menos de 5 km del centro de la ciudad?
- ¿Cuál es el precio medio de alquiler para un piso de dos dormitorios?
- ¿Cuál es el importe total de la nómina anual de todo el personal?
- ¿Cuál es la comparación entre los ingresos del último mes y los ingresos previstos para este?
- ¿Cuales son los ingresos mensuales previstos para el próximo año?

Hoy en día, los clientes, los gestores de las empresas y el personal de las mismas quieren cada vez más información. En algunas áreas, existe la obligación legal de generar informes mensuales, trimestrales y anuales detallados. Claramente, los sistemas manuales no resultan adecuados para este tipo de tarea. Los sistemas basados en archivos fueron desarrollados para dar respuesta a la necesidad que las empresas tenían de acceder de forma más eficiente a los datos. Sin embargo, en lugar de establecer un sistema centralizado de gestión de los datos operacionales de las organizaciones, lo que se hizo fue adoptar un enfoque descentralizado, en el que cada departamento, con la ayuda de personal especializado en **procesamiento de datos**, almacenaba y controlaba sus propios datos. Para comprender las implicaciones de esto, vamos a utilizar el ejemplo de *DreamHome*.

El departamento de ventas (Sales) es responsable de la venta y alquiler de los inmuebles. Por ejemplo, cada vez que un cliente contacta con el departamento de ventas con la intención de vender o alquilar un inmueble, se rellena un formulario similar al que se muestra en la Figura 1.1(a). Esto proporciona diversos detalles sobre el inmueble, como la dirección del número de habitaciones, junto con la información personal acerca del propietario. El departamento de ventas también gestiona las consultas de los clientes interesados en comprar o alquilar un inmueble, rellenándose para cada uno un formulario similar al que se muestra en la Figura 1.1(b). Con la asistencia del departamento de procesos de datos, el departamento de ventas crea un sistema de información para gestionar el alquiler de los inmuebles. El sistema está compuesto por tres

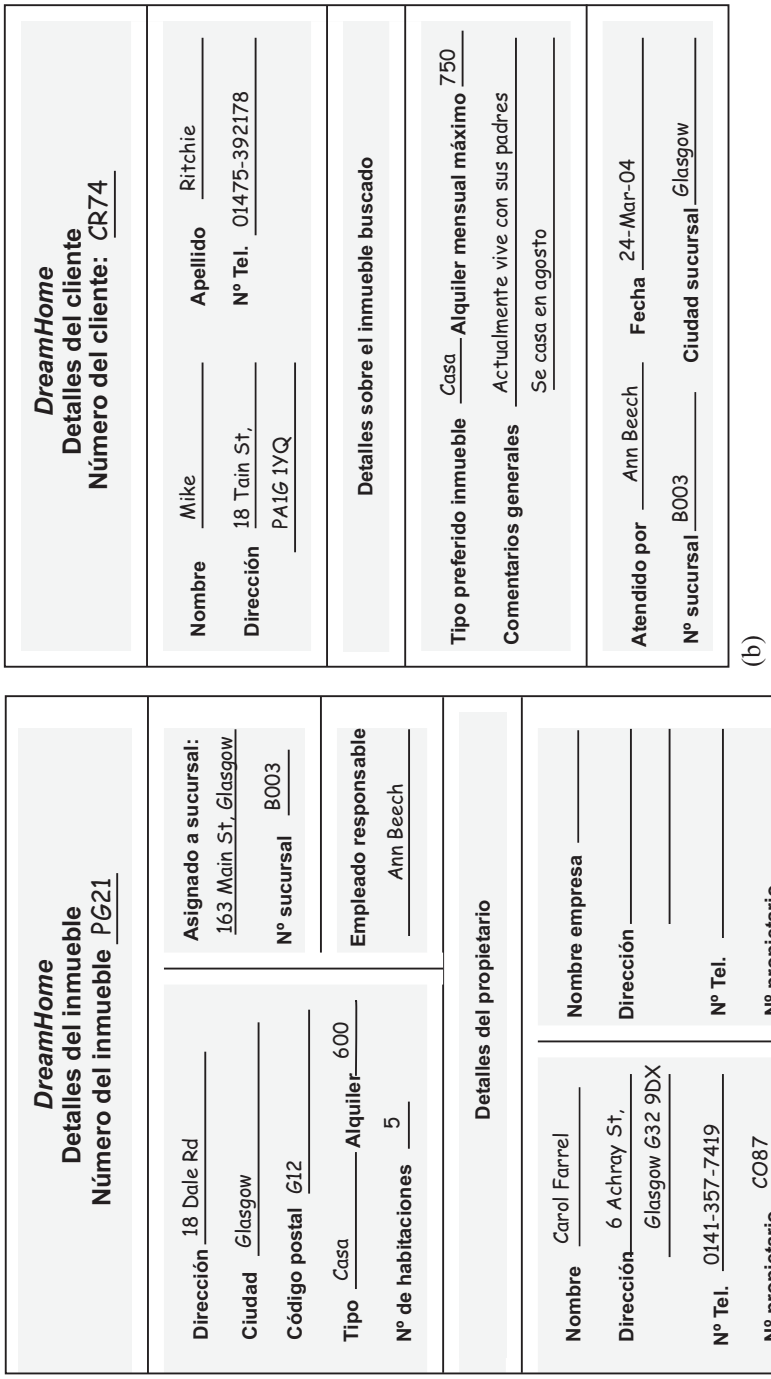


Figura 1.1. Formularios del departamento de ventas: (a) formulario de detalles para inmuebles en alquiler; (b) formulario de detalles de los clientes.

archivos que contienen los detalles sobre los inmuebles, sobre el propietario y sobre el cliente, como se ilustra en la Figura 1.2. Por simplicidad, vamos a omitir los detalles relativos al personal, a las sucursales de la empresa y a los propietarios de la misma.

PropertyForRent

propertyNo	street	city	postcode	type	rooms	rent	ownerNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93

PrivateOwner

ownerNo	fName	lName	address	telNo
CO46	Joe	Keogh	2 Fergus Dr, Aberdeen AB2 7SX	01224-861212
CO87	Carol	Farrel	6 Achray St, Glasgow G32 9DX	0141-357-7419
CO40	Tina	Murphy	63 Well St, Glasgow G42	0141-943-1728
CO93	Tony	Shaw	12 Park Pl, Glasgow G4 0QR	0141-225-7025

Client

clientNo	fName	lName	address	telNo	prefType	maxRent
CR76	John	Kay	56 High St, London SW1 4EH	0207-774-5632	Flat	425
CR56	Aline	Stewart	64 Fern Dr, Glasgow G42 0BL	0141-848-1825	Flat	350
CR74	Mike	Ritchie	18 Tain St, PA1G 1YQ	01475-392178	House	750
CR62	Mary	Tregear	5 Tarbot Rd, Aberdeen AB9 3ST	01224-196720	Flat	600

Figura 1.2. Archivos PropertyForRent, PrivateOwner y Client utilizados por el departamento de ventas.

El departamento de contratos (Contracts) es responsable de gestionar los contratos de alquiler relativos a los distintos inmuebles. Cuando un cliente accede a alquilar un inmueble, un miembro del departamento de ventas rellena un formulario en el que se indican los detalles relativos al cliente y al inmueble, como se muestra en la Figura 1.3. Este formulario es enviado al departamento de contratos, que le asigna un número de referencia y termina de rellenar los detalles relativos al pago y al periodo de alquiler. De nuevo, con la ayuda del departamento de proceso de datos, el departamento de contratos crea un sistema de información para gestionar estos contratos de alquiler. El sistema está compuesto por tres archivos donde se almacenan los detalles relativos al contrato de alquiler, al inmueble y al cliente y donde se introducen datos similares a los que ya posee el departamento de ventas, como se ilustra en la Figura 1.4.

La situación completa se ilustra en la Figura 1.5. En ella podemos ver que cada departamento accede a sus propios archivos utilizando programas de aplicación escritos especialmente para ellos. Cada conjunto de programas de aplicación departamentales se encarga de gestionar la introducción de datos, el mantenimiento de los archivos y la generación de un conjunto fijo de informes específicos. Además, lo cual tiene mayor importancia, la estructura física y el almacenamiento de los archivos y registros de datos están definidos por el código de aplicación.

Podemos encontrar ejemplos similares en otros departamentos. Por ejemplo, el departamento de nóminas (Payroll) almacena información relativa al salario de cada miembro del personal, es decir:

StaffSalary(StaffNo, fName, lName, sex, salary, branchNo)

El departamento de personal (Personnel) también almacena información sobre los miembros del personal, es decir:

Staff(staffNo, fName, lName, position, sex, dateOfbirth, salary, branchNo)

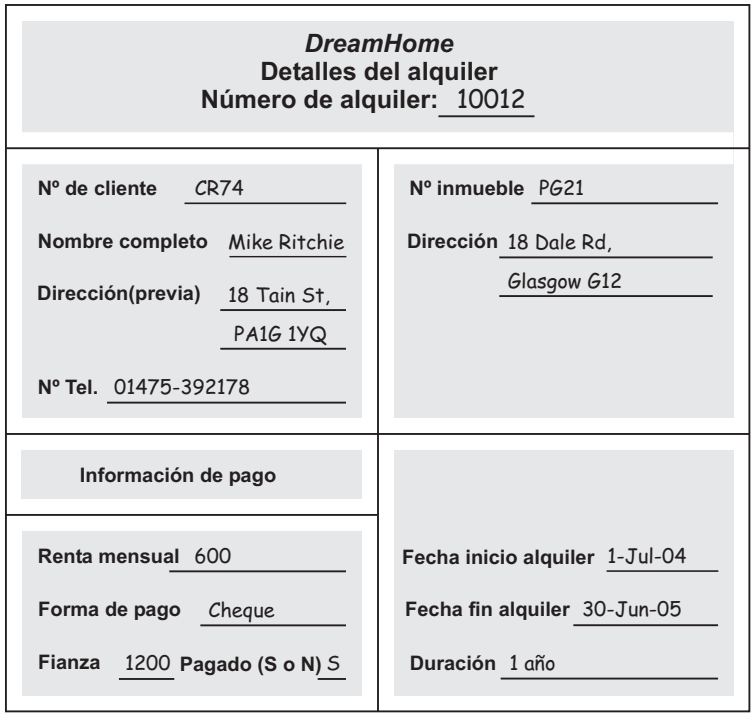


Figura 1.3. Formulario de alquiler utilizado por el departamento de contratos.

