Introducción

01

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 misma. 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 un 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.

1.1. Aplicaciones de los sistemas de bases de datos

Las bases de datos se usan ampliamente. Algunas de sus aplicaciones representativas son:

- *Información empresarial*
  - *Ventas:* para información de clientes, productos y compras.
  - *Contabilidad:* para pagos, recibos, contabilidad, asientos y otra información contable.
  - *Recursos humanos:* para información sobre empleados, salarios, pago de impuestos y beneficios, así como para la generación de las nóminas.
  - *Fabricación:* para la gestión de la cadena de suministros y el seguimiento de la producción en fábrica, la gestión de inventarios en los almacenes y la gestión de pedidos.
  - *Comercio en línea:* para los datos de ventas ya mencionados y para el seguimiento de los pedidos web, generación de listas de recomendaciones y mantenimiento de evaluaciones de productos en línea.
- *Banca y finanzas*
  - *Banca:* para información de los clientes, cuentas, préstamos y transacciones bancarias.
  - *Transacciones de tarjetas de crédito:* para compras con tarjeta de crédito y la generación de los extractos mensuales.
  - *Finanzas:* para almacenar información sobre compañías tenedoras; ventas y compras de productos financieros, como acciones y bonos, y también para almacenar datos del mercado en tiempo real que permitan a los clientes la compraventa en línea y a la compañía la compraventa automática.
- *Universidades:* para la información sobre estudiantes, la inscripción en cursos y la graduación (además de la información habitual de recursos humanos y contabilidad).
- *Líneas aéreas:* para reservas e información de horarios. Las líneas aéreas fueron de las primeras en usar las bases de datos de forma distribuida geográficamente.
- *Telecomunicaciones:* para guardar un registro de las llamadas realizadas, generar las facturas mensuales, mantener el saldo de las tarjetas telefónicas de prepago y almacenar información sobre las redes de comunicaciones.

Como se muestra en la lista, en la actualidad las bases de datos son una parte esencial de todas las empresas, donde se almacena no solo información típica de cualquier empresa, sino también la que es específica de cada tipo de empresa.

Durante las últimas cuatro décadas del siglo xx, el uso de las bases de datos creció en todas las empresas. En los primeros días, muy pocas personas interactuaban directamente con los sistemas de bases de datos, aunque sin darse cuenta interactuaban indirectamente con bases de datos (con informes impresos, como los extractos de las tarjetas de crédito, o mediante agentes, como los cajeros de los bancos y los agentes de reservas de las líneas aéreas). Después vinieron los cajeros automáticos y permitieron a los usuarios interactuar directamente con las bases de datos. Las interfaces telefónicas con las computadoras (sistemas de respuesta vocal interactiva) también permitieron a los usuarios tratar directamente con las bases de datos (la persona que llamaba podía marcar un número y pulsar las teclas del teléfono para introducir información o para seleccionar opciones alternativas, para conocer las horas de llegada o salida de los vuelos, por ejemplo, o para matricularse de asignaturas en una universidad).

La revolución de Internet a finales de los años noventa aumentó significativamente el acceso directo del usuario a las bases de datos. Las organizaciones convirtieron muchas de sus interfaces telefónicas a las bases de datos en interfaces web, y dejaron disponibles en línea muchos servicios. Por ejemplo, cuando se accede a una librería en línea y se busca en una colección de libros o de música, se está accediendo a datos almacenados en una base de datos. Cuando se realiza un pedido en línea, el pedido se almacena en una base de datos. Cuando se accede al sitio web de un banco y se consultan el estado de la cuenta y los movimientos, la información se recupera del sistema de bases de datos del banco. Cuando se accede a un sitio web, puede que se recupere información personal de una base de datos para seleccionar los anuncios que se deben mostrar. Más aún, los datos sobre los accesos web pueden almacenarse en una base de datos.

