SISTEMA DE GESTION DE BASES DE DATOS (SGBD)

Un **sistema gestor de bases de datos** (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada **base de datos**, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto *práctica* como *eficiente*.

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben garantizar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o de los intentos de acceso no autorizados. Si los datos van a ser compartidos entre diferentes usuarios, el sistema debe evitar posibles resultados anómalos.

Dado que la información es tan importante en la mayoría de las organizaciones, los científicos informáticos han desarrollado una gran cuerpo de conceptos y técnicas para la gestión de los datos. Estos conceptos y técnicas constituyen el objetivo central de este libro. En este capítulo se presenta una breve introducción a los principios de los sistemas de bases de datos.

gestores de bases de datos (SGBDs), las organizaciones
sistemas de este tipo.

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

- **Redundancia e inconsistencia de los datos**. Debido a que los archivos y programas de aplicación los crean diferentes programadores en el transcurso de un largo período de tiempo, es probable que los diversos archivos tengan estructuras diferentes y que los programas estén escritos en varios lenguajes de programación diferentes. Además, puede que la información esté duplicada en varios lugares (archivos). Por ejemplo, la dirección y el número de teléfono de un cliente dado pueden aparecer en un archivo que contenga registros de cuentas de ahorros y en un archivo que contenga registros de cuentas corrientes. Esta redundancia conduce a costes de almacenamiento y de acceso más elevados. Además, puede dar lugar a la **inconsistencia de los datos**; es decir, puede que las diferentes copias de los mismos datos no coincidan. Por ejemplo, puede que el cambio en la dirección de un cliente esté reflejado en los registros de las cuentas de ahorro pero no en el resto del sistema.

- **Dificultad en el acceso a los datos**. Supóngase que uno de los empleados del banco necesita averiguar los nombres de todos los clientes que viven en un código postal dado. El empleado pide al departamento de procesamiento de datos que genere esa lista. Debido a que esta petición no fue prevista por los diseñadores del sistema original, no hay un programa de aplicación a mano para satisfacerla. Hay, sin embargo, un programa de aplicación que genera la lista de todos los clientes. El empleado del banco tiene ahora dos opciones: bien obtener la lista de *todos* los clientes y extraer manualmente la información que necesita, o bien pedir a un programador de sistemas que escriba el programa de aplicación necesario. Ambas alternativas son obviamente insatisfactorias. Supóngase que se escribe el programa y que, varios días más tarde, el mismo empleado necesita reducir esa lista para que incluya únicamente a aquellos clientes que tengan una cuenta con saldo igual o superior a 10.000 €. Como se puede esperar, no existe ningún programa que genere tal lista. De nuevo, el empleado tiene que elegir entre dos opciones, ninguna de las cuales es satisfactoria.

  La cuestión aquí es que los entornos de procesamiento de archivos convencionales no permiten recuperar los datos necesarios de una forma práctica y eficiente. Hacen falta sistemas de recuperación de datos más adecuados para el uso general.

- **Aislamiento de datos.** Como los datos están dispersos en varios archivos, y los archivos pueden estar en diferentes formatos, es difícil escribir nuevos programas de aplicación para recuperar los datos correspondientes.

- **Problemas de integridad.** Los valores de los datos almacenados en la base de datos deben satisfacer ciertos tipos de **restricciones de consistencia**. Por ejemplo, el saldo de ciertos tipos de cuentas bancarias no puede nunca ser inferior a una cantidad predeterminada (por ejemplo, 25 €). Los desarrolladores hacen cumplir esas restricciones en el sistema añadiendo el código correspondiente en los diversos programas de aplicación. Sin embargo, cuando se añaden nuevas restricciones, es difícil cambiar los programas para hacer que se cumplan. El problema se complica cuando las restricciones implican diferentes elementos de datos de diferentes archivos.

1.4 Lenguajes de bases de datos

Los sistemas de bases de datos proporcionan un **lenguaje de definición de datos** para especificar el esquema de la base de datos y un **lenguaje de manipulación de datos** para expresar las consultas y las modificaciones de la base de datos. En la práctica, los lenguajes de definición y manipulación de datos no son dos lenguajes diferentes; en cambio, simplemente forman parte de un único lenguaje de bases de datos, como puede ser el muy usado SQL.

