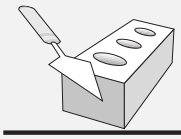

1

INTRODUCCIÓN A LOS SISTEMAS DE BASES DE DATOS

- ¿Qué son los SGBD y, en concreto, los SGBD relacionales?
- ¿Por qué se debe pensar en un SGBD a la hora de gestionar datos?
- ¿Cómo se representan los datos de las aplicaciones en los SGBD?
- ¿Cómo se recuperan y se tratan los datos en los SGBD?
- ¿Cómo soportan los SGBD el acceso concurrente y cómo protegen los datos durante los fallos del sistema?
- ¿Cuáles son los componentes principales de un SGBD?
- ¿Quién tiene relación con las bases de datos en la vida real?
- **Conceptos fundamentales:** administración de bases de datos, independencia con respecto a los datos, diseño de bases de datos, modelos de datos; bases de datos relacionales y consultas; esquemas, niveles de abstracción; transacciones, concurrencia y bloqueos, recuperación y registros; arquitectura de los SGBD; administradores de bases de datos, programadores de aplicaciones, usuarios finales.

> ¿Se han dado cuenta todos de que todas las letras de la palabra *database* (base de datos en inglés) se escriben a máquina con la mano izquierda? Ahora bien, la disposición del teclado de máquina de escribir QWERTY se diseñó, entre otras cosas, para facilitar un uso equilibrado de ambas manos. Se deduce, por tanto, que escribir sobre bases de datos no sólo es antinatural, sino también mucho más difícil de lo que parece.
>
> — Anónimo

La cantidad de información disponible crece, literalmente, de manera explosiva, y el valor de los datos como activo de las organizaciones está ampliamente reconocido. Para que los usuarios obtengan el máximo rendimiento de sus enormes y complejos conjuntos de datos son necesarias herramientas que simplifiquen las tareas de administrar los datos y de extraer

El campo de los sistemas gestores de bases de datos es un microcosmos dentro de la informática en general. Los aspectos abordados y las técnicas empleadas abarcan un amplio espectro, que incluye los lenguajes, la orientación a objetos y otros paradigmas de la programación, la compilación, los sistemas operativos, la programación concurrente, las estructuras de datos, los algoritmos, la teoría, los sistemas paralelos y distribuidos, las interfaces de usuario, los sistemas expertos y la inteligencia artificial, las técnicas estadísticas y la programación dinámica. No es posible tratar todos los aspectos de la gestión de bases de datos en un solo libro, pero los autores esperan transmitir al lector una idea de la excitación que provoca esta rica y vibrante disciplina.

información útil en el momento preciso. En caso contrario, los datos pueden convertirse en una carga cuyo coste de adquisición y de gestión supere ampliamente el valor obtenido de ellos.

Una **base de datos** es un conjunto de datos, que generalmente describe las actividades de una o varias organizaciones relacionadas. Por ejemplo, la base de datos de una universidad puede contener información sobre:

- *Entidades* como alumnos, profesores, asignaturas y aulas.
- *Relaciones* entre esas entidades, como la matriculación de los alumnos en las diversas asignaturas, los profesores que imparten cada asignatura y el empleo de las aulas para las diferentes asignaturas.

Un **sistema gestor de bases de datos**, o **SGBD**, es el software diseñado para colaborar en el mantenimiento y empleo de grandes conjuntos de datos. La necesidad de este tipo de sistemas, así como su uso, está aumentando rápidamente. La alternativa al empleo de un SGBD es almacenar los datos en archivos y escribir código específico para una aplicación que los gestione. El empleo de un SGBD presenta varias ventajas de importancia, como se verá en el Apartado 1.4.

1.1 GESTIÓN DE DATOS

El objetivo de este libro es presentar una introducción en profundidad a los sistemas gestores de bases de datos, con el énfasis puesto en la manera de *diseñar* bases de datos y *usar* los SGBD de manera efectiva. No cabe sorprenderse de que muchas de las decisiones sobre la manera de usar un SGBD concreto para una aplicación dada dependan de las posibilidades que ese SGBD soporte eficientemente. Por tanto, para usar bien un SGBD, es necesario comprender también la manera en que *funciona*.

Actualmente se utilizan muchos tipos de sistemas gestores de bases de datos, pero este libro se concentrará en los **sistemas gestores de bases de datos relacionales (SGBDR)**, que son, por mucho, el tipo de SGBD dominante hoy en día. Las preguntas siguientes se responden en los capítulos principales de este libro:

1. **Diseño de bases de datos y desarrollo de aplicaciones.** ¿Cómo puede un usuario describir una empresa real (por ejemplo, una universidad) en términos de los datos alma-

cenados en un SGBD? ¿Qué factores se deben tomar en consideración a la hora de decidir la manera de organizar los datos almacenados? ¿Cómo se pueden desarrollar aplicaciones basadas en SGBD? (Capítulos 2, 3, 6, 7, 12, 13 y 14).

2. **Análisis de datos.** ¿Cómo puede un usuario responder a preguntas sobre la empresa planteando consultas sobre los datos del SGBD? (Capítulos 4 y 5)[1].
3. **Concurrencia.** ¿Cómo permite un SGBD que muchos usuarios tengan acceso concurrente a los datos? (Capítulo 8).
4. **Eficiencia y escalabilidad.** ¿Cómo almacena un SGBD conjuntos de datos de gran tamaño y responde preguntas sobre esos datos de manera eficiente? (Capítulos 9, 10 y 11).

Los capítulos posteriores tratan temas importantes y que se hallan en rápida evolución, como los almacenes de datos (*data warehouses*) y las consultas complejas de ayuda a la toma de decisiones, la minería de datos (*data mining*), las bases de datos y la recuperación de la información, los repositorios XML y las bases de datos orientadas a objetos.

En el resto de este capítulo se introducirán varios temas. En el Apartado 1.2 se comenzará una breve historia de este campo científico y una discusión sobre el papel de la gestión de las bases de datos en los sistemas de información modernos. Posteriormente se identificarán las ventajas del almacenamiento de datos en SGBD en lugar de en sistemas de archivos en el Apartado 1.3, y se discutirán las ventajas del empleo de SGBD para gestionar datos en el Apartado 1.4. En el Apartado 1.5 se considerará la manera en que se debe organizar y almacenar en un SGBD la información relativa a las empresas. Probablemente los usuarios piensen en esta información en términos de alto nivel que se correspondan con las entidades de la organización y con sus relaciones, mientras que los SGBD acaban almacenando los datos en forma de (muchísimos) bits. La distancia entre la manera en que los usuarios piensan en sus datos y el modo en que se acaban almacenando se salva mediante varios *niveles de abstracción* soportados por los SGBD. De manera intuitiva, los usuarios pueden empezar describiendo los datos en términos de un nivel bastante elevado y refinar luego esa descripción en lo que haga falta cuando se consideren detalles tanto de su almacenamiento como de su representación.

En el Apartado 1.6 se considera la manera en que los usuarios pueden recuperar los datos almacenados en los SGBD y la necesidad de técnicas que calculen de manera eficiente las respuestas a las preguntas que afecten a esos datos. En el Apartado 1.7 se ofrece una visión general del modo en que los SGBD permiten que múltiples usuarios accedan concurrentemente a los datos.

Luego se describe brevemente la arquitectura de los SGBD en el Apartado 1.8 y se mencionan diversos grupos de personas asociadas con el desarrollo y empleo de los SGBD en el Apartado 1.9.