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

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

1.2. Propósito de los sistemas de bases de datos

Los sistemas de bases de datos surgieron en respuesta a los primeros métodos de gestión informatizada de los datos comerciales. Como ejemplo de este tipo de métodos, típicos de los años sesenta, suponga una parte de una organización, como una universidad, que entre otros datos guarda información sobre profesores, estudiantes, departamentos y oferta de cursos. Una manera de guardar la información en la computadora es almacenarla en archivos del sistema operativo. Para permitir que los usuarios manipulen la información, el sistema tiene varios programas de aplicación que gestionan los archivos, incluyendo programas para:

- Añadir nuevos estudiantes, profesores y cursos.
- Inscribir a los estudiantes en cursos y generar grupos de clases.
- Poner notas a los estudiantes, calcular notas medias y generar certificados.

Estos programas de aplicación los han escrito programadores de sistemas en respuesta a las necesidades de la universidad.

Según se vayan necesitando, se añaden nuevas funciones al sistema. Por ejemplo, suponga que la universidad decide ofertar un nuevo título, digamos que en Informática. La universidad debe crear un nuevo departamento y nuevos archivos permanentes, o añadir información a los archivos existentes, para registrar la información de los profesores del departamento, los estudiantes que se encuentran en dicho título, la oferta de asignaturas, los requisitos de acceso, etc. Puede que se necesite también escribir programas para manejar las restricciones que existan para el nuevo título, así como reglas que imponga la universidad. En este sentido, según pasa el tiempo el sistema va ganando más y más archivos y programas de aplicación.

Los sistemas operativos convencionales soportan este **sistema de procesamiento de archivos** típico. El sistema almacena los registros permanentes en varios archivos y necesita diferentes programas de aplicación para extraer y añadir datos a los archivos correspondientes. Antes de la aparición de los sistemas gestores de bases de datos (SGBD), las organizaciones normalmente almacenaban la información en 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 periodo 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, si un estudiante está matriculado en una titulación doble, por ejemplo Música y Matemáticas, la dirección y el número de teléfono de ese estudiante pueden estar en un archivo con los registros de todos los estudiantes del departamento de Música y en otro archivo con los registros de los estudiantes del departamento de Matemáticas. Esta redundancia genera mayores necesidades de almacenamiento a un mayor coste. Además, puede generar **inconsistencia de datos,** es decir, puede que las distintas copias de los datos no coincidan; por ejemplo, el estudiante cambia su dirección en el registro del departamento de Matemáticas pero no en el resto del sistema.
- **Dificultad en el acceso a los datos.** Suponga que uno de los empleados de la universidad necesita conocer los nombres de todos los estudiantes que viven en un determinado código postal. Este 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, no hay un programa de aplicación para satisfacerla. Hay, sin embargo, un programa de aplicación que genera la lista de todos los estudiantes. El empleado tiene dos opciones: bien obtener la lista de *todos* los estudiantes y extraer manualmente la información que necesita, o bien pedir a un programador que escriba el programa de aplicación necesario. Ambas alternativas son obviamente poco satisfactorias. Suponga que se escribe el programa y que, varios días más tarde, el mismo empleado necesita reducir esa lista para que incluya únicamente a aquellos estudiantes matriculados al menos en 60 créditos. Como cabría 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 entonces 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.** Suponga que la universidad mantiene una cuenta para cada departamento y registros de los saldos de dicha cuenta. Suponga también, que la universidad requiere que el saldo de un departamento nunca sea menor que cero. 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 diversos archivos.
- **Problemas de atomicidad.** Los sistemas informáticos, como cualquier otro dispositivo mecánico o eléctrico, están sujetos a fallos. En muchas aplicaciones es crucial asegurar que, si se produce algún fallo, los datos se restauren al estado consistente que existía antes del fallo. Suponga un programa para transferir 500 € de la cuenta de un departamento A a la cuenta de un departamento B. Si se produce un fallo en el sistema durante la ejecución del programa, es posible que se hayan sacado 500 € de la cuenta del departamento A pero todavía no se hayan ingresado en el departamento B, generando un estado inconsistente de la base de datos. Evidentemente, para la consistencia de la base de datos resulta esencial que tengan lugar tanto el abono como el cargo, o que no tenga lugar ninguno. Es decir, la transferencia de fondos debe ser *atómica*; debe ocurrir en su totalidad o no ocurrir en absoluto. Resulta difícil asegurar la atomicidad en los sistemas convencionales de procesamiento de archivos.