1.4.1 Lenguaje de manipulación de datos

Un **lenguaje de manipulación de datos** (LMD) es un lenguaje que permite a los usuarios tener acceso a los datos organizados mediante el modelo de datos correspondiente o manipularlos. Los tipos de acceso son:

- La recuperación de la información almacenada en la base de datos.

Los LMDs declarativos suelen resultar más fáciles de aprender y de usar que los procedimentales. Sin embargo, como el usuario no tiene que especificar cómo conseguir los datos, el sistema de bases de datos tiene que determinar un medio eficiente de acceso a los datos.

Una **consulta** es una instrucción que solicita que se recupere información. La parte de los LMDs implicada en la recuperación de información se denomina **lenguaje de consultas**. Aunque técnicamente sea incorrecto, resulta habitual usar las expresiones *lenguaje de consultas* y *lenguaje de manipulación de datos* como sinónimas.

Existen varios lenguajes de consultas de bases de datos en uso, tanto comercial como experimentalmente. El lenguaje de consultas más ampliamente usado, SQL, se estudiará en los Capítulos 3 y 4. También se estudiarán otros lenguajes de consultas en el Capítulo 5.

Los niveles de abstracción que se trataron en el Apartado 1.3 no sólo se aplican a la definición o estructuración de datos, sino también a su manipulación. En el nivel físico se deben definir los algoritmos que permitan un acceso eficiente a los datos. En los niveles superiores de abstracción se pone el énfasis en la facilidad de uso. El objetivo es permitir que los seres humanos interactúen de manera eficiente con el sistema. El componente procesador de consultas del sistema de bases de datos (que se estudia en los Capítulos 13 y 14) traduce las consultas LMD en secuencias de acciones en el nivel físico del sistema de bases de datos.

...ncias de acciones en el nivel físico del sistema de

1.4.2 Lenguaje de definición de datos

Los esquemas de las bases de datos se especifican mediante un conjunto de definiciones expresadas mediante un lenguaje especial denominado **lenguaje de definición de datos (LDD)**. El LDD también se usa para especificar más propiedades de los datos.

La estructura de almacenamiento y los métodos de acceso usados por el sistema de bases de datos se especifican mediante un conjunto de instrucciones en un tipo especial de LDD denominado **lenguaje de almacenamiento y definición de datos**. Estas instrucciones definen los detalles de implementación de los esquemas de las bases de datos, que suelen ocultarse a los usuarios.

Los valores de los datos almacenados en la base de datos deben satisfacer ciertas **restricciones de consistencia**. Por ejemplo, supóngase que el saldo de una cuenta no debe caer por debajo de 100 €. El LDD proporciona facilidades para especificar tales restricciones. Los sistemas de bases de datos las comprueban cada vez que se modifica la base de datos. En general, las restricciones pueden ser predicados arbitrarios relativos a la base de datos. No obstante, los predicados arbitrarios pueden resultar costosos de comprobar. Por tanto, los sistemas de bases de datos se concentran en las restricciones de integridad que pueden comprobarse con una sobrecarga mínima:

- **Restricciones de dominio.** Se debe asociar un dominio de valores posibles a cada atributo (por ejemplo, tipos enteros, tipos de carácter, tipos fecha/hora). La declaración de un atributo como parte de un dominio concreto actúa como restricción de los valores que puede adoptar. Las restricciones de dominio son la forma más elemental de restricción de integridad. El sistema las comprueba fácilmente siempre que se introduce un nuevo elemento de datos en la base de datos.
- **Integridad referencial.** Hay casos en los que se desea asegurar que un valor que aparece en una relación para un conjunto de atributos dado aparece también para un determinado conjunto de atributos en otra relación (integridad referencial). Las modificaciones de la base de datos pueden causar violaciones de la integridad referencial. Cuando se viola una restricción de integridad, el procedimiento normal es rechazar la acción que ha causado esa violación.
- **Asertos.** Un aserto es cualquier condición que la base de datos debe satisfacer siempre. Las restricciones de dominio y las restricciones de integridad referencial son formas especiales de asertos. No obstante, hay muchas restricciones que no pueden expresarse empleando únicamente esas formas especiales. Por ejemplo: "Cada préstamo tiene como mínimo un cliente tenedor de una cuenta con un saldo mínimo de 1.000,00 €" debe expresarse en forma de aserto. Cuando se crea un aserto, el sistema comprueba su validez. Si el aserto es válido, cualquier modificación futura de la base de datos se permite únicamente si no hace que se viole ese aserto.
- **Autorización.** Puede que se desee diferenciar entre los usuarios en cuanto al tipo de acceso que se les permite a diferentes valores de los datos de la base de datos. Estas diferenciaciones se

expresan en términos de **autorización**, cuyas modalidades más frecuentes son: **autorización de lectura**, que permite la lectura pero no la modificación de los datos; **autorización de inserción**, que permite la inserción de datos nuevos, pero no la modificación de los datos ya existentes; **autorización de actualización**, que permite la modificación, pero no la eliminación, de los datos; y la **autorización de eliminación**, que permite la eliminación de datos. A cada usuario se le pueden asignar todos, ninguno o una combinación de estos tipos de autorización.

El LDD, al igual que cualquier otro lenguaje de programación, obtiene como entrada algunas instrucciones y genera una salida. La salida del LDD se coloca en el **diccionario de datos**, que contiene **metadatos**—es decir, datos sobre datos. El diccionario de datos se considera un tipo especial de tabla, a la que sólo puede tener acceso y actualizar el propio sistema de bases de datos (no los usuarios normales). El sistema de bases de datos consulta el diccionario de datos antes de leer o modificar los datos reales.

1.5 Bases de datos relacionales

Las bases de datos relacionales se basan en el modelo relacional y usan un conjunto de tablas para representar tanto los datos como las relaciones entre ellos. También incluyen un LMD y un LDD. La mayor parte de los sistemas de bases de datos relacionales comerciales emplean el lenguaje SQL, que se trata en este apartado y que se tratará con gran detalle en los Capítulos 3 y 4. En el Capítulo 5 se estudiarán otros lenguajes influyentes.

1.5.1 Tablas

Cada tabla tiene varias columnas, y cada columna tiene un nombre único. En la Figura 1.2 se presenta un ejemplo de base de datos relacional consistente en tres tablas: una muestra detalles de los clientes de un banco, la segunda muestra las cuentas y la tercera muestra las cuentas que pertenecen a cada cliente.

La primera tabla, la tabla *cliente*, muestra, por ejemplo, que el cliente identificado por *id_cliente* 19.283.746 se llama González y vive en la calle Arenal en La Granja. La segunda tabla, *cuenta*, muestra, por ejemplo, que la cuenta C-101 tiene un saldo de 500 € y la C-201 un saldo de 900 €.

La tercera tabla muestra las cuentas que pertenecen a cada cliente. Por ejemplo, la cuenta C-101 pertenece al cliente cuyo *id_cliente* es 19.283.746 (González), y los clientes 19.283.746 (González) y 01.928.374 (Gómez) comparten el número de cuenta C-201 (pueden compartir un negocio).

El modelo relacional es un ejemplo de modelo basado en registros. Los modelos basados en registros se denominan así porque la base de datos se estructura en registros de formato fijo de varios tipos. Cada tabla contiene registros de un tipo dado. Cada tipo de registro define un número fijo de campos, o atributos. Las columnas de la tabla se corresponden con los atributos del tipo de registro.

No es difícil ver cómo se pueden almacenar las tablas en archivos. Por ejemplo, se puede usar un carácter especial (como la coma) para delimitar los diferentes atributos de un registro, y otro carácter especial (como el carácter de nueva línea) para delimitar los registros. El modelo relacional oculta esos detalles de implementación de bajo nivel a los desarrolladores de bases de datos y a los usuarios.