1.2 PERSPECTIVA HISTÓRICA

Desde los primeros días de la informática el almacenamiento y el tratamiento de los datos han sido el centro de atención de importantes aplicaciones. El primer SGBD de finalidad

[1] En Internet también se dispone de un capítulo en inglés sobre QBE (Query-by-Example).

generalista, diseñado por Charles Bachman en General Electric a principios de los años sesenta del siglo veinte se denominó Almacén integrado de datos (Integrated Data Store). Aportó la base para el *modelo de datos en red*, que se normalizó en la Conferencia sobre lenguajes de sistemas de datos (Conference on Data Systems Languages, CODASYL) y tuvo gran influencia en los sistemas de bases de datos a lo largo de los años sesenta del siglo veinte. Bachman fue el primer ganador del Premio Turing de la ACM (el equivalente informático al Premio Nobel) por su trabajo en el campo de las bases de datos; lo recibió en 1973.

A finales de los años sesenta del siglo veinte IBM desarrolló el SGBD Sistema de Gestión de la Información (Information Management System, IMS), empleado incluso hoy en día en muchas instalaciones importantes. IMS constituyó la base para un entramado alternativo para la representación de los datos denominado *modelo jerárquico de datos*. El sistema SABRE para la realización de reservas en las líneas aéreas fue desarrollado conjuntamente por American Airlines e IBM por aquella época, y permitía que varias personas tuvieran acceso a los mismos datos mediante una red informática. Resulta interesante que hoy en día se utilice el mismo sistema SABRE para el funcionamiento de servicios de viajes basados en Internet tan conocidos como Travelocity.

En 1970, Edgar Codd, del Laboratorio de Investigación de San José de IBM, propuso un nuevo entramado de representación de los datos denominado *modelo relacional de datos*. Este modelo ha demostrado ser un hito en el desarrollo de los sistemas de bases de datos: provocó el rápido desarrollo de varios SGBD basados en el modelo relacional, junto con una amplia variedad de resultados teóricos que han dado sólidos fundamentos a esta rama de la ciencia. Codd ganó el Premio Turing de 1981 por su trabajo pionero. Los sistemas de bases de datos maduraron como disciplina académica, y la popularidad de los SGBD cambió el paisaje comercial. Sus ventajas fueron ampliamente reconocidas y el empleo de SGBD para la gestión de los datos empresariales se volvió algo habitual.

En los años ochenta del siglo veinte el modelo relacional consolidó su posición como paradigma dominante de los SGBD y los sistemas de bases de datos siguieron ampliando su ámbito de aplicación. El lenguaje de consultas SQL para las bases de datos relacionales, desarrollado como parte del proyecto System R de IBM, es en la actualidad el lenguaje de consultas estándar. SQL fue normalizado a finales de los años ochenta del siglo veinte y la norma actual, SQL:1999, ha sido adoptada por el Instituto Nacional Americano de Normalización (American National Standards Institute, ANSI) y la Organización Internacional para la Normalización (International Organization for Standardization, ISO). Puede afirmarse que la forma más utilizada de programación concurrente es la ejecución concurrente de programas de bases de datos (denominados *transacciones*). Los usuarios escriben los programas como si se fueran a ejecutar de manera independiente, y la responsabilidad de ejecutarlos de manera concurrente recae en el SGBD. James Gray ganó el Premio Turing 1999 por sus aportaciones a la gestión de las transacciones de las bases de datos.

A finales de los años ochenta y en los años noventa del siglo veinte se realizaron avances en muchos aspectos de los sistemas de bases de datos. Se llevó a cabo gran cantidad de investigaciones en lenguajes de consulta más potentes y en modelos de datos más ricos, con el énfasis puesto en el soporte de complejos análisis de todas las partes de una empresa. Varios fabricantes (por ejemplo, DB2 de IBM, Oracle 8 y UDS de Informix[2]) ampliaron sus

[2]Informix fue adquirida recientemente por IBM.

sistemas con la posibilidad de almacenar nuevos tipos de datos como las imágenes y el texto, y la de formular consultas más complejas. Muchos fabricantes han desarrollado sistemas especializados para la creación de *almacenes de datos*, la consolidación de datos procedentes de varias bases de datos y la ejecución de análisis especializados.

Un fenómeno interesante es la aparición de varios paquetes de **planificación de recursos empresariales** (enterprise resource planning, ERP) y de **planificación de recursos de gestión** (management resource planning, MRP), que añaden una capa sustancial de características orientadas a las aplicaciones por encima de los SGBD. Entre los paquetes más usados hay sistemas de Baan, Oracle, PeopleSoft, SAP y Siebel. Estos paquetes identifican un conjunto de tareas frecuentes (por ejemplo, la gestión de inventarios, la planificación de los recursos humanos, el análisis financiero) que se llevan a cabo en gran número de organizaciones y ofrecen una capa general de aplicaciones para desempeñarlas. Los datos se almacenan en un SGBD relacional y la capa de aplicaciones se puede personalizar para diferentes empresas, lo que permite reducir los costes globales para las empresas en comparación con el coste de creación de la capa de aplicaciones desde el principio.

Lo que quizás resulte más significativo es que los SGBD han entrado en la Era de Internet. Mientras la primera generación de sitios Web guardaba sus datos exclusivamente en archivos del sistema operativo, el empleo de SGBD para almacenar los datos a que se tiene acceso mediante los navegadores Web se está generalizando. Las consultas se generan mediante formularios accesibles por Web y se da formato a las respuestas mediante un lenguaje de marcas como HTML para que se muestren con facilidad en los navegadores. Todos los fabricantes de bases de datos están añadiendo a sus SGBD características dirigidas a hacerlos más adecuados para su implantación en Internet.

La gestión de bases de datos sigue adquiriendo importancia a medida que se van poniendo en línea más datos y se hacen cada vez más accesibles mediante las redes de ordenadores. Hoy en día este campo está siendo impulsado por visiones excitantes como las bases de datos multimedia, el vídeo interactivo, las corrientes de datos, las bibliotecas digitales, buen número de proyectos científicos como el esfuerzo de elaboración del mapa del genoma humano y el proyecto del Sistema de observación de la Tierra de la NASA (Earth Observation System) y el deseo de las empresas de consolidar sus procesos de toma de decisiones y *explorar* sus repositorios de datos en búsqueda de información útil sobre sus negocios. Comercialmente, los sistemas gestores de bases de datos representan uno de los mayores y más vigorosos segmentos del mercado. Por tanto, el estudio de los sistemas de bases de datos puede resultar muy provechoso por muchos motivos.

1.3 SISTEMAS DE ARCHIVOS Y SGBD

Para comprender la necesidad de los SGBD, considérese la siguiente situación: una empresa tiene un gran conjunto de datos (digamos que unos 500 GB[3]) sobre empleados, departamentos, productos, ventas, etcétera. A estos datos tienen acceso de manera concurrente varios empleados. Las preguntas sobre los datos se deben responder rápidamente, las modificaciones

[3]Un kilobyte (KB) equivale a 1024 bytes, un megabyte (MB) a 1024 KB, un gigabyte (GB) a 1024 MB, un terabyte (TB) a 1024 GB y un petabyte (PB) a 1024 terabytes.

de los datos realizadas por los diferentes usuarios deben aplicarse de manera consistente y el acceso a ciertas partes de los datos (por ejemplo, los sueldos) debe estar restringido.