- **Anomalías en el acceso concurrente.** Para aumentar el rendimiento global del sistema y obtener una respuesta más rápida, muchos sistemas permiten que varios usuarios actualicen los datos simultáneamente. En realidad, hoy en día los principales sitios de comercio electrónico de Internet pueden tener millones de accesos diarios de compradores a sus datos. En tales entornos es posible la interacción de actualizaciones concurrentes que pueden dar lugar a datos inconsistentes. Suponga que un departamento A tiene una cuenta con un saldo de 10.000 €. Si dos empleados retiran fondos (por ejemplo, 500 € y 100 €, respectivamente) de la cuenta *A* exactamente a la vez, el resultado de las ejecuciones concurrentes puede dejar la cuenta en un estado incorrecto, o inconsistente. Suponga que los programas que se ejecutan para cada retirada leen el saldo anterior, reducen su valor en el importe que se retira y luego escriben el resultado. Si los dos programas se ejecutan concurrentemente, pueden leer el valor 10.000 €, y escribir después 9.500 € y 9.900 €, respectivamente. Dependiendo de cuál escriba el valor en último lugar, la cuenta puede contener 9.500 € o 9.900 €, en lugar del valor correcto de 9.400 €. Para protegerse contra esta posibilidad, el sistema debe mantener alguna forma de supervisión. Pero es difícil ofrecer supervisión, ya que pueden tener acceso a los datos muchos programas de aplicación diferentes que no se hayan coordinado con anterioridad.

  Otro ejemplo: suponga que un programa de matriculación mantiene la cuenta de estudiantes matriculados en una asignatura, para restringir el número de estudiantes que se pueden matricular. Cuando un estudiante se matricula, el programa lee la cuenta actual, verifica que la cuenta no ha superado el límite, añade 1 a la cuenta y la guarda en la base de datos. Suponga que dos estudiantes se matriculan concurrentemente cuando la cuenta vale, por ejemplo, 39. Las dos ejecuciones del programa pueden leer el valor de cuenta de 39, y ambos escribirán el valor de 40, lo que supone incrementar de forma incorrecta el valor en 1, aunque realmente se hayan registrado efectivamente dos estudiantes y la cuenta debería ser de 41. Más aún, suponga que el límite de matriculación en el curso fuese de 40; en el caso anterior ambos estudiantes se habrían conseguido matricular violando el límite de los 40 estudiantes.
- **Problemas de seguridad.** Ningún usuario del sistema de la base de datos debería poder acceder a todos los datos. Por ejemplo, en una universidad, el personal del departamento financiero debería poder acceder exclusivamente a la parte de la base de datos con información financiera. No necesitan acceder a la información de registros académicos. Como los programas de aplicación se van añadiendo al sistema según se necesitan, resulta difícil forzar este tipo de restricciones de seguridad.

Estas dificultades, entre otras, han motivado el desarrollo de los sistemas de bases de datos. En el resto del libro se examinarán los conceptos y algoritmos que permiten que los sistemas de bases de datos resuelvan los problemas de los sistemas de procesamiento de archivos. En la mayor parte del libro se usa una universidad como ejemplo de aplicación típica de procesamiento de datos.

1.3. Visión de los datos

Un sistema de base de datos es una colección de datos interrelacionados y un conjunto de programas que permiten a los usuarios tener acceso a esos datos y modificarlos. Una de las principales finalidades de los sistemas de bases de datos es ofrecer a los usuarios una visión *abstracta* de los datos. Es decir, el sistema oculta ciertos detalles del modo en que se almacenan y mantienen los datos.