Lease

leaseNo	propertyNo	clientNo	rent	payment Method	deposit	paid	rentStart	rentFinish	duration
10024	PA14	CR62	650	Visa	1300	Y	1-Jun-05	31-May-05	12
10075	PL94	CR76	400	Cash	800	N	1-Aug-05	31-Jan-05	6
10012	PG21	CR74	600	Cheque	1200	Y	1-Jul-05	30-Jun-05	12

PropertyForRent

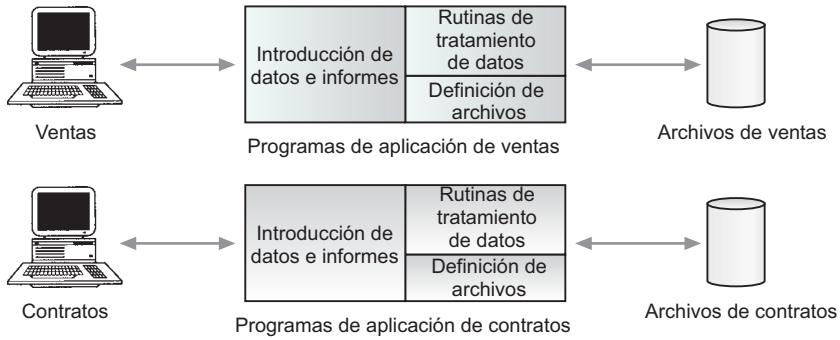
propertyNo	street	city	postcode	rent
PA14	16 Holhead	Aberdeen	AB7 5SU	650
PL94	6 Argyll St	London	NW2	400
PG21	18 Dale Rd	Glasgow	G12	600

Client

clientNo	fName	lName	address	telNo
CR76	John	Kay	56 High St, London SW1 4EH	0171-774-5632
CR74	Mike	Ritchie	18 Tain St, PA1G 1YQ	01475-392178
CR62	Mary	Tregear	5 Tarbot Rd, Aberdeen AB9 3ST	01224-196720

Figura 1.4. Archivos Lease, PropertyForRent y Client utilizados por el departamento de contratos.

Podemos ver claramente que existe una gran cantidad de duplicación de datos en estos departamentos, y esto siempre suele ser así en los sistemas basados en archivos. Antes de analizar las limitaciones de este



Archivos de ventas

PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)

Archivos de contratos

Lease (leaseNo, propertyNo, clientNo, rent, paymentMethod, deposit, paid, rentStart, rentFinish, duration)
PropertyForRent (propertyNo, street, city, postcode, rent)
Client (clientNo, fName, lName, address, telNo)

Figura 1.5. Procesamiento basado en archivos.

enfoque, puede resultar útil comprender la terminología utilizada en los sistemas basados en archivos. Un archivo es simplemente una colección de **registros**, que contienen **datos lógicamente relacionados**. Por ejemplo, el archivo PropertyForRent de la Figura 1.2 contiene seis registros, uno para cada inmueble. Cada registro contiene un conjunto lógicamente relacionado de uno o más **campos**, donde cada campo representa alguna característica del objeto del mundo real que se quiere modelar. En la Figura 1.2 los campos del archivo PropertyForRent representan características de los inmuebles, como su dirección, tipo de inmueble y número de habitaciones.

1.2.2 Limitaciones de la técnica basada en archivos

Esta breve descripción de los sistemas tradicionales basados en archivo debería ser suficiente para explicar las limitaciones de este enfoque. En la Tabla 1.1 se enumeran cinco problemas diferentes.

Tabla 1.1. Limitaciones de los sistemas basados en archivos.

Separación y aislamiento de los datos
Duplicación de los datos
Dependencias entre los datos
Formatos de archivos incompatibles
Consultas fijas/proliferación de programas de aplicación

Separación y aislamiento de los datos

Cuando se aíslan los datos en archivos separados, resulta más difícil acceder a los datos que deben estar disponibles. Por ejemplo, si queremos generar una lista de todos los chalets que satisfacen los requisitos de los clientes, necesitamos primero crear un archivo temporal de aquellos clientes cuyo tipo preferido de inmueble sea 'chalet'. A continuación hay que explorar el archivo PropertyForRent para localizar aquellos inmuebles cuyo tipo sea 'chalet' y cuyo alquiler sea inferior al alquiler máximo fijado por el cliente. Con los sistemas de archi-

vos, este tipo de procesamiento resulta difícil. El desarrollador de aplicaciones debe sincronizar el procesamiento de los dos archivos para garantizar que se extraigan los datos correctos. Esta dificultad se hace todavía mayor si se necesita extraer datos de más de dos archivos.

Duplicación de los datos

Debido al enfoque descentralizado adoptado por los departamentos, la técnica basada en archivos, promueve, si es que no requiere, una duplicación incontrolada de los datos. Por ejemplo, en la Figura 1.5 podemos ver claramente que existe una duplicación de la información tanto de los inmuebles como de los clientes en los departamentos de ventas y de contratos. La duplicación descontrolada de los datos resulta indeseable por varias razones, como por ejemplo:

- La duplicación implica un desperdicio de recursos. Cuesta tiempo y dinero introducir los datos más de una vez.
- Se consume un espacio de almacenamiento innecesario, lo que también tiene su coste asociado. A menudo, puede evitarse la duplicación de los datos compartiendo los archivos de datos.
- Lo más importante quizá sea que la duplicación puede conducir a que se pierda la integridad de los datos. En otras palabras, los datos podrían dejar de ser coherentes. Por ejemplo, considere el caso de la duplicación de datos en los departamentos de personal y de nóminas, como hemos descrito anteriormente. Si un empleado cambia de domicilio y ese cambio de dirección sólo se comunica al departamento de personal y no al de nóminas, la nómina de ese empleado podría ser enviada a la dirección incorrecta. Otro problema más grave sucede cuando se asciende a un empleado, incrementándole a su vez el sueldo. De nuevo, si el cambio se notifica al departamento de personal pero no llega al de nóminas, el empleado recibirá una paga incorrecta. Cuando este error se detecte, se necesitará tiempo y esfuerzo para resolverlo. Estos dos ejemplos ilustran el tipo de incoherencia que puede surgir como resultado de la duplicación de los datos. Puesto que los miembros del departamento de personal no disponen de ningún sistema automático para actualizar los datos contenidos en los archivos del departamento de nóminas, no resulta difícil prever que dichas incoherencias terminarán por surgir. Incluso aunque se notifiquen los cambios al departamento de nóminas, es posible que éstos se introduzcan de forma incorrecta.

Dependencias entre los datos

Como ya hemos mencionado, la estructura física y el almacenamiento de los archivos y registros de datos están definidos en el código de la aplicación. Esto significa que resulta difícil realizar cambios a una estructura existente. Por ejemplo, si se incrementa el tamaño del campo PropertyForRent address de 40 a 41 caracteres, parece que dicho cambio debería poder incrementarse de forma simple, pero se requiere la creación de un programa de un solo uso (es decir, un programa que sólo se ejecute una vez y luego pueda descartarse) que convierta el archivo PropertyForRent al nuevo formato. Este programa tiene que:

- abrir el archivo PropertyForRent original para lectura;
- abrir un archivo temporal con la nueva estructura;
- leer un registro del archivo original, convertir los datos para adecuarlos a la nueva estructura y escribirlo en el archivo temporal, repitiendo este paso para todos los registros del archivo original;
- borrar el archivo PropertyForRent original;
- renombrar el archivo temporal, denominándolo PropertyForRent.

Además, todos los programas que accedan al archivo PropertyForRent deberán ser modificados para adecuarlos a la nueva estructura del archivo. Puede que existan muchos de tales programas accediendo al archivo PropertyForRent. Por tanto, el programador necesitará identificar todos los programas afectados, modificarlos y luego volverlos a probar. Observe que los programas ni siquiera tienen que utilizar el campo address para verse afectados: basta con que utilicen el archivo PropertyForRent. Obviamente, todo esto puede requerir un tiempo considerable y se trata de un proceso sujeto a errores. Esta característica de los sistemas basados en archivos se denomina **dependencia entre los programas y los datos**.

Formatos de archivo incompatibles

Puesto que la estructura de los archivos está incrustada en los programas de aplicación, dichas estructuras dependen del leguaje de programación de aplicaciones que se utilice. Por ejemplo, la estructura de un archivo generador por un programa COBOL puede ser diferente de la estructura de un archivo creada por un programa C. La incompatibilidad directa de dichos archivos hace difícil que se los pueda procesar conjuntamente.

Por ejemplo, suponga que el departamento de contratos quiere averiguar los nombres y direcciones de todos los propietarios cuyos inmuebles estén actualmente alquilados. Desafortunadamente, el departamento de contratos no dispone de la información de detalles sobre los propietarios de los inmuebles, ya que sólo el departamento de ventas tiene esa información. Sin embargo, el departamento de contratos sí que dispone del número de inmueble (propertyNo), que puede utilizarse para localizar el número de inmueble correspondiente en el archivo PropertyForRent del departamento de ventas. Este archivo contiene el número de propietario (ownerNo), que puede utilizarse para localizar los detalles del propietario en el archivo PrivateOwner. Los programas del departamento de contratos han sido desarrollados en Cobol, mientras que los del departamento de ventas han sido desarrollados en C. Por tanto, para establecer la correspondencia entre los campos propertyNo de los dos archivos PropertyForRent se necesita que un desarrollador de aplicaciones escriba un programa software que convierta los archivos a un formato común para facilitar el procesamiento. De nuevo, este proceso puede requerir mucho tiempo y resultar muy caro.

Consultas fijas/proliferación de programas de aplicación

Desde el punto de vista del usuario final, los sistemas basados en archivos representaron una enorme mejora con respecto a los sistemas manuales. En consecuencia, las peticiones de nuevas consultas o de modificaciones de las ya existentes comenzaron a crecer. Sin embargo, los sistemas basados en archivos son muy dependientes del desarrollador de aplicaciones, que es quien tiene que escribir todas las consultas e informes requeridos. Como resultado, sucedieron dos cosas, en algunas organizaciones, el tipo de consulta o de informe que podía producirse era fijo. No existía ninguna posibilidad de solicitar consultas no planificadas (es decir, consultas pensadas en el momento o *ad hoc*) ni acerca de los propios datos ni acerca de los datos disponibles.