Se puede intentar gestionar los datos almacenándolos en archivos del sistema operativo. Este enfoque tiene muchos inconvenientes y, entre ellos, los siguientes:

- Probablemente no se disponga de 500 GB de memoria principal para albergar todos los datos. Por tanto, se deben guardar los datos en un dispositivo de almacenamiento como discos o cintas y llevar las partes de importancia a la memoria principal para su procesamiento cuando resulte necesario.
- Aunque se dispusiera de 500 GB de memoria principal, en los sistemas informáticos con direccionamiento de treinta y dos bits no se puede hacer referencia directamente más que a unos 4 GB de datos. Hay que programar algún método para identificar todos los datos.
- Hay que escribir programas especiales para responder a cada pregunta que puedan formular los usuarios sobre esos datos. Es muy probable que esos programas sean complejos debido al gran volumen de datos que hay que analizar.
- Hay que proteger los datos de las modificaciones inconsistentes realizadas por usuarios diferentes que acceden a los datos de manera concurrente. Si las aplicaciones deben afrontar los detalles de ese acceso concurrente se incrementa notablemente su complejidad.
- Hay que garantizar que los datos vuelvan a un estado consistente si el sistema falla mientras se está realizando alguna modificación.
- Los sistemas operativos sólo ofrecen para la seguridad un mecanismo de contraseñas. Esto no es lo bastante flexible para la aplicación de directivas de seguridad en las que los diferentes usuarios tengan permiso para el acceso a diferentes subconjuntos de los datos.

Los SGBD son aplicaciones software diseñadas para facilitar las tareas mencionadas. Al almacenar los datos en un SGBD en vez de en un conjunto de archivos del sistema operativo, se pueden utilizar las características del SGBD para gestionar los datos de un modo robusto y eficiente. A medida que crecen el volumen de los datos y el número de usuarios —en las bases de datos corporativas actuales son habituales los centenares de gigabytes y los millares de usuarios— el apoyo de los SGBD se vuelve indispensable.

1.4 VENTAJAS DE LOS SGBD

El empleo de SGBD para gestionar los datos tiene muchas ventajas:

- **Independencia con respecto a los datos.** Los programas de las aplicaciones no deben, en principio, exponerse a los detalles de la representación y el almacenamiento de los datos. El SGBD ofrece una vista abstracta de los datos que oculta esos detalles.
- **Acceso eficiente a los datos.** Los SGBD emplean gran variedad de técnicas sofisticadas para almacenar y recuperar los datos de manera eficiente. Esta característica resulta especialmente importante si los datos se guardan en dispositivos de almacenamiento externos.

- **Integridad y seguridad de los datos.** Si siempre se tiene acceso a los datos mediante el SGBD, éste puede hacer que se cumplan las restricciones de integridad. Por ejemplo, antes de introducir la información salarial de un empleado, el SGBD puede comprobar que no se haya superado el presupuesto del departamento. Además, puede hacer que se cumplan los *controles de acceso* que determinan los datos que son visibles para las diferentes clases de usuarios.
- **Administración de los datos.** Cuando varios usuarios comparten los mismos datos, la centralización de la administración de esos datos puede ofrecer mejoras significativas. Los profesionales experimentados que comprenden la naturaleza de los datos que se gestionan y la manera en que los utilizan los diferentes grupos de usuarios, pueden responsabilizarse de la organización de la representación de los datos para minimizar la redundancia y mejorar el almacenamiento de los datos para hacer que la recuperación sea eficiente.
- **Acceso concurrente y recuperación en caso de fallo.** Los SGBD programan los accesos concurrentes a los datos de tal manera que los usuarios puedan creer que sólo tiene acceso a los datos un usuario a la vez. Además, los SGBD protegen a los usuarios de los efectos de los fallos del sistema.
- **Reducción del tiempo de desarrollo de las aplicaciones.** Evidentemente, los SGBD soportan funciones importantes que son comunes con muchas aplicaciones que tienen acceso a los datos del SGBD. Esto, junto con la interfaz de alto nivel con los datos, facilita el rápido desarrollo de aplicaciones. También es probable que las aplicaciones de los SGBD sean más robustas que las aplicaciones independientes similares, ya que el SGBD maneja muchas tareas importantes (y no hay que depurarlas y probarlas en la aplicación).

Dadas todas estas ventajas, ¿puede haber alguna razón para *no* utilizar un SGBD? A veces, sí. Los SGBD son piezas de software complejas, optimizados para ciertos tipos de cargas de trabajo (por ejemplo, la respuesta a consultas complejas o el manejo de muchas solicitudes concurrentes), y puede que su rendimiento no sea el adecuado para ciertas aplicaciones especializadas. Entre los ejemplos están las aplicaciones con estrictas restricciones de tiempo real o tan sólo unas pocas operaciones críticas bien definidas para las cuales hay que escribir un código a medida eficiente. Otro motivo para no emplear un SGBD es que puede que una aplicación necesite manipular los datos de maneras no soportadas por el lenguaje de consultas. En esas situaciones, la vista abstracta de los datos ofrecida por el SGBD no coincide con las necesidades de la aplicación y, en realidad, estorba. A modo de ejemplo, las bases de datos relacionales no soportan el análisis flexible de datos de texto (aunque los fabricantes estén ampliando actualmente sus productos en esa dirección). Si el rendimiento especializado o las exigencias del tratamiento de los datos son fundamentales para una aplicación, ésta puede no utilizar un SGBD, especialmente si las ventajas añadidas de los SGBD (por ejemplo, la flexibilidad de las consultas, la seguridad, el acceso concurrente y la recuperación de fallos) no son necesarias. En la mayor parte de las situaciones que requieren la gestión de datos a gran escala, no obstante, los SGBD se han convertido en una herramienta indispensable.

1.5 DESCRIPCIÓN Y ALMACENAMIENTO DE DATOS EN LOS SGBD

El usuario de un SGBD está, en última instancia, preocupado por alguna empresa real, y los datos que hay que guardar describen diversos aspectos de esa empresa. Por ejemplo, en las universidades hay alumnos, profesores y asignaturas, y los datos de las bases de datos de las universidades describen esas entidades y sus relaciones.

Un **modelo de datos** es un conjunto de estructuras descriptivas de datos de alto nivel que oculta muchos detalles de almacenamiento de bajo nivel. Los SGBD permiten a los usuarios definir los datos que se van a almacenar en términos de un modelo de datos. La mayor parte de los sistemas actuales de gestión de bases de datos se basan en el **modelo relacional de datos**, en el que se centrará este libro.

Aunque el modelo de datos del SGBD oculta muchos detalles, pese a todo está más cercano al modo en que el SGBD almacena los datos que al modo en que el usuario piensa en la empresa subyacente. Los **modelos semánticos de datos** son modelos de datos de alto nivel más abstractos que facilitan que los usuarios obtengan una buena descripción inicial de los datos de las empresas. Estos modelos contienen una amplia variedad de estructuras que ayudan a describir la situación real de las aplicaciones. No se pretende que los SGBD soporten directamente todas esas estructuras; suele crearse alrededor de un modelo de datos que tiene tan sólo unas cuantas estructuras básicas, como puede ser el modelo relacional. El diseño de bases de datos en términos de un modelo semántico sirve como útil punto de partida y se traduce posteriormente en el diseño de una base de datos en términos del modelo de datos que soporta realmente el SGBD.

Un modelo semántico de datos muy utilizado, denominado modeloentidad-relación (ER) nos permitirá denotar de manera gráfica las entidades y las relaciones existentes entre ellas. El modelo ER se trata en el Capítulo 2.

1.5.1 El modelo relacional

En este apartado se ofrece una breve introducción al modelo relacional. La estructura central para la descripción de los datos en este modelo son las **relaciones**, que se pueden considerar conjuntos de **registros**.

La descripción de los datos en términos de un modelo de datos se denomina **esquema**. En el modelo relacional los esquemas de las relaciones especifican su nombre, el nombre de cada **campo** (o **atributo** o **columna**) y el tipo de cada campo. A modo de ejemplo, puede que la información sobre los alumnos de la base de datos de una universidad se guarde en una relación con el esquema siguiente:

Alumnos(*ide:* `string`, *nombre:* `string`, *usuario:* `string`,
edad: `integer`, *nota:* `real`)

El esquema anterior indica que cada registro de la relación Alumnos tiene cinco campos, con los nombres y tipos de los campos que se indican. En la Figura 1.1 puede verse un ejemplar de la relación Alumnos.