1.3.1. Abstracción de datos

Para que el sistema sea útil debe recuperar los datos eficientemente. La necesidad de eficiencia ha llevado a los diseñadores a usar estructuras de datos complejas para la representación de los datos en la base de datos. Dado que muchos de los usuarios de sistemas de bases de datos no tienen formación en informática, los desarrolladores ocultan esa complejidad a los usuarios mediante varios niveles de abstracción para simplificar la interacción de los usuarios con el sistema:

- **Nivel físico.** El nivel más bajo de abstracción describe *cómo* se almacenan realmente los datos. El nivel físico describe en detalle las estructuras de datos complejas de bajo nivel.
- **Nivel lógico.** El nivel inmediatamente superior de abstracción describe *qué* datos se almacenan en la base de datos y qué relaciones existen entre esos datos. El nivel lógico, por tanto, describe toda la base de datos en términos de un número pequeño de estructuras relativamente simples. Aunque la implementación de esas estructuras simples en el nivel lógico puede involucrar estructuras complejas del nivel físico, los usuarios del nivel lógico no necesitan preocuparse de esta complejidad. A esto se denomina **independencia de los datos físicos.** Los administradores de bases de datos, que deben decidir la información que se guarda en la base de datos, usan el nivel de abstracción lógico.
- **Nivel de vistas.** El nivel más elevado de abstracción solo describe parte de la base de datos. Aunque el nivel lógico usa estructuras más simples, queda cierta complejidad debido a la variedad de información almacenada en las grandes bases de datos. Muchos usuarios del sistema de bases de datos no necesitan toda esta información; en su lugar solo necesitan tener acceso a una parte de la base de datos. El nivel de abstracción de vistas existe para simplificar su interacción con el sistema. El sistema puede proporcionar muchas vistas para la misma base de datos.

En la Figura 1.1 se muestra la relación entre los tres niveles de abstracción.

Una analogía con el concepto de tipos de datos en lenguajes de programación puede clarificar la diferencia entre los niveles de abstracción. La mayoría de los lenguajes de programación de alto nivel soportan el concepto de tipo estructurado. Por ejemplo, en los lenguajes tipo Pascal se pueden declarar registros de la manera siguiente[1]:

```
type profesor = record
  ID: char (5);
  nombre: char (20);
  nombre_dept: char (20);
  sueldo: numeric (8,2);
end;
```

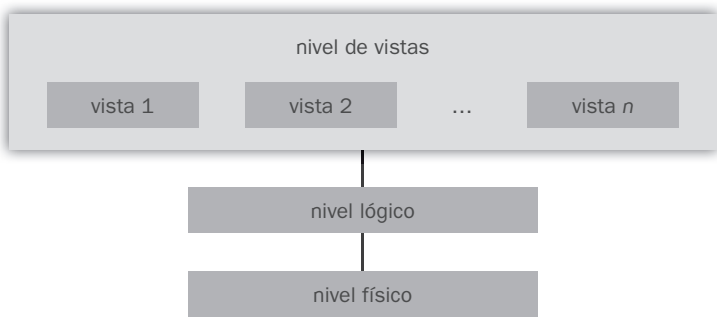


Figura 1.1. Los tres niveles de abstracción de datos.

1 La declaración de tipos real dependerá del lenguaje que se utilice. En C y C++ se utiliza la declaración **struct.** En Java no existe este tipo de declaración, sino que se obtiene el mismo efecto definiendo una clase simple.

Este código define un nuevo tipo de registro denominado *profesor*, con cuatro campos. Cada campo tiene un nombre y un tipo asociados. Una organización como una universidad puede tener muchos tipos de registros, entre otros:

- *departamento*, con los campos *nombre_dept, edificio* y *presupuesto*.
- *asignatura*, con los campos *id_asignatura, nombre_asig, nombre_dept* y *créditos*.
- *estudiante*, con los campos *nombre, nombre_dept* y *total_créditos*.

En el nivel físico, los registros profesor, departamento o estudiante se pueden describir como bloques de posiciones consecutivas de almacenamiento. El compilador oculta este nivel de detalle a los programadores. De manera parecida, el sistema de base de datos oculta muchos de los detalles de almacenamiento de los niveles inferiores a los programadores de bases de datos. Los administradores de bases de datos, por otro lado, pueden ser conscientes de ciertos detalles de la organización física de los datos.