En otras organizaciones, se produjo una proliferación de archivos y de programas de aplicación. Al final, se alcanzó un punto en el que el departamento de proceso de datos, con sus recursos existentes, era incapaz de gestionar todo el trabajo. Esto normalmente llevaba a que se presionara en exceso al personal del departamento de proceso de datos, lo que daba como resultado programas inadecuados o ineficientes a la hora de satisfacer las demandas de los usuarios, o bien sistemas de documentación excesivamente limitados o bien programas cuyo mantenimiento era muy complicado. A menudo, se tendía a omitir diversos tipos de funcionalidad:

- no se incluían mecanismos de seguridad o integridad;
- la recuperación, para los casos de fallos de hardware o software, era limitada o inexistente;
- el acceso a los archivos estaba restringido, de modo que sólo un usuario podía acceder en cada instante: no había ningún tipo de mecanismo para facilitar el acceso compartido por parte de distintas personas pertenecientes al mismo departamento.

En cualquier caso, lo que está claro es que ese tipo de situación no era aceptable. Se necesitaba algún otro tipo de solución.

1.3 Sistemas de bases de datos

Todas estas limitaciones de los sistemas basados en archivos pueden atribuirse a dos factores distintos:

(1) la definición de los datos está incluida en los programas de aplicación, en lugar de almacenarse de forma separada e independiente;

(2) no existe ningún control sobre el acceso y manipulación de los datos, más allá del que imponen los propios programas de aplicación.

Para poder ser más efectivos, se necesitaba una nueva técnica y lo que surgió fue el concepto de **base de datos** y los **Sistemas de Gestión de Bases de Datos** (SGBD). En esta sección, vamos a proporcionar una definición más formal de estos términos y a examinar los componentes que podemos esperar encontrarnos en un entorno SGBD.

1.3.1 La base de datos

Base de datos Una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, que están diseñados para satisfacer las necesidades de información de una organización.

Examinemos ahora la definición de base de datos para tratar de entender el concepto adecuadamente. Una base de datos es un repositorio centralizado, posiblemente de gran tamaño, compuesto por datos que pueden ser utilizados simultáneamente por múltiples departamentos y usuarios. En lugar de disponer de una serie de archivos desconectados con datos redundantes, todos los elementos de datos están integrados, manteniéndose al mínimo las posibles duplicaciones. La base de datos deja de ser propiedad de un departamento y pasa a ser un recurso corporativo compartido. La base de datos almacena no sólo los datos operacionales de la organización, sino también una descripción de dichos datos. Por esta razón, a veces se suele describir a las bases de datos como una *colección autodescriptiva de registros integrados*. La descripción de los datos se conoce con el nombre de **catálogo del sistema** (o **diccionario de datos** o **metadatos**, es decir, '**datos acerca de los datos**'). Es esta naturaleza autodescriptiva de la bases de datos la que proporciona la **independencia entre programas y datos**.

El enfoque adoptado por los sistemas de bases de datos, en el que la definición de los datos está separada de los programas de aplicación, es similar a la técnica utilizada en el desarrollo moderno de software, donde se proporciona tanto una definición interna de un objeto como una definición externa independiente de la anterior. Los usuarios de un objeto sólo ven la definición externa y no son conscientes del modo en que el objeto está definido ni de su manera de funcionar. Una de las ventajas de esta técnica, denominada **abstracción de datos**, es que podemos modificar la definición interna de un objeto sin afeptar a los usuarios de dicho objeto, siempre y cuando la definición externa continúe siendo la misma. De la misma forma, los sistemas de bases de datos separan la estructura de los datos de los programas de aplicación y almacenan dicha estructura en la propia base de datos. Si se añaden nuevas estructuras de datos o se modifican las existentes, los programas de aplicación no se verán afectados, siempre y cuando no dependan directamente de la información que haya sido modificada. Por ejemplo, si añadimos un nuevo campo a un registro o creamos un nuevo archivo, las aplicaciones existentes no se verán afectadas. Sin embargo, si eliminamos de un archivo un campo utilizado por un programa de aplicación, entonces dicho programa de aplicación sí se verá afectado por el cambio y deberá ser modificado correspondientemente.

El último término en la definición de base de datos que debemos explicar es el 'lógicamente relacionado'. Al analizar las necesidades de información de una organización, tratamos de identificar entidades, atributos y relaciones. Una **entidad** es un objeto distintivo (una persona, lugar, cosa, concepto o suceso) dentro de la organización y que hay que representar en la base de datos. Un **atributo** es una propiedad que describe algún aspecto del objeto que queremos almacenar y una **relación** es una asociación entre entidades. Por ejemplo, la Figura 1.6 muestra un diagrama Entidad-Relación (ER) para una parte del caso de estudio de *DreamHome*. Está compuesto por:

- seis entidades (los rectángulos): Branch (sucursal), Staff (personal), PropertyForRent (inmueble en alquiler) Client (cliente) PrivateOwner (propietario) y Lease (alquiler);
- siete relaciones (los nombres adyacentes a las líneas): *Has* (tiene), *Offers* (ofrece), *Oversees* (gestiona), *Views* (vista), *Owns* (posee), *LeasedBy* (alquilado por) y *Holds* (alquila);
- seis atributos, uno para cada entidad: branchNo (número de sucursal), staffNo (número de empleado), propertyNo (número de inmueble), clientNo (número de cliente), ownerNo (número de propietario) y leaseNo (número de contrato de alquiler).

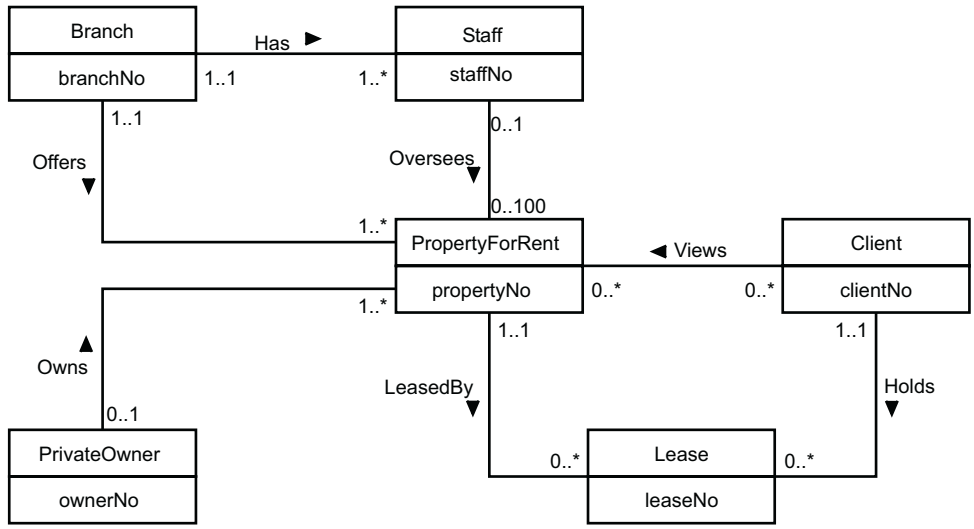


Figura 1.6. Ejemplo del diagrama de entidad-relación.

La base de datos representa las entidades, los atributos y las relaciones lógicas entre entidades. En otras palabras, la base de datos almacena un conjunto de datos que están lógicamente relacionados. Hablaremos en detalle del modelo entidad-relación en los Capítulos 11 y 12.

1.3.2 Sistema de gestión de base de datos (SGBD)

SGBD Un sistema software que permite a los usuarios definir, crear, mantener y controlar el acceso a la base de datos.

El SGBD es el software que interactúa con los programas de aplicación del usuario y con la base de datos. Normalmente, un SGBD proporciona la siguiente funcionalidad:

- Permite a los usuarios definir la base de datos, usualmente mediante un **lenguaje de definición de datos** (DDL, Data Definition Language). El DDL permite a los usuarios especificar las estructuras y tipos de datos y las restricciones aplicables a los datos que hay que almacenar en la base de datos.
- Permite a los usuarios insertar, actualizar, borrar y extraer datos de la base de datos, usualmente mediante un **lenguaje de manipulación de datos** (DML, Data Manipulation Language). Al disponer de un repositorio centralizado para todos los datos y las descripciones de los datos, el lenguaje DML puede proporcionar un mecanismo general de consulta de esos datos, denominado **lenguaje de consulta**. La existencia de un lenguaje de consulta resuelve el problema de los sistemas basados en archivos en los que el usuario tenía que trabajar con un conjunto fijo de consultas, o bien en los que existía una proliferación de programas que provocaban graves problemas de gestión del software. El lenguaje de consulta más común es el lenguaje SQL (**Structured Query Language**, lenguaje estructurado de consulta), que es ahora tanto el estándar formal como el estándar *de facto* para los SGBD relacionales. Para recalcar la importancia del SQL, hemos dedicado los Capítulos 5 y 6, la mayor parte del Capítulo 28 y el Apéndice E a un estudio en profundidad de este lenguaje.
- Proporciona un acceso controlado a la base de datos. Por ejemplo, puede proporcionar:
  - un sistema de seguridad, que evita que los usuarios no autorizados accedan a la base de datos;
  - un sistema de integridad, que mantiene la coherencia de los datos almacenados;
  - un sistema de control de concurrencia que permite el acceso compartido a la base de datos;
  - un sistema de control de recuperación, que restaura la base de datos a un estado previo coherente después de cada fallo hardware o software;

- un catálogo accesible por el usuario, que contiene descripciones de los datos que están almacenados en la base de datos.