Un ejemplo de mal diseño. El esquema relacional Alumnos muestra una mala decisión de diseño; *nunca* se deben crear campos como *edad*, cuyo valor cambia constantemente. Una opción mejor hubiera sido *FDN* (por *fecha de nacimiento*); la edad se puede calcular a partir de ella. No obstante, en los ejemplos se seguirá empleando *edad*, ya que los hace más fáciles de leer.

ide	*nombre*	*usuario*	*edad*	*nota*
53666	Jiménez	jimenez@inf	18	6,8
53688	Sánchez	sanchez@ii	18	6,4
53650	Sánchez	sanchez@mat	19	7,6
53831	Martínez	martinez@musica	11	3,6
53832	García	garcia@musica	12	4,0

Figura 1.1 Ejemplar de la relación Alumnos

Cada fila de la relación Alumnos es un registro que describe a un alumno. La descripción no está completa —por ejemplo, no se ha incluido la altura de cada alumno— pero, presumiblemente, resulta adecuada para las aplicaciones deseadas de una base de datos universitaria. Cada fila sigue el esquema de la relación Alumnos. El esquema puede, por tanto, considerarse como una plantilla para la descripción de los alumnos.

Se puede hacer más precisa la descripción de un conjunto de alumnos especificando **restricciones de integridad**, que son condiciones que deben satisfacer los registros de una relación. Por ejemplo, se podría especificar que cada alumno tenga un valor único de ide. Obsérvese que no se puede capturar esta información añadiendo simplemente otro campo al esquema Alumnos. Por tanto, la posibilidad de especificar la unicidad de los valores de un campo aumenta la precisión con que se pueden describir los datos. La expresividad de las estructuras disponibles para la especificación de restricciones de integridad es un aspecto importante de los modelos de datos.

Otros modelos de datos

Además del modelo de datos relacional (que se utiliza en numerosos sistemas, como DB2 de IBM, Informix, Oracle, Sybase, Access de Microsoft, FoxBase, Paradox, Tandem y Teradata) hay otros modelos de datos importantes, como el modelo jerárquico (que se emplea, por ejemplo en IDS y en IDMS), el modelo orientado a objetos (que se usa, por ejemplo, en Objectstore y en Versant) y el modelo relacional orientado a objetos (que se utiliza, por ejemplo, en productos de SGBD de IBM, Informix, ObjectStore, Oracle, Versant y otros). Aunque muchas bases de datos emplean los modelos jerárquico y de red, y los sistemas basados en los modelos orientado a objetos y relacional orientado a objetos están ganando aceptación en el mercado, el modelo dominante en la actualidad es el relacional.

En este libro nos centraremos en el modelo relacional debido a su empleo generalizado y a su importancia. En realidad, el modelo relacional orientado a objetos, que está ganando

popularidad, es un esfuerzo para combinar las mejores características de los modelos relacional y orientado a objetos, y es necesaria una buena comprensión del modelo relacional para entender los conceptos relacionales orientados a objetos.

1.5.2 Niveles de abstracción en los SGBD

Los datos en los SGBD se describen en tres niveles de abstracción, como puede verse en la Figura 1.2. La descripción de las bases de datos consta de un esquema en cada uno de esos tres niveles de abstracción: *conceptual*, *físico* y *externo*.

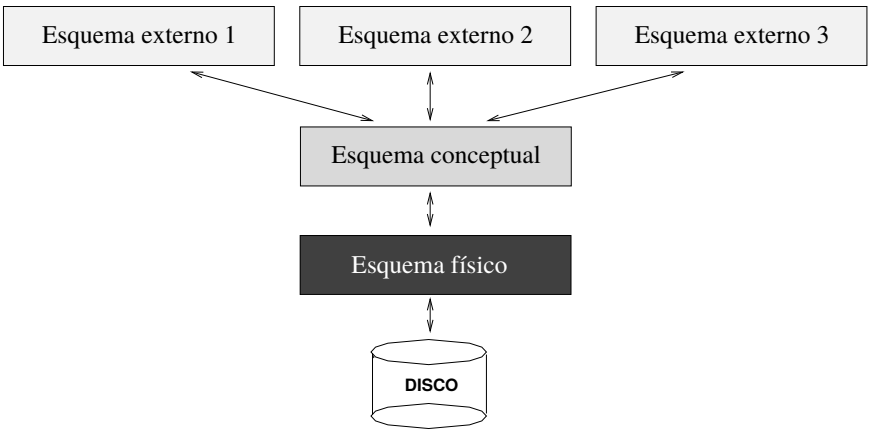

Figura 1.2 Niveles de abstracción en los SGBD

Los **lenguajes de definición de datos** (LDD) se emplean para definir los esquemas externo y conceptual. Las posibilidades como LDD del lenguaje de bases de datos más utilizado, SQL, se tratan en el Capítulo 3. Todos los fabricantes de SGBD soportan también las órdenes de SQL para la descripción de aspectos del esquema físico, pero esas órdenes no forman parte de la norma del lenguaje SQL. La información sobre los esquemas conceptual, externo y físico se guarda en los **catálogos del sistema**. Los tres niveles de abstracción se tratarán en el resto de este apartado.

Esquema conceptual

El **esquema conceptual** (denominado a veces **esquema lógico**) describe los datos almacenados en términos del modelo de datos del SGBD. En un SGBD relacional, el esquema conceptual describe todas las relaciones almacenadas en la base de datos. En la base de datos universitaria del ejemplo, esas relaciones contienen información sobre *entidades*, como los alumnos y el profesorado, y sobre *relaciones*, como la matrícula de los alumnos en las asignaturas. Todas las entidades alumno se pueden describir empleando registros de la relación Alumnos, como ya se ha visto. De hecho, cada conjunto de entidades y cada conjunto de relaciones pueden describirse como relaciones, lo que lleva al siguiente esquema conceptual:

Alumnos(*ide:* `string`, *nombre:* `string`, *usuario:*
`string`, *edad:* `integer`, *nota:* `real`)

Profesores(*idp:* `string`, *nombrep:* `string`, *sueldo:* `real`)
Asignaturas(*ida:* `string`, *nombrea:* `string`, *créditos:* `integer`)
Aulas(*nau:* `integer`, *dirección:* `string`, *capacidad:* `integer`)
Matriculado(*ide:* `string`, *ida:* `string`, *curso:* `string`)
Imparte(*idp:* `string`, *ida:* `string`)
Impartida_en(*ida:* `string`, *nau:* `integer`, *hora:* `string`)

La elección de las relaciones, y la de los campos de cada relación, no resulta siempre evidente, y el proceso de búsqueda de un buen esquema conceptual se denomina **diseño conceptual de bases de datos**. El diseño conceptual de bases de datos se trata en los Capítulos 2 y 12.

Esquema físico

El **esquema físico** especifica detalles adicionales del almacenamiento. Esencialmente, el esquema físico resume el modo en que las relaciones descritas en el esquema conceptual se guardan realmente en dispositivos de almacenamiento secundario como discos y cintas.

Es necesario decidir la organización de archivos que se va a utilizar para guardar las relaciones y crear estructuras de datos auxiliares, denominadas **índices**, para acelerar las operaciones de recuperación. Un ejemplo de esquema físico para la base de datos universitaria es el siguiente:

- Almacenar todas las relaciones como archivos de registros sin ordenar. (Cada archivo de un SGBD es un conjunto de registros o un conjunto de páginas, en vez de una cadena de caracteres como en los sistemas operativos.)
- Crear índices en la primera columna de las relaciones Alumnos, Profesores y Asignaturas, en la columna *sueldo* de Profesores y en la columna *capacidad* de Aulas.