En el nivel lógico, cada registro de este tipo se describe mediante una definición de tipo, como en el fragmento de código anterior, y también se define la relación entre estos tipos de registros. Los programadores que usan un lenguaje de programación trabajan en este nivel de abstracción. De manera parecida, los administradores de bases de datos suelen trabajar en este nivel de abstracción.

Finalmente, en el nivel de vistas, los usuarios ven un conjunto de programas de aplicación que ocultan los detalles de los tipos de datos. En el nivel de vistas se definen varias vistas de la base de datos, y los usuarios de estas pueden verlas todas o solamente una parte. Además de ocultar los detalles del nivel lógico de la base de datos, las vistas también proporcionan un mecanismo de seguridad para evitar que los usuarios tengan acceso a ciertas partes. Por ejemplo, los empleados de matriculación de la universidad solamente pueden ver la parte sobre información de los estudiantes, pero no pueden acceder a la información sobre sueldos ni profesores.

1.3.2. Ejemplares y esquemas

Las bases de datos van cambiando a lo largo del tiempo conforme la información se inserta y se elimina. La colección de información almacenada en la base de datos en un momento dado se denomina **ejemplar** de la base de datos. El diseño general de la base de datos se denomina **esquema** de la base de datos. Los esquemas se modifican rara vez, si es que se modifican.

El concepto de esquemas y ejemplares de las bases de datos se puede comprender por analogía con los programas escritos en un lenguaje de programación. El esquema de la base de datos se corresponde con las declaraciones de las variables (junto con las definiciones de tipos asociadas) de los programas. Cada variable tiene un valor concreto en un instante dado. Los valores de las variables de un programa se corresponden en cada momento con un *ejemplar* del esquema de la base de datos.

Los sistemas de bases de datos tienen varios esquemas, divididos según los niveles de abstracción. El **esquema físico** describe el diseño de la base de datos en el nivel físico, mientras que el **esquema lógico** describe su diseño en el nivel lógico. Las bases de datos también pueden tener varios esquemas en el nivel de vistas, a veces denominados **subesquemas,** que describen diferentes vistas de la base de datos.

De estos, el esquema lógico es con mucho el más importante en términos de su efecto sobre los programas de aplicación, ya que los programadores crean las aplicaciones usando el esquema lógico. El esquema físico está oculto bajo el esquema lógico, y generalmente puede modificarse fácilmente sin afectar a los programas de aplicación. Se dice que los programas de aplicación muestran **independencia física respecto de los datos** si no dependen del esquema físico y, por tanto, no hace falta volver a escribirlos si se modifica el esquema físico.

Se estudiarán los lenguajes para la descripción de los esquemas, después de introducir el concepto de modelos de datos en la siguiente sección.

1.3.3. Modelos de datos

Bajo la estructura de las bases de datos se encuentra el **modelo de datos:** se trata de una colección de herramientas conceptuales para describir los datos, sus relaciones, su semántica y las restricciones de consistencia. Los modelos de datos ofrecen un modo de describir el diseño de las bases de datos en los niveles físico, lógico y de vistas.

En este texto se van a tratar varios modelos de datos diferentes. Los modelos de datos pueden clasificarse en cuatro categorías diferentes:

- **Modelo relacional.** El modelo relacional utiliza una colección de tablas para representar tanto los datos como sus relaciones. Cada tabla tiene varias columnas, y cada columna tiene un nombre único. Las tablas también son conocidas como **relaciones.** 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. El modelo de datos relacional es el modelo de datos más ampliamente utilizado, y una gran mayoría de sistemas de bases de datos actuales se basan en el modelo relacional. Los Capítulos 2 al 8 tratan el modelo relacional en detalle.
- **El modelo entidad-relación.** El modelo de datos entidad-relación (E-R) consiste en una colección de objetos básicos, denominados *entidades*, y de las *relaciones* entre ellos. Una entidad es una «cosa» u «objeto» del mundo real que es distinguible de otros objetos. El modelo entidad-relación se usa mucho en el diseño de bases de datos, y en el Capítulo 7 se examina detalladamente.
- **Modelo de datos basado en objetos.** La programación orientada a objetos, especialmente en Java, C++ o C#, se ha convertido en la metodología de desarrollo software dominante, lo que ha llevado al desarrollo de modelos de datos orientados a objetos. El modelo orientado a objetos se puede considerar como una extensión del modelo E-R con conceptos de encapsulación, métodos (funciones) e identidad de objetos. El modelo de datos relacional de objetos combina las características de los modelos de datos orientados a objetos y el modelo de datos relacional. En el Capítulo 22 se examina este modelo de datos.
- **Modelo de datos semiestructurados.** El modelo de datos semiestructurados permite la especificación de datos en el que los elementos de datos individuales del mismo tipo pueden tener diferentes conjuntos de atributos. Esto lo diferencia de los modelos mencionados anteriormente, en los que cada elemento de datos de un tipo particular debe tener el mismo conjunto de atributos. El **lenguaje de marcas extensible** (XML: *eXtensible Markup Language*) se emplea mucho para representar datos semiestructurados. Se estudia en el Capítulo 23.