El modelo de datos relacional es el modelo de datos más ampliamente usado, y una gran mayoría de los sistemas de bases de datos actuales se basan en el modelo relacional. Los Capítulos 2 al 7 tratan el modelo relacional con detalle.

Obsérvese también que en el modelo relacional es posible crear esquemas que tengan problemas tales como información duplicada innecesariamente. Por ejemplo, supóngase que se almacena *número_cuenta* como atributo de un registro *cliente*. Entonces, para representar el hecho de que tanto la cuentas C-101 como la cuenta C-201 pertenece al cliente González (con *id_cliente* 19.283.746) sería necesario almacenar dos filas en la tabla *cliente*. Los valores de *nombre_cliente*, *calle_cliente* y *ciudad_cliente* de González estarían innecesariamente duplicados en las dos filas. En el Capítulo 7 se estudiará el modo de distinguir los buenos diseños de esquema de los malos.

id_cliente	*nombre_cliente*	*calle_cliente*	*ciudad_cliente*
19.283.746	González	Arenal, 12	La Granja
67.789.901	López	Mayor, 3	Peguerinos
18.273.609	Abril	Preciados, 123	Valsaín
32.112.312	Santos	Mayor, 100	Peguerinos
33.666.999	Rupérez	Ramblas, 175	León
01.928.374	Gómez	Carretas, 72	Cerceda

(a) La tabla *cliente*

número_cuenta	*saldo*
C-101	500
C-215	700
C-102	400
C-305	350
C-201	900
C-217	750
C-222	700

(b) La tabla *cuenta*

id_cliente	*número_cuenta*
19.283.746	C-101
19.283.746	C-201
01.928.374	C-215
67.789.901	C-102
18.273.609	C-305
32.112.312	C-217
33.666.999	C-222
01.928.374	C-201

(c) La tabla *impositor*

Figura 1.2 Un ejemplo de base de datos relacional.

1.5.2 Lenguaje de manipulación de datos

El lenguaje de consultas de SQL no es procedimental. Usa como entrada varias tablas (posiblemente sólo una) y devuelve siempre una sola tabla. A continuación se ofrece un ejemplo de consulta SQL que halla el nombre de todos los clientes que residen en Peguerinos:

```
select cliente.nombre_cliente
from cliente
where cliente.ciudad_cliente = 'Peguerinos'
```

La consulta especifica que hay que recuperar (*select*) las filas de (*from*) la tabla *cliente* en las que (*where*) la *ciudad_cliente* es Peguerinos, y que sólo debe mostrarse el atributo *nombre_cliente* de esas filas. Más concretamente, el resultado de la ejecución de esta consulta es una tabla con una sola columna denominada *nombre_cliente* y un conjunto de filas, cada una de las cuales contiene el nombre de un cliente cuya *ciudad_cliente* es Peguerinos. Si la consulta se ejecuta sobre la tabla de la Figura 1.2, el resultado constará de dos filas, una con el nombre López y otra con el nombre Santos.

Las consultas pueden involucrar información de más de una tabla. Por ejemplo, la siguiente consulta busca todos los números de cuenta y sus saldos del cliente con *id_cliente* 19.283.746.

```
select cuenta.número_cuenta, cuenta.saldo
from impositor, cuenta
where impositor.id_cliente = '19.283.746' and
      impositor.número_cuenta = cuenta.número_cuenta
```

Si la consulta anterior se ejecutase sobre las tablas de la Figura 1.2, el sistema encontraría que las dos cuentas denominadas C-101 y C-201 pertenecen al cliente 19.283.746 y el resultado consistiría en una tabla con dos columnas (*número_cuenta, saldo*) y dos filas (C-101, 500) y (C-201,900).

1.5.3 Lenguaje de definición de datos

SQL ofrece un LDD elaborado que permite definir tablas, restricciones de integridad, asertos, etc.

Por ejemplo, la siguiente instrucción del lenguaje SQL define la tabla *cuenta*:

```
create table cuenta
    (número_cuenta char(10),
     saldo integer)
```

La ejecución de esta instrucción LDD crea la tabla *cuenta*. Además, actualiza el diccionario de datos, que contiene metadatos (1.4.2). Los esquemas de las tablas son ejemplos de metadatos.