1.3.3 Programa de aplicación

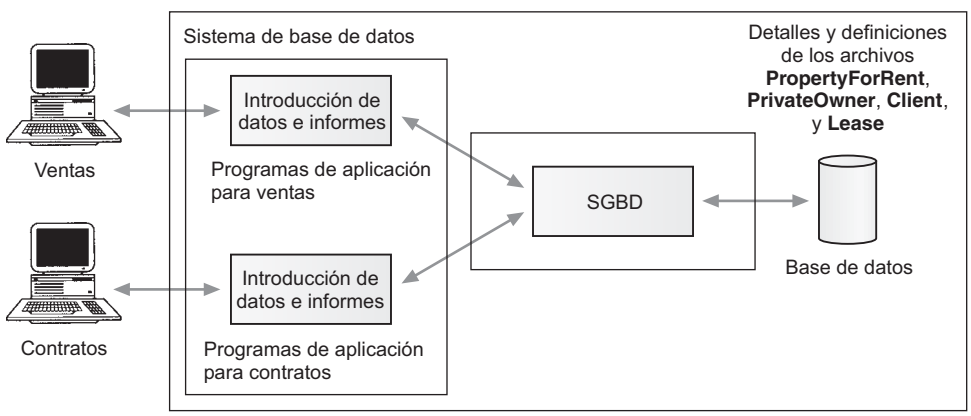
Programa de aplicación Un programa informático que interactúa con la base de datos emitiendo las apropiadas solicitudes (normalmente una instrucción SQL) dirigidas al SGBD.

Los usuarios interactúan con la base de datos mediante una serie de **programas de aplicación** que se utilizan para crear y mantener la base de datos y para generar información. Estos programas pueden ser programas de procesamiento por lotes convencionales o, lo que resulta más habitual hoy en día, aplicaciones en línea. Los programas de aplicación pueden estar escritos en algún leguaje de programación o en un lenguaje de cuarta generación de mayor nivel.

La Figura 1.7 ilustra la técnica de bases de datos, utilizando los archivos indicados en la Figura 1.5. La figura muestra cómo los departamentos de ventas y de contratos utilizan sus programas de aplicación para acceder a la base de datos a través del SGBD. Cada conjunto de programas de aplicación departamentales gestiona la introducción de datos, el mantenimiento de los mismos y la generación de informes. Sin embargo, si lo comparamos con la técnica basada en archivos, la estructura física y el almacenamiento de los datos ahora son responsabilidad del SGBD.

Vistas

Con esta funcionalidad, el SGBD es una herramienta extremadamente potente y útil. Sin embargo, como a los usuarios finales no les interesa demasiado si una determinada tarea resulta sencilla o compleja para el sistema, podría argumentarse que los SGBD han hecho que las cosas se compliquen, ya que ahora los usuarios ven más datos de los que quieren o necesitan. Por ejemplo, los detalles que el departamento de contratos quiere ver en lo que respecta a un inmueble en alquiler, como se indica en la Figura 1.5, han cambiado en la técnica que utiliza una base de datos, como se muestra en la Figura 1.7. Ahora la base de datos también almacena el tipo de inmueble, el número de habitaciones y los detalles referidos al propietario. Para hacer frente a este problema, un SGBD proporciona otra funcionalidad denominada **mecanismo de vistas** que permite que cada usuario disponga de su propia vista de la base de datos (una **vista** es, en esencia, un cierto subconjunto de la base de datos). Por ejemplo, podríamos definir una lista que permita que el departamento de contratos sólo vea los datos que les interesan referidos a los inmuebles en alquiler.



PropertyForRent (propertyNo, street, city, postcode, type, rooms, rent, ownerNo)
PrivateOwner (ownerNo, fName, lName, address, telNo)
Client (clientNo, fName, lName, address, telNo, prefType, maxRent)
Lease (leaseNo, propertyNo, clientNo, paymentMethod, deposit, paid, rentStart, rentFinish)

Figura 1.7. Procesamiento con bases de datos.

Además de reducir la complejidad al permitir que los usuarios vean los datos en la forma que desean verlos, las vistas tienen otras diversas ventajas:

- *Las vistas proporcionan un cierto nivel de seguridad*. Pueden configurarse las vistas para excluir aquellos datos que algunos usuarios no deban ver. Por ejemplo, podemos crear una lista que permita que los gerentes de sucursal y el departamento de nóminas vean todos los datos referidos al personal, incluyendo los detalles salariales, y una segunda vista que utilizaría el resto del personal y de la que se excluirían esos detalles salariales.
- *Las vistas proporcionan un mecanismo para personalizar la apariencia de la base de datos*. Por ejemplo, el departamento de contratos podría denominar al campo de alquiler mensual (rent) utilizando un nombre más conveniente, como por ejemplo Monthly Rent.
- *Una vista puede presentar una imagen coherente y estática de la estructura de la base de datos*, aún cuando se modifique la base de datos subyacente (por ejemplo, podrían añadirse o eliminarse campos, podrían modificarse relaciones o podrían partirse, reestructurarse o renombrarse los archivos). Si se añaden o eliminan campos de un archivo y estos campos no son requeridos por la vista, ésta no se verá afectada por la modificación. Por tanto, las vistas ayudan a conseguir la independencia entre programas y datos de la que hemos hablado en la sección precedente.

La explicación anterior es de carácter general y el nivel real de funcionalidad ofrecido por un SGBD difiere entre unos productos y otros. Por ejemplo, un SGBD para una computadora personal puede no permitir el acceso compartido concurrente y proporcionar sólo mecanismos limitados de seguridad, integridad y control de recuperación. Sin embargo, los productos SGBD multiusuario modernos y de gran complejidad ofrecen todas las funciones anteriores y mucha otras. Los sistemas modernos son programas software extremadamente complejos compuestos por millones de líneas de código y para los que la documentación está formada por múltiples volúmenes. Esto se debe a la necesidad de proporcionar un programa software que gestione requisitos de naturaleza muy general. Además, la utilización de un SGBD hoy en día suele requerir un sistema que proporcione una fiabilidad prácticamente total y una disponibilidad 24/7 (24 horas al día, 7 días a la semana) incluso en el caso de fallos de hardware o software. Los SGBD están continuamente evolucionándose y expandiéndose para adaptarse a las nuevas necesidades de los usuarios. Por ejemplo, algunas aplicaciones requieren ahora el almacenamiento de imágenes gráficas, vídeo, sonido y otros tipos de información similar. Para poder satisfacer las necesidades de este mercado, es necesario cambiar los SGBD. Resulta previsible que cada vez se vayan recibiendo nuevas funcionalidades, por lo que las funciones de un SGBD nunca serán estáticas. Hablaremos de las funciones básicas proporcionadas por un SGBD en los capítulos posteriores.

1.3.4 Componentes de un entorno SGBD

Podemos identificar cinco componentes principales dentro del entorno SGBD: hardware, software, datos, procedimientos y personas, como se ilustra en la Figura 1.8.

Hardware

El SGBD y las aplicaciones requieren una plataforma hardware sobre la que ejecutarse. El hardware puede ir desde una única computadora personal hasta un único mainframe o una red de computadoras. El hardware concreto dependerá de las necesidad de la organización y del SGBD utilizado. Algunos SGBD sólo se ejecutan sobre una plataforma hardware concreta o sobre un sistema operativo particular, mientras que otros se ejecutan sobre un rango más amplio de plataformas hardware y sistemas operativos. Todo SGBD requiere una cantidad mínima de memoria principal de espacio de disco para poder ejecutarse, pero esta configuración

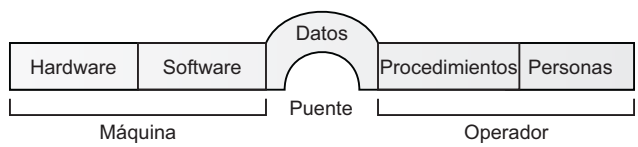


Figura 1.8. Entorno SGBD.

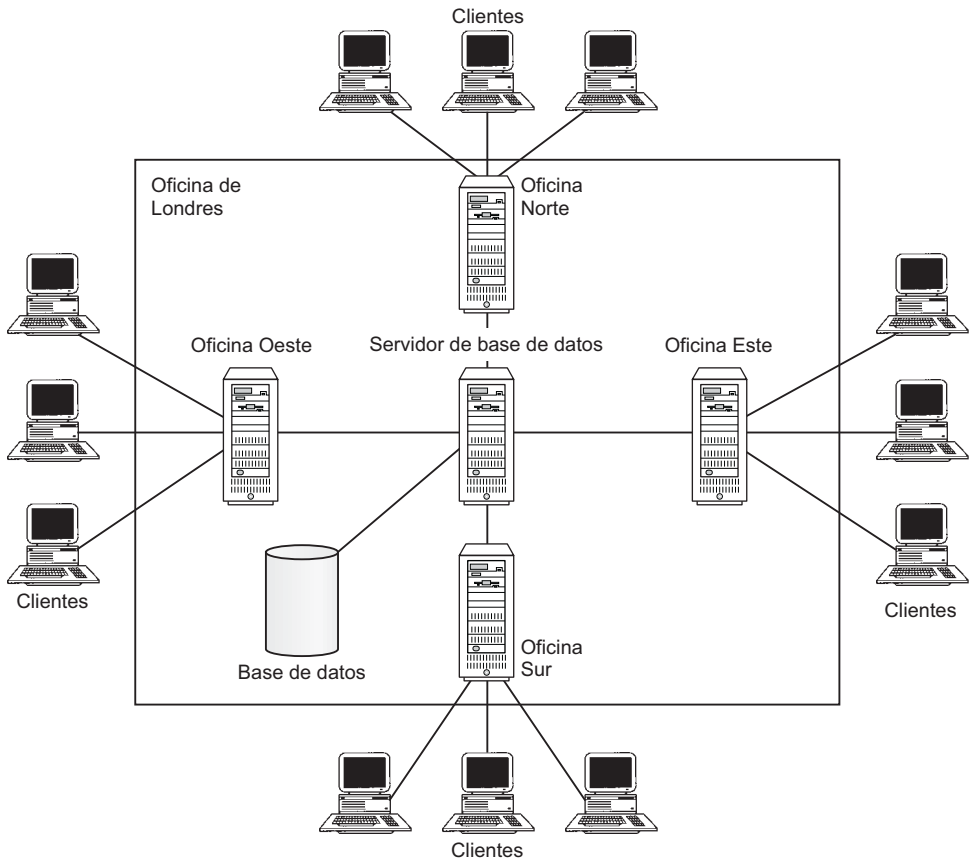


Figura 1.9. Configuración hardware para DreamHome.

mínima puede no necesariamente proporcionar un rendimiento aceptable. En la Figura 1.9 se ilustra una configuración hardware simplificada para *DreamHome*. Está compuesta de una red de minicomputadoras, con una computadora central ubicada en Londres y en la que se ejecuta el sistema de servicio (**backend**) del SGBD, es decir, la parte del SGBD que gestiona y controla el acceso a la base de datos. También se muestran varias computadoras en distintas ubicaciones en las que se ejecuta el sistema de interfaz (**frontend**) del SGBD, es decir, la parte del SGBD que implementa la interfaz con el usuario. Este tipo de arquitectura se denomina **cliente-servidor**: el backend es el servidor y el frontend es el cliente. Hablaremos de este tipo de arquitectura en la Sección 2.6.