El **modelo de datos de red** y el **modelo de datos jerárquico** precedieron cronológicamente al relacional. Estos modelos estuvieron íntimamente ligados a la implementación subyacente y complicaban la tarea del modelado de datos. En consecuencia, se usan muy poco hoy en día, excepto en el código de bases de datos antiguas que siguen estando en servicio en algunos lugares. Se describen brevemente en los Apéndices D y E para los lectores interesados.

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; 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.
- La inserción de información nueva en la base de datos.
- El borrado de la información de la base de datos.
- La modificación de la información almacenada en la base de datos.

Hay fundamentalmente dos tipos:

- Los **LMD procedimentales** necesitan que el usuario especifique *qué* datos se necesitan y *cómo* obtener esos datos.
- Los **LMD declarativos** (también conocidos como **LMD no procedimentales**) necesitan que el usuario especifique *qué* datos se necesitan *sin* que haga falta especificar cómo obtener esos datos.

Los LMD 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 mismos.

Una **consulta** es una instrucción que solicita recuperar información. La parte de los LMD 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 experimental. El lenguaje de consultas más ampliamente usado, SQL, se estudiará en los Capítulos 3, 4 y 5. También se estudiarán otros lenguajes de consultas en el Capítulo 6.

Los niveles de abstracción que se trataron en la Sección 1.3 no solo 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 el énfasis se pone 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 12 y 13) traduce las consultas LMD en secuencias de acciones en el nivel físico del sistema de bases de datos.

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, suponga que la universidad requiere que el saldo de una cuenta de un departamento nunca pueda ser un valor negativo. El LDD proporciona elementos para especificar tales restricciones. Los sistemas de bases de datos los 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 aparezca también para un determinado conjunto de atributos en otra relación (integridad referencial). Por ejemplo, el departamento asociado a cada asignatura debe existir realmente. De forma más precisa, el valor *nombre_dept* de un registro de *asignatura* tiene que aparecer en el atributo *nombre_dept* de algún registro de la relación *departamento*. 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, «todos los departamentos deben ofertar al menos cinco asignaturas cada semestre» 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 únicamente se permite 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 los 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 (sentencias) 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, solo accesible y actualizable por el propio sistema de bases de datos (no por 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. En el Capítulo 2 se presenta una introducción a los fundamentos del modelo relacional. La mayor parte de los sistemas de bases de datos relacionales comerciales emplean el lenguaje SQL, que se tratará con gran detalle en los Capítulos 3, 4 y 5. En el Capítulo 6 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 dos tablas: una muestra detalles de los profesores, la otra muestra los distintos departamentos.

La primera tabla, la tabla *profesor*, muestra, por ejemplo, que el profesor de nombre Einstein, cuyo *ID* es 22222, forma parte del departamento de Física y tiene un sueldo anual de 95.000 €. La segunda tabla, de *departamento*, muestra, por ejemplo, que el departamento de Biología está ubicado en el edificio Watson y tiene un presupuesto de 95.000 €. Por supuesto, una universidad real tendrá muchos más departamentos y profesores. En el texto se usarán tablas pequeñas para ilustrar los conceptos. Se encuentra disponible *online* un ejemplo mucho mayor del mismo esquema.