1.5.4 Acceso a las bases de datos desde los programas de aplicación

SQL no es tan potente como la máquina universal de Turing; es decir, hay algunos cálculos que no pueden obtenerse mediante ninguna consulta SQL. Esos cálculos deben escribirse en un lenguaje *anfitrión*, como Cobol, C, C++ o Java. Los **programas de aplicación** son programas que se usan para interactuar de esta manera con las bases de datos. Algunos de los ejemplos de un sistema bancario serían los programas que generan las nóminas, realizan cargos en las cuentas, realizan abonos en las cuentas o transfieren fondos entre las cuentas.

Para tener acceso a la base de datos, las instrucciones LMD deben ejecutarse desde el lenguaje anfitrión. Hay dos maneras de conseguirlo:

- Proporcionando una interfaz de programas de aplicación (conjunto de procedimientos) que se pueda usar para enviar instrucciones LMD y LDD a la base de datos y recuperar los resultados.

  El estándar de conectividad abierta de bases de datos (ODBC, Open Data Base Connectivity) definido por Microsoft para su empleo con el lenguaje C es un estándar de interfaz de programas de aplicación usado habitualmente. El estándar de conectividad de Java con bases de datos (JDBC, Java Data Base Connectivity) ofrece las características correspondientes para el lenguaje Java.

- Extendiendo la sintaxis del lenguaje anfitrión para que incorpore las llamadas LMD dentro del programa del lenguaje anfitrión. Generalmente, un carácter especial precede a las llamadas LMD y un preprocesador, denominado **precompilador LMD**, convierte las instrucciones LMD en llamadas normales a procedimientos en el lenguaje anfitrión.

1.8 Almacenamiento de datos y consultas

Los sistemas de bases de datos están divididos en módulos que tratan con cada una de las responsabilidades del sistema general. Los componentes funcionales de los sistemas de bases de datos pueden dividirse grosso modo en los componentes gestor de almacenamiento y procesador de consultas.

El gestor de almacenamiento es importante porque las bases de datos suelen necesitar una gran cantidad de espacio de almacenamiento. Las bases de datos corporativas tienen un tamaño que va de los centenares de gigabytes hasta, para las bases de datos de mayor tamaño, los terabytes de datos. Un gigabyte son aproximadamente 1.000 megabytes (1.000 millones de bytes), y un terabyte es aproximadamente un millón de megabytes (1 billón de bytes). Debido a que la memoria principal de las computadoras no puede almacenar toda esta información, la información se almacena en discos. Los datos se intercambian entre los discos de almacenamiento y la memoria principal cuando sea necesario. Como el intercambio de datos con el disco es lento comparado con la velocidad de la unidad central de procesamiento, es fundamental que el sistema de base de datos estructure los datos para minimizar la necesidad de intercambio de datos entre los discos y la memoria principal.

El procesador de consultas es importante porque ayuda al sistema de bases de datos a simplificar y facilitar el acceso a los datos. Las vistas de alto nivel ayudan a conseguir este objetivo; con ellas, los usuarios del sistema no se ven molestados innecesariamente con los detalles físicos de la implementación del sistema. Sin embargo, el rápido procesamiento de las actualizaciones y de las consultas es importante. Es función del sistema de bases de datos traducir las actualizaciones y las consultas escritas en lenguajes no procedimentales, en el nivel lógico, en una secuencia eficiente de operaciones en el nivel físico.

1.8.1 Gestor de almacenamiento

Un *gestor de almacenamiento* es un módulo de programa que proporciona la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y las consultas remitidas al sistema. El gestor de almacenamiento es responsable de la interacción con el gestor de archivos. Los datos en bruto se almacenan en el disco mediante el sistema de archivos que suele proporcionar un sistema operativo convencional. El gestor de almacenamiento traduce las diferentes instrucciones LMD a comandos de bajo nivel del sistema de archivos. Así, el gestor de almacenamiento es responsable del almacenamiento, la recuperación y la actualización de los datos de la base de datos.