Software

El componente software comprende el propio software SGBD y los programas de aplicación, junto con el sistema operativo, que incluye el software de red si el SGBD se está utilizando en una red. Normalmente, los programas de aplicación se escriben en un lenguaje de aplicación de tercera generación (3GL), como C, C++, Java, Visual Basic, COBOL, Fortran, Ada o Pascal, o utilizando un lenguaje de cuarta generación (4GL) como SQL, incrustado dentro de un lenguaje de tercera generación. El SGBD objetivo puede disponer de sus propias herramientas de cuarta generación que permitan el desarrollo rápido de aplicaciones gracias a la existencia de lenguajes de consulta no procedimentales, generadores de informes, generadores de formularios, generadores gráficos y generadores de aplicaciones. La utilización de herramientas de cuarta generación puede mejorar de forma significativa la productividad y dar como resultado programas que son más fáciles de mantener. Hablaremos de las herramientas de cuarta generación en la Sección 2.2.3.

Datos

Quizá el componente más importante de un entorno SGBD, al menos desde el punto de vista de los usuarios finales sean los datos. En la Figura 1.8 podemos observar que los datos actúan como una especie de puente entre los componentes ligados a la máquina y los componentes ligados al operador humano. La base de datos contiene tanto los datos operacionales como los metadatos, es decir, los 'datos acerca de los datos'. La estructura de la base de datos se denomina **esquema**. En la Figura 1.7, el esquema está compuesto por cuatro archivos o **tablas**, que son: PropertyForRent, PrivateOwner, Client y Lease. La tabla PropertyForRent tiene ocho campos o **atributos**, a saber: propertyNo, street, city, postcode, type (el tipo de inmueble), rooms (el número de habitaciones), rent (el alquiler mensual) y ownerNo. El atributo ownerNo modela la relación entre PropertyForRent y PrivateOwner: es decir, el propietario posee (*Owns*) un inmueble para alquiler, como se muestra en el diagrama entidad-relación de la Figura 1.6. Por ejemplo, en la Figura 1.2 podemos observar que el propietario CO46, Joe Keogh, posee el inmueble PA14. Los datos incorporan también el catálogo del sistema, del que hablaremos en detalle en la Sección 2.4.

Procedimientos

Los procedimientos son las instrucciones y reglas que gobiernan el diseño y utilización de la base de datos. Los usuarios del sistema y el personal que gestiona la base de datos requieren una serie de procedimientos documentados que les permitan saber cómo utilizar o ejecutar el sistema. Estos procedimientos pueden estar compuestos de instrucciones que les digan cómo:

- iniciar una sesión en el SGBD;
- utilizar una funcionalidad concreta del SGBD o un programa de aplicación;
- iniciar y detener el SGBD;
- realizar copias de seguridad de la base de datos;
- gestionar los fallos de hardware o de software. Esto puede incluir procedimientos para identificar el componente fallido, para reparar el componente fallido (por ejemplo, telefonear al ingeniero de hardware apropiado) y, después de la reparación del fallo, para recuperar la base de datos;
- cambiar la estructura de una tabla, reorganizar la base de datos entre múltiples discos, mejorar el rendimiento o archivar los datos en un almacenamiento secundario.

Personas

El componente final son las personas que se relacionan con el sistema. Hablaremos de este componente fundamental en la Sección 1.4.

1.3.5 Diseño de bases de datos: un cambio en el paradigma

Hasta ahora, hemos dado por supuesto que los datos de la base de datos tenían una estructura. Por ejemplo, en la Figura 1.7 hemos identificado cuatro tablas: PropertyForRent, PrivateOwner, Client y Lease. Pero, ¿cómo obtenemos esta estructura? La respuesta es muy simple: la estructura de la base de datos se determina durante el **diseño de la base de datos**. Sin embargo, realizar el diseño de una base de datos puede ser extremadamente complejo. Para producir un sistema que satisfaga las necesidades de información de una organización, se necesita un enfoque distinto del utilizado en los sistemas basados en archivo, donde el trabajo estaba dirigido por las necesidades de los departamentos individuales en términos de aplicaciones. Para que el enfoque de base de datos tenga éxito, la organización debe ahora pensar primero en los datos y luego en las aplicaciones. Este cambio en el enfoque se denomina en ocasiones *cambio de paradigma*. Para que el sistema sea aceptable para los usuarios finales, resulta crucial la actividad de diseño de la base de datos. Una base de datos inadecuadamente diseñada generará errores que pueden conducir a que se tomen decisiones incorrectas, lo cual podría tener repercusiones serias para la organización. Por otro lado, una base de datos bien diseñada produce un sistema que proporciona la información correcta para que el proceso de toma de decisiones tenga éxito y funcione de una manera eficiente. El objetivo de este libro es ayudar a llevar a cabo este cambio de

paradigma. Dedicaremos diversos capítulos a la presentación de una metodología completa de diseños de bases de datos (*véanse* los Capítulos 15–18). Dicha metodología se presenta como una serie de pasos fáciles de seguir, proporcionándose directrices a todo lo largo del texto. Por ejemplo, en el diagrama entidad-relación de la Figura 1.6, hemos identificado seis entidades, siete relaciones y seis atributos. Proporcionaremos directrices para ayudar a identificar las entidades, los atributos y relaciones que deben ser representados en la base de datos.

Desafortunadamente, las metodologías de diseño de bases de datos no son muy populares. Muchas organizaciones y diseñadores individuales utilizan bastante poco las metodologías para realizar el diseño de bases de datos, lo cual se considera comúnmente como una de las principales causas de fallo en el desarrollo de sistemas de bases de datos. Debido a la falta de enfoques estructurados del diseño de bases de datos, suelen subestimarse el tiempo y los recursos necesarios para un proyecto de base de datos y las bases de datos desarrolladas resultan inadecuadas o poco eficientes a la hora de satisfacer las demandas de las aplicaciones, o bien la documentación es limitada o puede que el mantenimiento resulte muy difícil.

1.4 Papeles en un entorno de base de datos

En esta sección, vamos a examinar el quinto de los componentes básicos de un entorno SGBD que mencionamos en la sección anterior: las **personas**. Podemos identificar cuatro tipos distintos de personas que pueden participar en un entorno SGBD: administradores de datos y de la base de datos, diseñadores de bases de datos, desarrolladores de aplicaciones y usuarios finales.

1.4.1 Administradores de datos y de la base de datos

La base de datos y el SGBD son recursos corporativos que deben gestionarse igual que cualquier otro recurso. La administración de datos y de la base de datos son papeles que generalmente se asocian con la gestión y control de un SGBD y de los datos en él almacenados. El **administrador de datos** (DA, Data Administrator) es responsable de gestionar los recursos de datos, lo que incluye la planificación de la base de datos, el desarrollo y mantenimiento de estándares, políticas y procedimientos y el diseño procedimental/lógico de la base de datos. El DA consulta con los gerentes de mayor nivel y les aconseja, para garantizar que la dirección seguida por el desarrollo de la base de datos permita soportar los objetivos corporativos.

El **administrador de la base de datos** (DBA, Database Administrator) es responsable de la materialización física de la base de datos, incluyendo la implementación y diseño físicos de la base de datos, el control de la seguridad y de la integridad, el mantenimiento de la fiabilidad del sistema y la garantía de que las aplicaciones exhiban un rendimiento satisfactorio para los usuarios. El papel de un DBA tiene una orientación más técnica que el de DA, requiriéndose un conocimiento detallado del SGBD de destino y del entorno de sistema en el que está implementado. En algunas organizaciones no hay distinción entre estos dos papeles, mientras que en otras la importancia de los recursos corporativos se ve reflejada en la asignación de equipos de personas a cada uno de estos dos papeles. Hablaremos de la administración de datos y de la base de datos con más detalle en la Sección 9.15.

1.4.2 Diseñadores de bases de datos

En los grandes proyectos de diseño de bases de datos, podemos distinguir entre dos tipos de diseñador: los diseñadores lógicos de la base de datos y los diseñadores físicos de la base de datos. Las responsabilidades del **diseñador lógico de la base de datos** son identificar los datos (es decir, las entidades y atributos), las relaciones entre los datos y las restricciones que hay que aplicar a los datos que se almacenen en la base de datos. El diseñador lógico de la base de datos debe tener una comprensión profunda y completa de los datos de la organización y de las restricciones aplicables (las restricciones se denominan en ocasiones **reglas de negocio**). Estas restricciones describen las principales características de los datos, tal como la organización los ve. Como ejemplos de restricciones para *DreamHome* podemos citar:

- un empleado no puede gestionar más de 100 inmuebles para alquiler o venta al mismo tiempo;

- un empleado no puede gestionar la venta o alquiler de un inmueble de su propiedad;
- un mismo representante legal no puede actuar en representación tanto del comprador como del vendedor de un inmueble.

Para poder tener éxito, el diseñador lógico de la base de datos debe implicar a todos los potenciales usuarios de la base de datos en el desarrollo del modelo de datos, y esta implicación debe producirse lo más pronto posible dentro del proceso. En este libro, dividiremos el trabajo del diseñador lógico de la base de datos en dos etapas:

- diseño conceptual de la base de datos, que es independiente de los detalles de implementación, como el SGBD de destino, los programas de aplicación, los lenguajes de programación o cualquier otra consideración física;
- diseño lógico de la base de datos, dirigido a un modelo de datos específico, como por ejemplo el modelo relacional, el modelo en red, el modelo jerárquico o el modelo orientado a objetos.

El **diseñador físico de la base de datos** decide cómo materializar físicamente el diseño lógico de la base de datos. Esto implica:

- establecer la correspondencia entre el diseño lógico de la base de datos y un conjunto de tablas y restricciones de integridad;
- seleccionar estructuras de almacenamiento y métodos de acceso específicos para los datos con el fin de conseguir unas buenas prestaciones;
- diseñar las medidas de seguridad que los datos requieran.