Las decisiones sobre el esquema físico se basan en conocer el modo en que se suele tener acceso a los datos. El proceso de obtención de un buen esquema físico se denomina **diseño físico de bases de datos**. El diseño físico de bases de datos se trata en el Capítulo 13.

Esquema externo

Los **esquemas externos**, que suelen serlo también en términos del modelo de datos del SGBD, permiten personalizar (y autorizar) el acceso a los datos a los usuarios y grupos de ellos. Cualquier base de datos tiene exactamente un esquema conceptual y un esquema físico, porque sólo tiene guardado un conjunto de relaciones, pero puede tener varios esquemas externos, cada uno de ellos adaptado a un grupo de usuarios concreto. Cada esquema externo consiste en un conjunto de una o varias **vistas** y relaciones del esquema conceptual. Una vista es, conceptualmente, una relación pero los registros de la vista no se guardan en el SGBD. Por el contrario, se calculan empleando una definición de la vista en términos de las relaciones guardadas en el SGBD. Las vistas se tratan con más detalle en los Capítulos 3 y 16.

El diseño del esquema externo se guía por las necesidades del usuario final. Por ejemplo, puede que se desee que los alumnos averigüen el nombre de los profesores que imparten cada curso y la matrícula de cada curso. Esto puede hacerse definiendo la vista siguiente:

Infoasignatura(*ida:* `string`, *nombrep:* `string`, *matrícula:* `integer`)

Los usuarios pueden tratar la vista igual que a una relación y formularle preguntas sobre sus registros. Aunque estos registros no se guarden de manera explícita, se calculan a medida que hacen falta. Infoasignatura no se incluyó en el esquema conceptual porque se puede calcular a partir de las relaciones de ese esquema, y guardarla también resultaría redundante. Esa redundancia, además del espacio desaprovechado, podría dar lugar a inconsistencias. Por ejemplo, se puede insertar una tupla en la relación Matriculado que indique que un alumno dado se ha matriculado en un curso concreto, sin incrementar el valor del campo *matrícula* del registro correspondiente de Infoasignatura (si esta última forma también parte del esquema conceptual y sus tuplas se guardan en el SGBD).

1.5.3 Independencia con respecto a los datos

Una ventaja muy importante del empleo de un SGBD es que ofrece **independencia con respecto a los datos**. Es decir, los programas de aplicación quedan aislados de las modificaciones debido al modo en que se estructuran y se guardan los datos. La independencia con respecto a los datos se consigue mediante el empleo de los tres niveles de abstracción de los datos; en concreto, el esquema conceptual y el externo ofrecen claras ventajas en este campo.

Las relaciones en el esquema externo (relaciones de vistas) se generan, en principio, a petición de los usuarios a partir de las relaciones correspondientes del esquema conceptual[4]. Si se reorganiza la capa subyacente, es decir, se modifica el esquema conceptual, se puede modificar la definición de la relación de vistas para que esa relación se siga calculando como antes. Por ejemplo, supóngase que la relación Profesores de la base de datos universitaria se sustituye por las dos relaciones siguientes:

Profesores_públicos(*idp:* `string`, *nombrep:* `string`, *despacho:* `integer`)
Profesores_privados(*idp:* `string`, *sueldo:* `real`)

De manera intuitiva, cierta información confidencial sobre el profesorado se ha colocado en una relación diferente y se ha añadido la información relativa a los despachos. Se puede redefinir la relación de vistas Infoasignatura en términos de Profesores_públicos y Profesores_privados, que contienen entre las dos toda la información de Profesores, por lo que los usuarios que consulten Infoasignatura obtendrán las mismas respuestas que antes.

Por tanto, se puede proteger a los usuarios de las modificaciones en la estructura lógica de los datos, o en la elección de las relaciones que se guardan. Esta propiedad se denomina **independencia con respecto a los datos**.

A su vez, el esquema conceptual aisla a los usuarios respecto de las modificaciones en los detalles del almacenamiento físico. Esta propiedad se conoce como **independencia física con respecto a los datos**. El esquema conceptual oculta detalles como el modo en que se

[4] En la práctica se podrían calcular con antelación y guardarse para acelerar las consultas relativas a las relaciones de vistas, pero las relaciones de vistas calculadas deben actualizarse siempre que se actualicen las relaciones subyacentes.

disponen realmente los datos en el disco, la estructura de los archivos y la elección de los índices. En tanto en cuanto el esquema conceptual siga siendo el mismo, se pueden modificar esos detalles del almacenamiento sin alterar las aplicaciones. (Por supuesto, puede que el rendimiento se vea afectado por esas modificaciones.)

1.6 LAS CONSULTAS EN LOS SGBD

La facilidad con la que se puede obtener información de las bases de datos suele determinar su valor para los usuarios. A diferencia de los sistemas antiguos de bases de datos, los sistemas relacionales de bases de datos permiten formular con facilidad una amplia gama de preguntas; esta característica ha contribuido mucho a su popularidad. Considérese la base de datos universitaria del ejemplo del Apartado 1.5.2. He aquí algunas de las preguntas que los usuarios podrían plantear:

1. ¿Cómo se llama el alumno con ID 123456?
2. ¿Cuál es el sueldo medio de los profesores que imparten la asignatura CS564?
3. ¿Cuántos alumnos se han matriculado en CS564?
4. ¿Qué proporción de los alumnos de CS564 obtuvieron una nota superior a notable?
5. ¿Hay algún alumno con nota menor de 6,0 matriculado en CS564?

Esas preguntas relativas a los datos guardados en el SGBD se denominan **consultas**. Los SGBD proporcionan un lenguaje especializado, denominado **lenguaje de consultas**, en el que se pueden formular las consultas. Una característica muy atractiva del modelo relacional es que soporta lenguajes de consulta potentes. El **cálculo relacional** es un lenguaje de consultas formal basado en la lógica matemática, y las consultas en ese lenguaje tienen un significado preciso e intuitivo. El álgebra relacional es otro lenguaje formal de consultas, basado en un conjunto de **operadores** para la manipulación de las relaciones, que es equivalente en potencia al cálculo relacional.

Los SGBD tiene mucho cuidado en evaluar las consultas de la manera más eficiente posible. Por supuesto, la eficiencia de la evaluación de las consultas viene determinada en gran parte por la manera en que los datos se han almacenado físicamente. Se pueden utilizar índices para acelerar muchas consultas —de hecho, una buena elección de índices para las relaciones subyacentes puede acelerar cada una de las consultas de la lista anterior—. El almacenamiento de datos y los índices se tratan en los Capítulos 9, 10 y 11.

Los SGBD permiten a los usuarios crear, modificar y consultar datos mediante los **lenguajes de manipulación de datos** (LMD). Por tanto, el lenguaje de consultas es tan sólo una parte del LMD, que también proporciona estructuras para añadir, eliminar y modificar datos. Las características LMD de SQL se tratan en el Capítulo 5. LMD y LDD se conocen en conjunto como **sublenguaje de datos** cuando se incluyen en un **lenguaje anfitrión** (como, por ejemplo, C o COBOL).