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.

ID	*nombre*	*nombre_dept*	*sueldo*
22222	Einstein	Física	95.000
12121	Wu	Finanzas	90.000
32343	El Said	Historia	60.000
45565	Kafz	Informática	75.000
98345	Kim	Electrónica	80.000
76766	Crick	Biología	72.000
10101	Srinivasan	Informática	65.000
58583	Califieri	Historia	62.000
83821	Brandt	Informática	92.000
15151	Mozart	Música	40.000
33456	Gold	Física	87.000
76543	Singh	Finanzas	80.000

(a) La tabla *profesor*.

nombre_dept	*edificio*	*presupuesto*
Informática	Taylor	100.000
Biología	Watson	90.000
Electrónica	Taylor	85.000
Música	Packard	80.000
Finanzas	Painter	120.000
Historia	Painter	50.000
Física	Watson	70.000

(b) La tabla *departamento*.

Figura 1.2. Un ejemplo de base de datos relacional.

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.

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 el *presupuesto* del departamento como atributo del registro *profesor*. Entonces, siempre que cambie el valor de un determinado presupuesto, por ejemplo el del departamento de Física, este cambio debe reflejarse en los registros de todos los profesores asociados con el departamento de Física. En el Capítulo 8 se tratará cómo distinguir los buenos diseños de esquema de los malos.

1.5.2. Lenguaje de manipulación de datos

El lenguaje de consultas de SQL no es procedimental. Usa como entrada varias tablas (posiblemente solo una) y devuelve siempre una única tabla. A continuación se ofrece un ejemplo de consulta SQL que halla el nombre de todos los profesores del departamento de Historia:

select *profesor.nombre*
from *profesor*
where *profesor.nombre_dept* = «Historia»

La consulta especifica que hay que recuperar *(select)* las filas de *(from)* la tabla *profesor* en las que *(where)* el *nombre_dept* es Historia, y que solo debe mostrarse el atributo *nombre* de esas filas. Más concretamente, el resultado de la ejecución de esta consulta es una tabla con una sola columna denominada *nombre* y un conjunto de filas, cada una de las cuales contiene el nombre de un profesor cuyo *nombre_dept* es Historia. Si la consulta se ejecuta sobre la tabla de la Figura 1.2, el resultado constará de dos filas, una con el nombre El Said y otra con el nombre Califieri.

Las consultas pueden requerir información de más de una tabla. Por ejemplo, la siguiente consulta busca todos los *ID* de profesor y el nombre del departamento de todos los profesores con un departamento cuyo presupuesto sea superior a 95.000 €.

select *profesor.ID*, *departamento.nombre_dept*
from *profesor*, *departamento*
where *profesor.nombre_dept* = *departamento.nombre_dept* **and** *departamento.presupuesto* > 95000;

Si la consulta anterior se ejecutase sobre las tablas de la Figura 1.2, el sistema encontraría que hay dos departamentos con un presupuesto superior a 95.000 €, Informática y Finanzas, y que hay cinco profesores en estos departamentos. Por tanto, el resultado consistiría en una tabla con dos columnas *(ID, nombre_dept)* y las cinco filas: (12121, Finanzas), (45565, Informática), (10101, Informática), (83821, Informática) y (76543, Finanzas).

1.5.3. Lenguaje de definición de datos

SQL proporciona un rico LDD que permite definir tablas, restricciones de integridad, aserciones, etc.

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

create table *departamento*
(*nombre_dept* **char** (20),
edificio **char** (15),
presupuesto **numeric** (12,2));

La ejecución de esta instrucción LDD crea la tabla *departamento* con tres columnas: *nombre_dept*, *edificio* y *presupuesto*, cada una de ellas con su tipo de datos específico asociado. Se tratarán los tipos de datos con más detalle en el Capítulo 3. Además, la instrucción LDD actualiza el diccionario de datos, que contiene metadatos (véase la Sección 1.4.2). El esquema de la tabla es un ejemplo de metadatos.