Muchas partes del diseño físico de una base de datos dependen en gran medida del SGBD de destino y puede haber más de una forma de implementar cada mecanismo concreto. Por tanto, el diseñador físico de la base de datos debe conocer a la perfección la funcionalidad del SGBD de destino y puede entender las ventajas y desventajas de cada alternativa para cada implementación concreta. El diseñador físico de la base de datos debe ser capaz de seleccionar una estrategia de almacenamiento adecuada que tenga en cuenta el uso de la base de datos. Mientras que el diseño conceptual y lógico de la base de datos estén relacionados con el *qué*, el diseño físico de la base de datos se preocupa de *cómo*. Se requieren capacidades y conocimientos diferentes, lo que implica en muchas ocasiones utilizar personas distintas. Presentaremos una metodología para el diseño conceptual de bases de datos en el Capítulo 15, mientras que en el Capítulo 16 se presentará el diseño lógico de bases de datos y el diseño físico de las bases de datos será objeto de los Capítulos 17 y 18.

1.4.3 Desarrolladores de aplicaciones

Una vez implementada la base de datos, es necesario implementar también los programas de aplicación que proporcionen la funcionalidad requerida por los usuarios finales. Esto es responsabilidad de los **desarrolladores de aplicaciones**. Normalmente, los desarrolladores de aplicaciones trabajan a partir de una especificación producida por los analistas de sistemas. Cada programa contiene enunciados que exigen al SGBD realizar algún tipo de operación sobre la base de datos. Esto incluye extraer datos, insertarlos, actualizarlos o borrarlos. Los programas pueden estar escritos en un lenguaje de programación de tercera generación o en un lenguaje de cuarta generación, como se ha mencionado en la sección precedente.

1.4.4 Usuarios finales

Los usuarios finales son los 'clientes' de la base de datos, que se diseña, implementa y mantiene precisamente para dar servicio a sus necesidades de información. Los usuarios finales pueden clasificarse de acuerdo con la forma en que utilizan el sistema:

- **Usuarios inexpertos**, que normalmente no son conscientes de la existencia de un SGBD. Acceden a la base de datos mediante programas de aplicación escritos a propósito y que intentan que las operaciones sean lo más simples posible. Estos usuarios invocan las operaciones sobre la base de datos introduciendo comandos simples o seleccionando una serie de opciones en un menú. Esto quiere decir que

no necesitan conocer ningún detalle ni sobre la base de datos, ni sobre el SGBD. Por ejemplo, el cajero de un supermercado utiliza un lector de código de barras para averiguar el precio de un artículo. Sin embargo, se está ejecutando un programa de aplicación que lee el código de barras, consulta el precio del artículo en la base de datos, ajusta el campo de la base de datos que contiene el número de artículos en almacén y muestra el precio en la pantalla.

- **Usuarios avanzados**. En el otro extremo del espectro, los usuarios avanzados están familiarizados con la estructura de la base de datos y con las funcionalidades ofrecidas por el SGBD. Los usuarios finales avanzados pueden utilizar un lenguaje de consulta de alto nivel, como SQL, para llevar a cabo las operaciones requeridas. Algunos usuarios finales avanzados pueden incluso escribir sus propios programas de aplicación para su uso personal.

1.5 Historia de los sistemas de gestión de bases de datos

Ya hemos visto que los predecesores de los SGBD eran los sistemas basados en archivos. Sin embargo, no puede identificarse un punto temporal concreto en el que diera comienzo la técnica de bases de datos y dejaran de utilizarse los sistemas basados en archivos. De hecho, los sistemas basados en archivos continúan existiendo en determinadas áreas específicas. Se ha aducido que los sistemas de gestión de bases de datos tienen sus raíces en el proyecto lunar Apollo de la década de1960, que fue iniciado en respuesta al objetivo establecido por el presidente Kennedy para enviar un hombre a la Luna al final de esa década. En aquel tiempo no había ningún sistema disponible que pudiera utilizarse para gestionar y controlar las enormes cantidades de información que el proyecto iba a generar.

Como resultado, North American Aviation (NAA, ahora Rockwell International), el contratista principal del proyecto, desarrolló un sistema software denominado **GUAM** (Generalized Update Access Method, método generalizado de acceso y actualización). GUAM estaba basado en el concepto de que pueden utilizarse componentes de menor tamaño para formar otros componentes mayores y así sucesivamente, hasta terminar por ensamblar el producto final. Esta estructura, que se asemeja a un árbol invertido, se denomina también **estructura jerárquica**. A mediados de la década de 1960, IBM unió sus fuerzas con NAA para desarrollar GUAM, lo que tuvo como resultado lo que ahora se conoce con el nombre de **IMS** (Information Management System, sistema de gestión de la información). La razón por la que IBM restringió IMS a la gestión de jerarquías de registros era poder utilizar dispositivos de almacenamiento en serie, especialmente las cintas magnéticas, lo cual era un requisito de mercado en aquella época. Posteriormente, dicha restricción desapareció. Aunque fue uno de los SGBD comerciales pioneros, IMS continúa siendo el principal SGBD jerárquico utilizado por la mayoría de las grandes instalaciones de tipo mainframe.

A mediados de la década de 1960, otro desarrollo significativo fue la aparición de **IDS** (Integrated Data Store, almacenamiento integrado de datos) de General Electric. Este trabajo fue liderado por uno de los primeros pioneros de los sistemas de bases de datos, Charles Bachmann. Este desarrollo condujo a un nuevo tipo de sistema de bases de datos denominado SGBD **en red**, que tubo un profundo impacto sobre los sistemas de información de dicha generación. La base de datos en red fue desarrollada parcialmente para resolver la necesidad de presentar relaciones de datos más complejas que las que podían modelarse con las estructuras jerárquicas, además de para tratar de imponer un estándar de base de datos. Para ayudar a establecer dichos estándares, la conferencia **CODASYL** (Conference on Data Systems Languages, conferencia sobre lenguajes de sistemas de datos), en la que participaron representantes del gobierno americano y del mundo empresarial, formó un grupo de trabajo en 1965 que recibió en 1967 el nombre de **DBTG** (Data Base Task Group, grupo de trabajo en bases de datos). La misión encomendada al DBTG era definir especificaciones estándar para un entorno que permitiera la creación de bases de datos y la manipulación de datos. En 1969 se emitió un informe preliminar, completándose el primer informe definitivo en 1971. La propuesta del DBTG identificaba tres componentes:

- el **esquema** de la red, es decir, la organización lógica de la base de datos completa, vista por el DBA, en la que se incluía una definición del nombre de la base de datos, del tipo de cada registro y de los componentes de cada tipo de registro;

- el **subesquema**, es decir, la parte de la base de datos vista por el usuario o por los programas de aplicación.
- un lenguaje de gestión de datos para definir las características y la estructura de los datos y para manipularlos.

De cara a la estandarización, el DBTG especificó tres lenguajes diferentes:

- un **lenguaje de definición de datos** (DDL, Data Definition Language) esquema, que permite al DBA definir los esquemas de base de datos;
- un **DDL** de subesquemas, que permite a los programas de aplicación definir qué parte de la base de datos requieren;
- un **lenguaje de manipulación de datos** (DML, Data Manipulation Language), para manipular los datos.

Aunque el informe no fue adoptado formalmente por ANSI (American National Standards Institute, Instituto nacional de estándares de los Estados Unidos), sí que se desarrollaron diversos sistemas posteriormente de acuerdo con la propuesta del DBTG. Estos sistemas se conocen hoy en día con el nombre de sistemas CODASYL o DBTG. Los sistemas CODASYL y jerárquicos representaron la **primera generación** de sistemas de gestión de bases de datos. En el sitio web de este libro (*véase* la URL en el Prefacio) se examinan más en detalle estos sistemas. Sin embargo, estos dos modelos presentan algunas desventajas fundamentales:

- es necesario escribir programas complejos para responder hasta la más simple de las consultas, utilizando un acceso de navegación orientado a registros;
- la independencia de los datos es mínima;
- no existe una base teórica ampliamente aceptada.

En 1970, E. F. Codd, del laboratorio IBM Research Laboratory elaboró su muy influyente artículo sobre el modelo de datos relacional. Este artículo fue muy oportuno y contemplaba las desventajas de las técnicas anteriores. Después de él, se implementaron muchos SGBD relacionales experimentales, llegando a aparecer los primeros productos comerciales a finales de la década de 1970 y principios de la de 1980. Mención especial merece el proyecto System R del laboratorio San Jose Research Laboratory de IBM en California, que se desarrolló a finales de la década de 1970 (Astrahan *et al*., 1976). Este proyecto se llevó a cabo para demostrar el carácter táctico del modelo relacional, proporcionando una implementación de sus estructuras de datos y operaciones, y condujo a su vez a dos desarrollos principales:

- el desarrollo de un lenguaje estructurado de consulta denominado SQL, que desde entonces se ha convertido en el lenguaje estándar para los SGBD relacionales;
- la implementación de diversos productos SGBD relacionales de carácter comercial durante la década de 1980, por ejemplo DB2 y SQL/DS de IBM y Oracle de Oracle Corporation.

Hoy en día existen centenares de sistemas SGBD relacionales tanto para entornos mainframe como para entornos PC, aunque muchos de ellos no se ajustan estrictamente a la definición del modelo relacional. Otros ejemplos de SGBD relacional multiusuario son Advantage Ingres Enterprise Relational Database, de Computer Associates, e Informix de IBM. Como ejemplos de SGBD relacional basado en PC están Office Access y Visual FoxPro de Microsoft, InterBase y JDataStore de Borland y R:Base de R:Base Technologies. Los SGBD relacionales se denominan sistemas de gestión de bases de datos de **segunda generación**. Hablaremos del modelo de datos relacional en el Capítulo 3.

El modelo relacional no carece de desventajas, en particular sus limitadas capacidades de modelado. Desde entonces, se ha investigado en profundidad, tratando de resolver este problema. En 1976, Chen presentó el modelo entidad-relación, que hoy en día constituye una técnica ampliamente aceptada para el diseño de bases de datos y constituye el fundamento para la metodología presentada en los Capítulos 15 y 16 de este libro. En 1979, el propio Codd trató de resolver algunas de las carencias de su propuesta original con una versión ampliada del modelo relacional denominada RM/T (1979) y posteriormente RM/V2 (1990). Los intentos de proporcionar un modelo de datos que represente el 'mundo real' de forma más precisa se han clasificado, con carácter general, bajo el término de **modelado de datos semántico**.