1.7 GESTIÓN DE TRANSACCIONES

Considérese una base de datos que guarde información sobre reservas de billetes de avión. En cualquier momento dado es posible (y probable) que varios agentes de viajes estén buscando información sobre plazas disponibles en diferentes vuelos y haciendo nuevas reservas de billetes. Cuando varios usuarios tienen acceso (y, posiblemente, modifican) una base de datos de manera concurrente, el SGBD debe ordenar sus solicitudes con sumo cuidado para evitar conflictos. Por ejemplo, cuando un agente de viajes busca el Vuelo 100 de un día dado y halla una plaza libre, puede que otro agente de viajes esté haciendo al mismo tiempo una reserva para esa misma plaza, lo que deja obsoleta la información vista por el primero.

Otro ejemplo de uso concurrente son las bases de datos de las entidades bancarias. Mientras el programa de aplicación de un usuario está calculando el total de depósitos, puede que otra aplicación transfiera dinero de una cuenta que la primera acabe de "ver" a una cuenta que todavía no ha "visto", lo que hace que el total parezca mayor de lo que debería. Evidentemente, no se debe permitir que ocurran tales anomalías. Sin embargo, impedir el acceso concurrente puede degradar el rendimiento.

Además, los SGBD deben proteger a los usuarios de los efectos de los fallos del sistema asegurando que todos los datos (y el estado de las aplicaciones activas) vuelvan a un estado consistente cuando se reinicie el sistema tras un fallo. Por ejemplo, si un agente de viajes pide que se lleve a cabo una reserva y el SGBD responde diciendo que esa reserva se ha completado, la reserva no se debe perder si el sistema falla. Por otro lado, si el SGBD no ha respondido todavía a esa solicitud pero está llevando a cabo las necesarias modificaciones a los datos cuando se produce el fallo, esas modificaciones parciales se deben deshacer cuando el sistema vuelva a estar en funcionamiento.

Una **transacción** es *cualquier ejecución* de un programa de usuario en un SGBD. (Las ejecuciones subsecuentes del mismo programa generan varias transacciones.) Se trata de la unidad básica de modificación desde el punto de vista del SGBD: no se permiten transacciones parciales, y el efecto de un grupo de transacciones es equivalente a la ejecución en serie de todas las transacciones. A continuación se describe brevemente la manera en que se garantizan estas propiedades.

1.7.1 Ejecución concurrente de las transacciones

Una importante tarea de los SGBD es la programación de los accesos concurrentes a los datos de modo que cada usuario pueda ignorar con seguridad el hecho de que otros tienen acceso a los datos de manera concurrente. La importancia de esta tarea no puede subestimarse, ya que las bases de datos suelen estar compartidas por gran número de usuarios, que remiten sus consultas al SGBD de manera independiente y, sencillamente, no puede esperarse que afronten las modificaciones arbitrarias que realicen otros usuarios de manera concurrente. El SGBD permite que los usuarios crean que sus programas se ejecutan aisladamente, uno tras otro en el orden escogido por el SGBD. Por ejemplo, si el programa que deposita efectivo en una cuenta se remite al SGBD al mismo tiempo que otro programa que retira dinero de esa misma cuenta, el SGBD puede ejecutar cualquiera de los dos, pero sus procesos no se intercalarán de modo que interfieran entre sí.

Los **protocolos de bloqueo** son conjuntos de reglas que debe seguir cada transacción (y que el SGBD debe hacer que se cumplan) para garantizar que, aunque las acciones de varias transacciones se intercalen, el efecto neto sea idéntico a la ejecución de todas las transacciones en un orden consecutivo dado. Un **bloqueo** es un mecanismo empleado para controlar el acceso a los objetos de la base de datos. Los SGBD suelen soportar dos tipos de **bloqueos**: los **compartidos** sobre un objeto pueden establecerlos dos transacciones diferentes al mismo tiempo, pero el **bloqueo exclusivo** de un objeto garantiza que ninguna otra transacción establezca *un* bloqueo sobre ese objeto.

Supóngase que se sigue el siguiente protocolo de bloqueo: *todas las transacciones comienzan por la obtención de un bloqueo compartido sobre cada objeto de datos que deban modificar y luego liberan todos los bloqueos tras completar todas las acciones*. Considérense dos transacciones $T1$ y $T2$ tales que $T1$ desea modificar un objeto de datos y $T2$ desea leer ese mismo objeto. De manera intuitiva, si la solicitud por $T1$ de un bloqueo exclusivo se concede en primer lugar, $T2$ no puede seguir adelante hasta que $T1$ libere ese bloqueo, ya que el SGBD no concederá la solicitud de un bloqueo compartido por parte de $T2$ hasta entonces. Por tanto, todas las acciones de $T1$ se completarán antes de que se inicie ninguna acción de $T2$. Se considerarán los bloqueos con más detalle en el Capítulo 8.

1.7.2 Las transacciones no completadas y los fallos del sistema

Las transacciones pueden interrumpirse antes de completarse por gran variedad de motivos como, por ejemplo, un fallo del sistema. Los SGBD deben garantizar que las modificaciones llevadas a cabo por esas transacciones no completadas se eliminen de las bases de datos. Por ejemplo, si el SGBD se halla en trance de transferir dinero de la cuenta A a la cuenta B y ha cargado la operación en la primera cuenta pero todavía no la ha anotado en la segunda cuando se produce el fallo, se debe devolver el dinero cargado a la cuenta A cuando el sistema se recupere del fallo.

Para ello, el SGBD mantiene un **registro** de todas las operaciones de escritura en la base de datos. Una propiedad fundamental de ese registro es que cada acción de escritura debe registrarse en el registro (en disco) *antes* de que la modificación correspondiente se refleje en la propia base de datos —en caso contrario, si el sistema fallase justo tras la realización de la modificación de la base de datos pero antes de que ésta se reflejara en el registro, el SGBD no sería capaz de detectarla y deshacerla—. Esta propiedad se denomina **registro de escritura previa** (Write-Ahead Log, **WAL**). Para garantizarla, el SGBD debe poder obligar de manera selectiva a que se guarde en el disco una página que se halle en la memoria.

El registro se utiliza también para garantizar que las modificaciones llevadas a cabo por las transacciones completadas con éxito no se pierdan debido a fallos del sistema. La devolución de la base de datos a un estado consistente tras un fallo del sistema puede ser un proceso lento, ya que el SGBD debe garantizar que el efecto de todas las transacciones que se completaron antes del fallo se restaure, y que el de las transacciones no completadas se deshaga. El tiempo necesario para recuperarse de un fallo puede reducirse obligando de manera periódica a que cierta información se guarde en disco; esas operaciones periódicas se denominan **puntos de revisión**.

1.7.3 Puntos a destacar

En resumen, hay tres puntos que recordar en relación con el soporte del control de la concurrencia y las recuperaciones de los SGBD:

1. Todos los objetos que lee o escribe una transacción se bloquean antes de modo compartido o exclusivo, respectivamente. El establecimiento de un bloqueo sobre un objeto restringe su disponibilidad para otras transacciones y, por tanto, afecta al rendimiento.
2. Para un mantenimiento eficiente del registro, el SGBD debe poder obligar de manera selectiva a que un conjunto de páginas de la memoria principal se guarde en el disco. El soporte de esta operación por parte del sistema operativo no siempre resulta satisfactorio.
3. El establecimiento de puntos de revisión periódicos puede reducir el tiempo necesario para la recuperación de un fallo. Por supuesto, esto debe contraponerse al hecho de que el establecimiento de puntos de revisión demasiado próximos hace más lenta la ejecución normal.

1.8 ARQUITECTURA DE LOS SGBD

La Figura 1.3 muestra la arquitectura (con algunas simplificaciones) de un SGBD típico basado en el modelo relacional de datos.