Entre los componentes del gestor de almacenamiento se encuentran:

- **Gestor de autorizaciones e integridad**, que comprueba que se satisfagan las restricciones de integridad y la autorización de los usuarios para tener acceso a los datos.

- **Gestor de transacciones**, que garantiza que la base de datos quede en un estado consistente (correcto) a pesar de los fallos del sistema, y que la ejecución concurrente de transacciones transcurra si conflictos.
- **Gestor de archivos**, que gestiona la asignación de espacio de almacenamiento de disco y las estructuras de datos usadas para representar la información almacenada en el disco.
- **Gestor de la memoria intermedia**, que es responsable de traer los datos desde el disco de almacenamiento a la memoria principal y decidir los datos a guardar en la memoria caché. El gestor de la memoria intermedia es una parte fundamental de los sistemas de bases de datos, ya que permite que la base de datos maneje tamaños de datos que son mucho mayores que el tamaño de la memoria principal.

El gestor de almacenamiento implementa varias estructuras de datos como parte de la implementación física del sistema:

- **Archivos de datos**, que almacenan la base de datos en sí misma.
- **Diccionario de datos**, que almacena metadatos acerca de la estructura de la base de datos; en particular, su esquema.
- **Índices**, que pueden proporcionar un acceso rápido a los elementos de datos. Como el índice de este libro de texto, los índices de las bases de datos facilitan punteros a los elementos de datos que tienen un valor concreto. Por ejemplo, se puede usar un índice para buscar todos los registros *cuenta* con un *número_cuenta* determinado. La asociación es una alternativa a la indexación que es más rápida en algunos casos, pero no en todos.

Se estudiarán los medios de almacenamiento, las estructuras de archivos y la gestión de la memoria intermedia en el Capítulo 11. Los métodos de acceso eficiente a los datos mediante indexación o asociación se explican en el Capítulo 12.

1.8.2 El procesador de consultas

Entre los componentes del procesador de consultas se encuentran:

- **Intérprete del LDD**, que interpreta las instrucciones del LDD y registra las definiciones en el diccionario de datos.
- **Compilador del LMD**, que traduce las instrucciones del LMD en un lenguaje de consultas a un plan de evaluación que consiste en instrucciones de bajo nivel que entienda el motor de evaluación de consultas.

  Las consultas se suelen poder traducir en varios planes de ejecución alternativos, todos los cuales proporcionan el mismo resultado. El compilador del LMD también realiza **optimización de consultas**, es decir, elige el plan de evaluación de menor coste de entre todas las opciones posibles.
- **Motor de evaluación de consultas**, que ejecuta las instrucciones de bajo nivel generadas por el compilador del LMD.

La evaluación de las consultas se trata en el Capítulo 13, mientras que los métodos por los que el optimizador de consultas elige entre las estrategias de evaluación posibles se tratan en el Capítulo 14.

1.9 Gestión de transacciones

A menudo, varias operaciones sobre la base de datos forman una única unidad lógica de trabajo. Un ejemplo son las transferencia de fondos, como se vio en el Apartado 1.2, en las que se realiza un cargo en una cuenta (llámese *A*) y un abono en otra cuenta (llámese *B*). Evidentemente, resulta fundamental que, o bien tengan lugar tanto el cargo como el abono, o bien que no se produzca ninguno. Es decir, la transferencia de fondos debe tener lugar por completo o no producirse en absoluto. Este requisito

1.12 Usuarios y administradores de bases de datos

Uno de los objetivos principales de los sistemas de bases de datos es recuperar información de la base de datos y almacenar en ella información nueva. Las personas que trabajan con una base de datos se pueden clasificar como usuarios o administradores de bases de datos.

1.12.1 Usuarios de bases de datos e interfaces de usuario

Hay cuatro tipos diferentes de usuarios de los sistemas de bases de datos, diferenciados por la forma en que esperan interactuar con el sistema. Se han diseñado diferentes tipos de interfaces de usuario para los diferentes tipos de usuarios.