En respuesta a la creciente complejidad de las aplicaciones de bases de datos, han surgido dos 'nuevos' sistemas: los **SGBD orientados a objetos** (OODBMS) y los **SGBD objeto-relacionales** (ORDBMS). Sin embargo, a diferencia de los modelos anteriores, la composición real de estos modelos no está clara. Esta evolución representa lo que se denomina sistemas de gestión de bases de datos de **tercera generación**, de los que hablaremos en los Capítulos 25–28.

1.6 Ventajas y desventajas de los SGBD

Los sistemas de gestión de bases de datos presentan enormes ventajas potenciales. Desafortunadamente, también tienen algunas desventajas y en esta sección vamos a examinar tanto unas como otras.

Ventajas

Las ventajas de los sistemas de gestión de bases de datos se enumeran en la Tabla 1.2.

Tabla 1.2. Ventajas de un SGBD.

Control de la redundancia de los datos	Economía de escala
Coherencia de los datos	Equilibrio entre requisitos conflictivos
Más información a partir de la misma cantidad de datos	Mejor accesibilidad a los datos y mayor capacidad de respuesta
Compartición de datos	Productividad mejorada
Mayor integridad de los datos	Mantenimiento más sencillo gracias a la independencia de los datos
Mayor seguridad	Mayor nivel de concurrencia
Imposición de estándares	Servicios mejorados de copia de seguridad y recuperación

Control de la redundancia de los datos

Como hemos explicado en la Sección 1.2, los sistemas tradicionales basados en archivos desperdician espacio al almacenar la misma información en más de un archivo. Por ejemplo, en la Figura 1.5 vemos que se almacenaban datos similares relativos a los inmuebles en alquiler y a los clientes tanto en el departamento de ventas como en el de contratos. Por contraste, la técnica de base de datos trata de eliminar la redundancia integrando los archivos, de modo que no se almacenen múltiples copias de los mismos datos. Sin embargo, la técnica de base de datos no elimina la redundancia por completo, sino que controla la cantidad de redundancia inherente a la base de datos. Algunas veces, resulta necesario duplicar elementos de datos clave para modelar las relaciones, mientras que en otras ocasiones resulta deseable duplicar algunos elementos de datos para mejorar las prestaciones. Las razones que impulsan a este tipo de duplicación controlada quedarán más claras después de leer los siguientes capítulos.

Coherencia de los datos

Al eliminar o controlar la redundancia, reducimos el riesgo de que se produzcan incoherencias. Si un elemento de datos sólo se almacena una vez en la base de datos, las actualizaciones de su valor sólo tienen que llevarse a cabo una vez y el nuevo valor estará disponible de forma inmediata para todos los usuarios. Si almacenamos un elemento de datos más de una vez y el sistema es consciente de esto, el sistema podrá garantizar que todas las copias del elemento sean coherentes. Desafortunadamente, muchos de los SGBD actuales no garantizan de manera automática este tipo de coherencia.

Más información a partir de la misma cantidad de datos

Al integrar los datos operacionales, la información puede deducir información adicional a partir del conjunto de datos existente. Por ejemplo, en el sistema basado en archivos de la Figura 1.5, el departamento de contra-

tos no sabe quién es el propietario de un inmueble alquilado. De forma similar, el departamento de ventas no conoce los detalles de los contratos de alquiler. Al integrar los archivos, el departamento de contratos tiene acceso a los detalles referidos a los propietarios y el departamento de ventas a los detalles referentes a los contratos. A partir de la misma cantidad de datos, podemos ahora obtener mucha más información.

Compartición de los datos

Normalmente, los archivos son propiedad de las personas o departamentos que los usan. Por otro lado, la base de datos pertenece a toda la organización y debe ser compartida por todos los usuarios autorizados. De este modo, un número mayor de usuarios puede compartir una mayor cantidad de datos. Además, pueden escribirse nuevas aplicaciones que utilicen los datos existentes en la base de datos y añadir sólo los datos que no estén actualmente almacenados, en lugar de tener que volver a definir todos los requisitos referidos a los datos. Las nuevas aplicaciones también pueden emplear las funciones proporcionadas por el SGBD, como los mecanismos de definición y manipulación y los de control de concurrencia y recuperación, en lugar de tener que proporcionar esas funciones por sí mismas.

Mayor integridad en los datos

El concepto de integridad de la base de datos hace referencia a la validez y coherencia de los datos almacenados. La integridad se suele expresar en términos de **restricciones**, que son reglas de coherencia que no se permite que la base de datos viole. Las restricciones pueden aplicarse a los elementos de datos contenidos en un único registro o aplicarse en las relaciones entre registros. Por ejemplo, una restricción de integridad podría establecer que el salario de un empleado no sea superior a 40.000 euros o que el número de sucursal contenido en el registro de un empleado, que representa la sucursal en la que el empleado trabaja, debe corresponderse con una sucursal existente. De nuevo, la integración permite que el DBA defina y que el SGBD imponga las restricciones de integridad.

Mayor seguridad

La seguridad de la base de datos es la protección de los datos frente a su uso por personas no autorizadas. Sin unas medidas de seguridad adecuadas, la integración hace que los datos sean más vulnerables que en los sistemas basados en archivos. Sin embargo, la integración permite que el DBA defina y que el SGBD imponga medidas de seguridad en la base de datos. Estas medidas pueden tomar la forma de nombres de usuario y contraseñas para identificar a las personas autorizadas a usar la base de datos. También puede restringirse el tipo de acceso a los datos para los usuarios autorizados, según los tipos de operación (extracción, inserción, actualización, borrado). Por ejemplo, el DBA tiene acceso a todos los datos de la base de datos; un gerente de sucursal puede tener acceso a todos los datos relacionados con su sucursal; y un vendedor puede tener acceso a todos los datos relativos a los inmuebles, pero no a datos confidenciales como puedan ser los detalles salariales referidos al personal.

Imposición de estándares

De nuevo, la integración permite al DBA definir e imponer los estándares necesarios. Puede tratarse de estándares departamentales, de la organización, nacionales o internacionales referidos a cosas tales como los formatos de datos necesarios para facilitar el intercambio de datos entre sistemas, los convenios de denominación, los estándares de documentación, los procedimientos de actualización y las reglas de acceso.

Economía de escala

Al combinar todos los datos operacionales de una organización en una única base de datos y crear un conjunto de aplicaciones que funcionan con esta fuente centralizada de datos, pueden reducirse enormemente los costes. En este caso, el presupuesto que normalmente se asignaría a cada departamento para el desarrollo y mantenimiento de su sistema basado en archivos puede condenarse, lo que posiblemente resulte en un coste total inferior, lo que conduce a la obtención de economías de escala. El presupuesto combinado puede em-

plearse para comprar una configuración de sistema más adecuada a las necesidades de la organización. Dicha configuración puede consistir de una única computadora de gran potencia y tamaño o de una red de computadoras más pequeñas.

Equilibrio entre los requisitos conflictivos

Cada usuario de departamento tiene necesidades que pueden entrar en conflicto con las de otros usuarios. Puesto que la base de datos está bajo control del DBA, éste puede tomar decisiones acerca del diseño y la utilización operacional de la base de datos que proporcionen el mejor uso de los recursos, considerando a la organización como un todo. Estas decisiones proporcionarán un rendimiento óptimo para las aplicaciones importantes, posiblemente a expensas de las que sean menos críticas.

Mejor accesibilidad de los datos y mayor capacidad de respuesta

De nuevo, como resultado de la integración, los datos que atraviesan las fronteras departamentales son accesibles de modo directo por los usuarios finales. Esto proporciona un sistema con una funcionalidad potencialmente mucho mayor que puede, por ejemplo, emplearse para proporcionar mejores servicios al usuario final o a los clientes de la organización. Muchos SGBD proporcionan lenguajes de consulta o herramientas de escritura de informes que permiten a los usuarios plantear cuestiones *ad hoc* y obtener la información requerida casi inmediatamente en su terminal, sin necesidad de que ningún programador escriba un programa software para extraer la información de la base de datos. Por ejemplo, el gerente de una sucursal podría obtener un listado de todos los apartamentos cuyo alquiler mensual sea superior a 400 euros introduciendo la siguiente instrucción SQL en un terminal:

```
SELECT*
FROM PropertyForRent
WHERE type = 'Flat' AND rent > 400;
```

Mayor productividad

Como hemos mencionado anteriormente, el SGBD proporciona muchas de las funciones estándar que el programador tendría normalmente que incluir dentro de su aplicación basada en archivos. A un nivel básico, el SGBD proporciona todas las rutinas de bajo nivel para gestión de archivos que se utilizan normalmente en los programas de aplicación. La disponibilidad de estas funciones permite al programador concentrarse en la funcionalidad específica requerida por los usuarios, sin tener que preocuparse por los detalles de implementación de bajo nivel. Muchos SGBD también proporcionan un entorno de cuarta generación compuesto por herramientas que simplifican el desarrollo de aplicaciones de base de datos. Como resultado, se mejora la productividad de los programadores y se reducen los tiempos de desarrollo (lo que implica, a su vez, unos menores costes).

Mantenimiento simplificado gracias a la independencia de los datos

En los sistemas basados en archivos, las descripciones de los datos y la lógica para acceder a los datos están integradas en cada programa de aplicación, haciendo que los programas sean dependientes de los datos. Un cambio en la estructura de los datos, por ejemplo, que hiciera que una dirección tuviera 41 caracteres en lugar de 40, o un cambio en la forma de almacenar los datos en disco, podría requerir efectuar alteraciones sustanciales en todos los programas que se vieran afectados por el cambio. Por contraste, un SGBD separa las descripciones de los datos de las aplicaciones, haciendo así que las aplicaciones sean inmunes a los cambios que se efectúen en las descripciones de los datos. Este concepto se conoce con el nombre de **independencia de los datos** y hablaremos más de él en la Sección 2.1.5. La independencia de los datos simplifica el mantenimiento de las aplicaciones de la base de datos.

Mayor nivel de concurrencia

En algunos sistemas basados en archivos, si se permite a dos o más usuarios acceder al mismo archivo simultáneamente, es posible que los accesos se interfieran entre sí, provocando una pérdida de información o inclu-

so de la integridad de los datos. Muchos SGBD se encargan de gestionar el acceso concurrente a la base de datos y garantizan que dichos problemas no puedan presentarse. Hablaremos del control de concurrencia en el Capítulo 20.