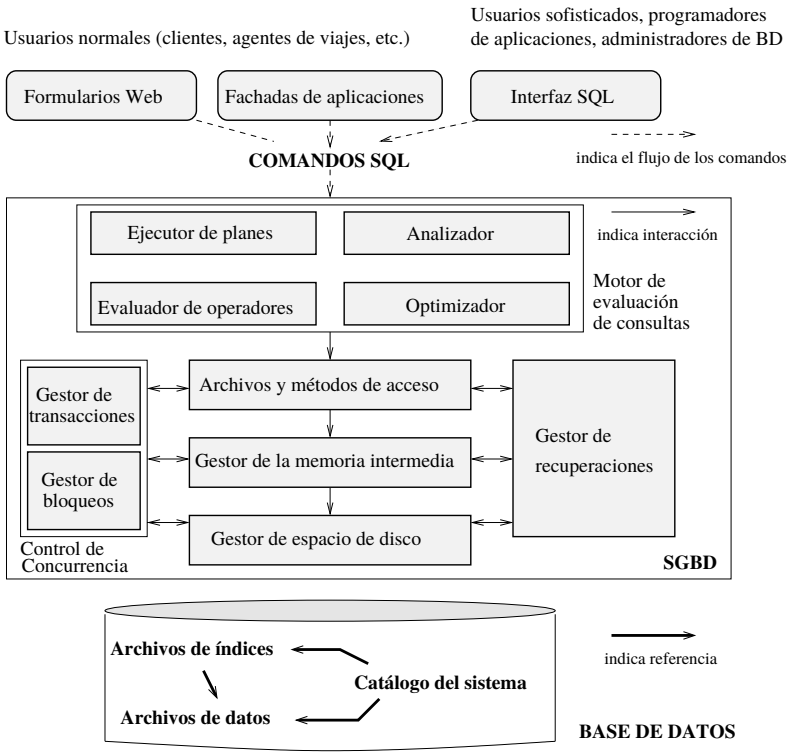

Figura 1.3 Arquitectura de un SGBD

El SGBD acepta las órdenes SQL generadas por gran variedad de interfaces de usuario, produce planes de evaluación de consultas, ejecuta esos planes contra la base de datos y devuelve las respuestas. (Esto no es más que una simplificación: las órdenes de SQL pueden estar incluidas en programas de aplicaciones de lenguajes anfitriones como, por ejemplo, programas de Java o de COBOL. Estos aspectos se pasan por alto para concentrarnos en la funcionalidad principal del SGBD.)

Cuando un usuario formula una consulta, se analiza y envía esta consulta a un **optimizador de consultas**, que utiliza información sobre el modo en que se guardan los datos para producir un plan de ejecución eficiente para la evaluación de esa consulta. Un **plan de ejecución** es un plan detallado para la evaluación de la consulta, representado habitualmente como un árbol de operadores relacionales (con anotaciones que contienen información detallada adicional sobre los métodos de acceso que se deben emplear, etcétera). Los operadores relacionales son como los elementos constitutivos de la evaluación de las consultas planteadas a los datos.

El código que implementa los operadores relacionales se sitúa por encima de la capa de los archivos y los métodos de acceso. Esta capa soporta el concepto de **archivo**, que es un conjunto de páginas o de registros en los SGBD. Se admiten tanto los **archivos en montículos**, o archivos de páginas sin ordenar, así como los índices. Además de realizar el seguimiento de las páginas de los archivos, esta capa organiza la información en el interior de cada página. Las organizaciones de los archivos y de los índices se consideran en el Capítulo 9.

El código de la capa de archivos y métodos de acceso se sitúan por encima del **gestor de la memoria intermedia**, que lleva las páginas desde el disco a la memoria principal según va haciendo falta, en respuesta a las solicitudes de lectura.

La capa inferior del software de los SGBD se ocupa de la administración del espacio de disco, donde se almacenan los datos. Las capas superiores asignan, desasignan, leen y escriben las páginas (mediante las oportunas rutinas) a través de esta capa, denominada **gestor del espacio de disco**.

Los SGBD soportan la concurrencia y la recuperación de fallos mediante la cuidadosa programación de las solicitudes de los usuarios y el mantenimiento de un registro de todas las modificaciones de la base de datos. Entre los componentes del SGBD asociados al control de la concurrencia y la recuperación están el **gestor de transacciones**, que garantiza que las transacciones soliciten y liberen los bloqueos de acuerdo con el correspondiente protocolo de bloqueo y programa la ejecución de las transacciones; el **gestor de bloqueos**, que realiza un seguimiento de las solicitudes de bloqueo y concede los bloqueos sobre los objetos de la base de datos cuando quedan disponibles; y el **gestor de recuperaciones**, que es responsable del mantenimiento de un registro y de la restauración del sistema a un estado consistente tras los fallos. El gestor del espacio de disco, el gestor de la memoria intermedia, y las capas de archivos y métodos de acceso deben interactuar con estos componentes. El control de la concurrencia y la recuperación tras los fallos se estudian con detalle en el Capítulo 8.

1.9 USUARIOS DE LAS BASES DE DATOS

Existe una gran variedad de personas asociadas con la creación y el empleo de bases de datos. Evidentemente, hay **implementadores de bases de datos**, que crean software de SGBD, y **usuarios finales**, que desean guardar y utilizar datos de los SGBD. Los implementadores de bases de datos trabajan para fabricantes como IBM u Oracle. Los usuarios finales vienen de un número de campos diverso y creciente. A medida que los datos aumentan en complejidad y en volumen, y se van reconociendo como un activo fundamental, la importancia de mantenerlos en los SGBD se acepta cada vez más. Muchos usuarios finales se limitan a utilizar aplicaciones escritas por programadores de aplicaciones de bases de datos (véase más abajo) y, por tanto, no necesitan tener muchos conocimientos técnicos sobre el software de SGBD. Por supuesto, los usuarios sofisticados que hacen un uso más amplio de los SGBD, como la escritura de sus propias consultas, necesitan una comprensión más profunda de sus características.

Además de los usuarios finales y de los implementadores, hay otras dos clases de usuarios asociados con los SGBD: los *programadores de aplicaciones* y los *administradores de bases de datos*.

Los **programadores de aplicaciones de bases de datos** desarrollan paquetes que facilitan el acceso a los datos por parte de los usuarios finales, que generalmente no son profesionales de la informática, mediante lenguajes anfitriones o de datos y herramientas software que proporcionan los fabricantes de SGBD. (Entre esas herramientas figuran las herramientas para la escritura de informes, las hojas de cálculo, los paquetes de estadística y similares.) Los programas de aplicación deberían tener acceso a los datos, en teoría, mediante el esquema externo. Es posible escribir aplicaciones que tengan acceso a los datos en un nivel inferior, pero esas aplicaciones pondrían en peligro la independencia con respecto a los datos.

Las bases de datos suelen estar mantenidas por la persona que las posee y las utiliza. Sin embargo, las bases de datos corporativas suelen ser lo bastante importantes y complejas como para que la tarea de diseñarlas y mantenerlas se confíe a profesionales, denominados **administradores de bases de datos** (database administrator, **DBA**). Los DBA son responsables de muchas tareas críticas:

- El **diseño de los esquemas conceptual y físico.** El DBA es responsable de interactuar con los usuarios del sistema para comprender los datos que se van a guardar en el SGBD y la manera en que es más probable que se utilicen. Con ese conocimiento, el DBA debe diseñar el esquema conceptual (decidir las relaciones que se van a guardar) y el físico (decidir la manera de guardarlas). Puede que el DBA también diseñe las partes más utilizadas del esquema externo, aunque los usuarios hagan crecer este esquema mediante la creación de vistas adicionales.