- Los **usuarios normales** son usuarios no sofisticados que interactúan con el sistema invocando alguno de los programas de aplicación que se han escrito previamente. Por ejemplo, un cajero bancario que necesita transferir 50 € de la cuenta *A* a la cuenta *B* invoca un programa llamado *transferencia*. Ese programa le pide al cajero el importe de dinero que se va a transferir, la cuenta desde la que se va a transferir el dinero y la cuenta a la que se va a transferir el dinero.

  Como ejemplo adicional, considérese un usuario que desea averiguar el saldo de su cuenta en World Wide Web. Ese usuario puede acceder a un formulario en el que introduce su número de cuenta. Un programa de aplicación en el servidor Web recupera entonces el saldo de la cuenta, usando el número de cuenta proporcionado, y devuelve la información al usuario.

  La interfaz de usuario habitual para los usuarios normales es una interfaz de formularios, donde el usuario puede rellenar los campos correspondientes del formulario. Los usuarios normales también pueden limitarse a leer *informes* generados por la base de datos.

- Los **programadores de aplicaciones** son profesionales informáticos que escriben programas de aplicación. Los programadores de aplicaciones pueden elegir entre muchas herramientas para desarrollar las interfaces de usuario. Las herramientas de **desarrollo rápido de aplicaciones**

(DRA) son herramientas que permiten al programador de aplicaciones crear formularios e informes con un mínimo esfuerzo de programación.

- Los **usuarios sofisticados** interactúan con el sistema sin escribir programas. En su lugar, formulan sus consultas en un lenguaje de consultas de bases de datos. Remiten cada una de las consultas al **procesador de consultas**, cuya función es dividir las instrucciones LMD en instrucciones que el gestor de almacenamiento entienda. Los analistas que remiten las consultas para explorar los datos de la base de datos entran en esta categoría.
- Los **usuarios especializados** son usuarios sofisticados que escriben aplicaciones de bases de datos especializadas que no encajan en el marco tradicional del procesamiento de datos. Entre estas aplicaciones están los sistemas de diseño asistido por computadora, los sistemas de bases de conocimientos y los sistemas expertos, los sistemas que almacenan datos con tipos de datos complejos (por ejemplo, los datos gráficos y los datos de sonido) y los sistemas de modelado del entorno. En el Capítulo 9 se estudian varias de estas aplicaciones.

1.12.2 Administrador de bases de datos

Una de las principales razones de usar SGBDs es tener un control centralizado tanto de los datos como de los programas que tienen acceso a esos datos. La persona que tiene ese control central sobre el sistema se denomina **administrador de bases de datos (ABD)**. Las funciones del ABD incluyen:

- La **definición del esquema**. El ABD crea el esquema original de la base de datos mediante la ejecución de un conjunto de instrucciones de definición de datos en el LDD.
- La **definición de la estructura y del método de acceso**.
- La **modificación del esquema y de la organización física**. El ABD realiza modificaciones en el esquema y en la organización física para reflejar las necesidades cambiantes de la organización o para alterar la organización física a fin de mejorar el rendimiento.
- La **concesión de autorización para el acceso a los datos**. Mediante la concesión de diferentes tipos de autorización, el administrador de bases de datos puede regular las partes de la base de datos a las que puede tener acceso cada usuario. La información de autorización se guarda en una estructura especial del sistema que el SGBD consulta siempre que alguien intenta tener acceso a los datos del sistema.
- El **mantenimiento rutinario**. Algunos ejemplos de las actividades de mantenimiento rutinario del administrador de la base de datos son:
  - ☐ Copia de seguridad periódica de la base de datos, bien sobre cinta o sobre servidores remotos, para impedir la pérdida de datos en caso de desastres como las inundaciones.
  - ☐ Asegurarse de que se dispone de suficiente espacio libre en disco para las operaciones normales y aumentar el espacio en disco según sea necesario.
  - ☐ Supervisar los trabajos que se ejecuten en la base de datos y asegurarse de que el rendimiento no se degrade debido a que algún usuario haya remitido tareas muy costosas.