Servicios mejorados de copia de seguridad y recuperación

Muchos sistemas basados en archivo asignan al usuario la responsabilidad de proporcionar medidas para proteger los datos frente a fallos del sistema informático o de los programas de aplicación. Esto puede requerir realizar copias de seguridad de los datos a diario, durante las horas nocturnas. En caso de que se produzca un fallo al día siguiente, se restaura la copia de seguridad y el trabajo que se haya realizado desde que fue hecha la copia de seguridad se pierde y es preciso volverlo a efectuar. Por contraste, los SGBD modernos proporcionan facilidades para minimizar la cantidad de procesamiento que se pierde debido a un fallo. Hablaremos de las cuestiones de recuperación de una base de datos en la Sección 20.3

Desventajas

Las desventajas de la técnica de base de datos se resumen en la Tabla 1.3.

Tabla 1.3. Desventajas de un SGBD.

Complejidad
Tamaño
Coste del SGBD
Costes del hardware adicional
Costes de conversión
Prestaciones
Mayor impacto de los fallos

Complejidad

Para que un buen SGBD pueda proporcionar la funcionalidad esperada, el SGBD tiene que ser un programa software de gran complejidad. Los desarrolladores y diseñadores de bases de datos, los administradores de datos y de bases de datos y los usuarios finales deben ser capaces de comprender esta funcionalidad para poder aprovecharla al máximo. Si no se comprende adecuadamente el sistema, pueden tomarse decisiones de diseño incorrectas, lo que podría tener serias consecuencias para la organización.

Tamaño

La complejidad y el amplio rango de funcionalidades hacen que el SGBD sea un programa software de gran tamaño, que ocupa muchos megabytes de espacio de disco y requiere una cantidad de memoria importante para poder ejecutarse de manera eficiente.

Coste del SGBD

El coste de los SGBD varía significativamente, dependiendo del entorno y de la funcionalidad proporcionada. Por ejemplo, un SGBD monousuario para una computadora personal puede costar sólo unos 100 euros. Sin embargo, un SGBD multiusuario de gran tamaño para mainframe, que preste servicio a centenares de usuarios, puede ser enormemente caro, quizá del orden de 100.000 o incluso 1.000.000 de euros. También hay que tener en cuenta el coste recurrente de mantenimiento anual, que normalmente suele ser un porcentaje del precio de venta.

Coste del hardware adicional

Los requisitos de almacenamiento en disco para el SGBD y la base de datos pueden imponer la compra de espacio de almacenamiento adicional. Además, para obtener las prestaciones requeridas, puede que sea nece-

sario comprar una plataforma hardware mayor, quizás incluso una máquina dedicada a ejecutar en exclusiva el SGBD. Estas compras de hardware adicional hacen que se incrementen los costes.

Costes de conversión

En algunas situaciones, el coste del SGBD y del hardware adicional puede ser insignificante si lo comparamos con el coste de convertir las aplicaciones existentes para que se ejecuten sobre el nuevo SGBD y sobre la nueva plataforma hardware. Este coste también incluye el coste de formar al personal en la utilización de los nuevos sistemas y, posiblemente, la contratación de personal especializado como ayuda durante la conversión y la operación del sistema. Este coste es una de las razones principales por las que algunas organizaciones se sienten atadas a sus sistemas actuales y no se deciden a cambiar a tecnologías de base de datos más modernas. Algunas veces se utiliza el término **sistema heredado** para referirse a un sistema más antiguo y usualmente inferior.

Prestaciones

Normalmente, los sistemas basados en archivos se escriben para una aplicación específica, como por ejemplo una aplicación de facturación. Como resultado, las prestaciones suelen ser muy buenas. Por el contrario, un SGBD se escribe para constituir una solución más general, con el fin de que puedan utilizarlo muchas aplicaciones y no sólo una aplicación concreta. El efecto es que algunas aplicaciones pueden ejecutarse más lentamente de lo que solían.

Mayor impacto de los fallos

La centralización de los recursos implementa la vulnerabilidad del sistema. Puesto que todos los usuarios y aplicaciones dependen de la disponibilidad del SGBD, el fallo de ciertos componentes puede hacer que se detengan todas las operaciones.

Resumen del capítulo

- El **sistema de gestión de bases de datos** (SGBD) constituye hoy en día la base fundamental de los sistemas de información y ha cambiado de modo radical la forma en que muchas organizaciones operan. Los sistemas de base de datos continúan siendo un área muy activa de investigación y son todavía muchos los problemas significativos que hay que resolver de manera satisfactoria.
- Los predecesores de los SGBD eran los **sistemas basados en archivos**, que son una colección de programas de aplicación que realizan una serie de servicios para los usuarios finales, usualmente la generación de informes. Cada programa define y gestiona sus propios datos. Aunque los sistemas basados en archivos constituyeron una gran mejora con respecto a los sistemas de archivo manual, siguen presentado problemas significativos, principalmente la redundancia de datos existente y la dependencia entre los programas y los datos.
- La técnica de bases de datos hizo su aparición para resolver los problemas asociados con los sistemas basados en archivos. Una **base de datos** es una colección compartida de datos lógicamente relacionados, junto con una descripción de estos datos, diseñada para satisfacer las necesidades de información de una organización. Un **SGBD** es un sistema software que permite a los usuarios definir, emplear, mantener y controlar el acceso a la base de datos. Un **programa de aplicación** es un programa informático que interactúa con la base de datos emitiendo solicitudes (normalmente instrucciones SQL) dirigidas al SGBD. Normalmente se utiliza el término **sistema de bases de datos** para definir una colección de programas de aplicación que interactúan con la base de datos, junto con el SGBD y la propia base de datos.
- Todos los accesos a la base de datos se realizan a través del SGBD. El SGBD proporciona un **lenguaje de definición de datos** (DDL, Data Definition Language), que permite a los usuarios definir la base de datos, y un **lenguaje de manipulación de datos** (DML, Data Manipulation Language), que permite a los usuarios insertar, actualizar, borrar y extraer datos de la base de datos.

- El SGBD proporciona un acceso controlado a la base de datos. Proporciona mecanismos de seguridad, integridad, control de concurrencia y control de recuperación, así como un catálogo accesible por el usuario. También proporciona un mecanismo de vistas para simplificar los datos con los que el usuario tiene que tratar.
- El entorno SGBD está compuesto de hardware (la computadora), de software (el SGBD, el sistema operativo y los sistemas de aplicación), de datos, de procedimientos y de personas. Las personas que se relacionan con el SGBD son los administradores de datos y de base de datos, los diseñadores de bases de datos, los desarrolladores de aplicaciones y los usuarios finales.
- Las raíces de los SGBD están en los sistemas basados en archivos. Los sistemas jerárquicos y CODASYL representan la primera generación de sistemas de gestión de bases de datos. El **modelo jerárquico** está representado por IMS (Information Management System), mientras que el **modelo en red** o **CODASYL** está ejemplificado por IDS (Integrate Data Store), ambos desarrollados a mediados de la década de 1960. El **modelo relacional**, propuesto por E. F. Codd en 1970, representa la segunda generación de sistemas de gestión de bases de datos. Este modelo tuvo una gran influencia sobre la comunidad de los SGBD y hoy en día existen más de 100 SGBD relacionales. La tercera generación de sistemas de gestión de bases de datos está representada por los SGBD **objeto-relacional** y los SGBD **orientados a objetos**.
- Entre las ventajas de la técnica de bases de datos podemos citar el control de la redundancia de los datos, la coherencia de los datos, la compartición de datos y unos mejores mecanismos de seguridad e integridad. Entre sus desventajas cabe resaltar la complejidad, el coste, la reducción de las prestaciones y el mayor impacto de los fallos sobre el sistema.

Cuestiones de repaso

1.1. Proporcione cuatro ejemplos de sistemas de bases de datos distintos de los enumerados en la Sección 1.1.

1.2. Explique cada uno de los siguientes términos:
   (a) datos
   (b) base de datos
   (c) sistema de gestión de bases de datos
   (d) programa de aplicación de bases de datos
   (e) independencia de los datos
   (f) seguridad
   (g) integridad
   (h) vistas.

1.3. Describa el enfoque de tratamiento de los datos adoptado en los antiguos sistemas basados en archivos. Indique las desventajas de este enfoque.

1.4. Describa las principales características del enfoque de base de datos y compárelas con la técnica basada en archivos.

1.5. Describa los cinco componentes del entorno SGBD y explique cómo se relacionan entre sí.

1.6. Explique el papel de cada una de las siguientes personas en un entorno de base de datos:
   (a) administrador de los datos
   (b) administrador de la base de datos
   (c) diseñador lógico de la base de datos
   (d) diseñador físico de la base de datos
   (e) desarrollador de aplicaciones
   (f) usuarios finales

1.7. Explique las ventajas y desventajas de los SGBD.

Ejercicios

1.8. Entreviste a algunos usuarios de sistemas de bases de datos. ¿Qué características de los SGBD encuentran más útiles y por qué? ¿Qué funciones de los SGBD encuentran menos útiles y por qué? ¿Cuáles creen estos usuarios que son las ventajas y desventajas de los SGBD?

1.9. Escriba un pequeño programa (utilizando pseudocódigo si fuera necesario) que permita la introducción y visualización de detalles de los clientes, incluyendo un número de cliente, el nombre, la dirección, el número telefónico, el número deseado de habitaciones y el alquiler máximo. Los detalles deben almacenarse en un archivo. Introduzca unos cuantos registros y muestre los detalles. Ahora repita este proceso pero, en lugar de escribir un programa especial, utilice cualquier SGBD al que tenga acceso. ¿Qué conclusiones puede sacar de estas dos técnicas?

1.10. Analice el caso de estudio de *DreamHome* presentado en la Sección 10.4 y en el Apéndice A. ¿De qué forma podría un SGBD ayudar a esta organización? Identifique los datos que es necesario representar en esta base de datos. ¿Qué relaciones existen entre los elementos de datos? ¿Qué consultas cree que se necesitarán?

1.11. Analice el caso de estudio de *Wellmeadows Hospital* presentado en el Apéndice B.3. ¿De qué forma podría un SGBD ayudar a esta organización? Identifique los datos que se necesiten representar en la base de datos. ¿Qué relaciones existen entre los elementos de datos?