- **Seguridad y autorización.** El DBA es responsable de garantizar que no se permita el acceso sin autorización a los datos. En general, no todo el mundo debe poder tener acceso a todos los datos. En los SGBD relacionales se puede conceder permiso a los usuarios para el acceso tan sólo a determinadas vistas y relaciones. Por ejemplo, aunque puede que se permita que los alumnos averigüen las matriculas de cada asignatura y quién las imparte, no es probable que se desee que vean los sueldos de los profesores o la información sobre las notas de los demás alumnos. El DBA puede aplicar esta política mediante la concesión a los alumnos de permiso para leer tan sólo la vista Infoasignatura.

- **Disponibilidad de los datos y recuperación en caso de fallo.** El DBA debe tomar medidas para garantizar que, en caso de fallo del sistema, los usuarios puedan seguir teniendo acceso a tanta cantidad de datos que no se hayan deteriorado como sea posible. El DBA también debe trabajar para devolver los datos a un estado consistente. Los SGBD ofrecen soporte de software para estas funciones, pero el DBA es responsable de la implementación de procedimientos para la realización periódica de copias de seguridad y para el mantenimiento de los registros de la actividad del sistema (para facilitar la recuperación en caso de fallo).
- **Ajuste de las bases de datos.** Es probable que las necesidades de los usuarios evolucionen con el tiempo. El DBA es responsable de modificar la base de datos, en especial los esquemas conceptual y físico, para garantizar un rendimiento adecuado a medida que varíen las exigencias.

1.10 PREGUNTAS DE REPASO

Las respuestas a las preguntas de repaso pueden hallarse en los apartados indicados.

- ¿Cuáles son las ventajas principales del empleo de SGBD para administrar los datos en aplicaciones que impliquen un amplio acceso a los datos? **(Apartados 1.1, 1.4)**
- ¿Cuándo se deben guardar los datos en SGBD en lugar de hacerlo en los archivos del sistema operativo y viceversa? **(Apartado 1.3)**
- ¿Qué son los modelos de datos? ¿Qué es el modelo de datos relacional? ¿Qué es la independencia con respecto a los datos y cómo la soportan los SGBD? **(Apartado 1.5)**
- Explíquense las ventajas de emplear un lenguaje de consultas para procesar los datos en lugar de los programas habituales. **(Apartado 1.6)**
- ¿Qué son las transacciones? ¿Qué garantías ofrecen los SGBD con respecto a las transacciones? **(Apartado 1.7)**
- ¿Qué son los bloqueos de los SGBD y por qué se emplean? ¿Qué es el registro previo a la escritura y por qué se utiliza? ¿Qué son los puntos de control y por qué se usan? **(Apartado 1.7)**
- Identifíquense los principales componentes de los SGBD y explíquese brevemente lo que hace cada uno de ellos. **(Apartado 1.8)**
- Explíquense los diferentes papeles de los administradores de bases de datos, los programadores de aplicaciones y los usuarios finales de las bases de datos. ¿Quién necesita saber más sobre los sistemas de bases de datos? **(Apartado 1.9)**

EJERCICIOS

Ejercicio 1.1 ¿Por qué elegir un sistema de bases de datos en lugar de limitarse a guardar los datos en los archivos del sistema operativo? ¿Cuándo tendría sentido *no* emplear un sistema de bases de datos?

Ejercicio 1.2 ¿Qué es la independencia lógica con respecto a los datos y por qué es importante?

Ejercicio 1.3 Explíquese la diferencia entre la independencia lógica con respecto a los datos y la física.

Ejercicio 1.4 Explíquense las diferencias entre los esquemas externo, interno y conceptual. ¿Cómo están relacionadas estas capas de esquemas con los conceptos de independencia lógica y física con respecto a los datos?

Ejercicio 1.5 ¿Cuáles son las responsabilidades de los DBA? Si se supone que el DBA no está interesado en ejecutar nunca sus propias consultas, ¿sigue necesitando comprender la optimización de las consultas? ¿Por qué?

Ejercicio 1.6 Avaro Puñocerrado desea guardar la información (nombres, direcciones, descripciones de momentos embarazosos, etc.) sobre las muchas víctimas de su lista. Como era de esperar, el volumen de datos le impulsa a comprar un sistema de bases de datos. Para ahorrar dinero, desea comprar uno con las mínimas características posibles, y piensa ejecutarlo como aplicación independiente en su ordenador personal. Por supuesto, Avaro no piensa compartir esa lista con nadie. Indíquese por cuáles de las siguientes características de los SGBD debe pagar Avaro; en cada caso, indíquese también el motivo de que Avaro deba (o no) pagar por esa característica del sistema que va a comprar.

1. Un dispositivo de seguridad.
2. Control de la concurrencia.
3. Recuperación de fallos.
4. Un mecanismo de vistas.
5. Un lenguaje de consultas.

Ejercicio 1.7 ¿Cuál de los elementos siguientes desempeña un papel importante en la *representación* de la información sobre el mundo real en las bases de datos? Explíquese brevemente.

1. El lenguaje de definición de datos.
2. El lenguaje de manipulación de datos.
3. El gestor de la memoria intermedia.
4. El modelo de datos.

Ejercicio 1.8 Descríbase la estructura de un SGBD. Si se actualiza el sistema operativo para que soporte alguna función nueva en los archivos del sistema operativo (por ejemplo, la posibilidad de obligar a que se guarde en disco alguna secuencia de bytes), ¿qué capa(s) del SGBD habría que volver a escribir para aprovechar esas funciones nuevas?

Ejercicio 1.9 Respóndanse las preguntas siguientes:

1. ¿Qué son las transacciones?
2. ¿Por qué los SGBD intercalan las acciones de las diferentes transacciones en lugar de ejecutar una transacción después de otra?
3. Qué deben garantizar los usuarios con respecto a las transacciones y a la consistencia de las bases de datos? ¿Qué deben garantizar los SGBD con respecto a la ejecución concurrente de varias transacciones y a la consistencia de las bases de datos?
4. Explíquese el protocolo de bloqueo estricto de dos fases.
5. ¿Qué es la propiedad WAL y por qué es importante?

EJERCICIOS BASADOS EN PROYECTOS

Ejercicio 1.10 Empléese un navegador Web para examinar la documentación en HTML de Minibase. Inténtese captar la idea de su estructura global.

NOTAS BIBLIOGRÁFICAS

La evolución de los sistemas de gestión de bases de datos se describe en [189]. El empleo de modelos de datos para la descripción de los datos del mundo real se trata en [270], mientras que [271] contiene una taxonomía de los modelos de datos. Los tres niveles de abstracción se introdujeron en [118, 432]. El modelo de datos en red se describe en [118], mientras que [472] trata de varios sistemas comerciales basados en ese modelo. [438] contiene un buen conjunto comentado de trabajos orientados a los sistemas sobre la gestión de las bases de datos.

Entre los textos que tratan de los sistemas de gestión de bases de datos están [137, 157, 199, 220, 298, 347, 414, 455, 463]. En [136] se ofrece un estudio detallado del modelo relacional desde un punto de vista conceptual y es destacable por su amplia bibliografía comentada. [346] presenta una perspectiva orientada al rendimiento con referencias a varios sistemas comerciales. En [157] y [414] se ofrece una amplia cobertura de los conceptos de los sistemas de bases de datos, incluido un estudio de los modelos de datos jerárquico y de red. [220] pone el énfasis en la conexión entre los lenguajes de consultas para bases de datos y la programación lógica. [463] se centra en los modelos de datos. De estos textos, [455] ofrece el tratamiento más detallado de los aspectos teóricos. Entre los textos dedicados a los aspectos teóricos están [1, 29, 309]. El libro [452] incluye un apartado sobre bases de datos que contiene artículos de investigación introductorios sobre diversos temas.