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

SQL no es tan potente como una máquina universal de Turing; es decir, hay algunos cálculos que se pueden conseguir mediante un lenguaje de programación general pero no se pueden obtener mediante consultas SQL. SQL tampoco permite acciones como la entrada de datos de usuario, mostrar datos en pantalla o la comunicación por la red. Estas operaciones se deben escribir en otro lenguaje *anfitrión,* como por ejemplo C, C++ o Java, capaces de albergar consultas SQL que acceden a los datos de la base de datos. Los **programas de aplicación** son programas que se usan para interactuar de esta manera con las bases de datos. Algunos ejemplos de un sistema universitario serían los programas que permiten a los estudiantes matricularse en un curso, generar horarios, calcular las notas medias de los estudiantes, generar las nóminas, etc.

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*) 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.

1.6. Diseño de bases de datos

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. Esas grandes cantidades de información no existen aisladas. Forman parte del funcionamiento de alguna empresa, cuyo producto final puede que sea la información obtenida de la base de datos o algún dispositivo o servicio para el que la base de datos solo desempeña un papel secundario.

El diseño de bases de datos implica principalmente el diseño de su esquema. El diseño de un entorno completo de aplicaciones para la base de datos que satisfaga las necesidades de la empresa que se está modelando exige prestar atención a un conjunto de aspectos más amplio. Este texto se centrará inicialmente en la escritura de las consultas a una base de datos y en el diseño de los esquemas de bases de datos. En el Capítulo 9 se estudia el proceso general de diseño de las aplicaciones.

1.6.1. Proceso de diseño

Los modelos de datos de alto nivel resultan útiles a los diseñadores de bases de datos al ofrecerles un marco conceptual en el que especificar, de manera sistemática, los requisitos de datos de los usuarios de las bases de datos y la manera en que se estructurará la base de datos para satisfacer esos requisitos. La fase inicial del diseño de las bases de datos, por tanto, consiste en caracterizar completamente los requisitos de datos de los hipotéticos usuarios de la base de datos. Los diseñadores de bases de datos deben interactuar ampliamente con los expertos y usuarios del dominio para llevar a cabo esta tarea. El resultado de esta fase es la especificación de los requisitos de los usuarios.

A continuación, el diseñador escoge un modelo de datos y, mediante la aplicación de los conceptos del modelo de datos elegido, traduce esos requisitos en un esquema conceptual de la base de datos. El esquema desarrollado en esta fase de **diseño conceptual** ofrece una visión general detallada de la empresa. El diseñador revisa el esquema para confirmar que todos los requisitos de datos se satisfacen realmente y no entran en conflicto entre sí. El diseñador también puede examinar el diseño para eliminar cualquier característica redundante. En este punto, la atención se centra en describir los datos y sus relaciones, más que en especificar los detalles del almacenamiento físico.

En términos del modelo relacional, el proceso de diseño conceptual implica decisiones sobre *qué* atributos se desea capturar en la base de datos y *cómo agruparlos* para formar las diferentes tablas. La parte del «qué» es, en esencia, una decisión conceptual, y no se seguirá estudiando en este texto. La parte del «cómo» es sobre todo, un problema informático. Hay dos vías principales para afrontar el problema. La primera supone usar el modelo entidad-relación (Sección 1.6.3); la otra es emplear un conjunto de algoritmos (denominados colectivamente como *normalización*) que toma como entrada el conjunto de todos los atributos y genera un conjunto de tablas (Sección 1.6.4).

Un esquema conceptual completamente desarrollado también indica los requisitos funcionales de la empresa. En la **especificación de requisitos funcionales,** los usuarios describen el tipo de operaciones (o transacciones) que se llevarán a cabo con los datos. Un ejemplo de estas operaciones es modificar o actualizar los datos, buscar y recuperar datos concretos y eliminar datos. En esta etapa del diseño conceptual, el diseñador puede revisar el esquema para asegurarse de que satisface los requisitos funcionales.

El proceso de pasar de un modelo de datos abstracto a la implementación de la base de datos continúa con dos fases de diseño finales. En la **fase de diseño lógico,** el diseñador relaciona el esquema conceptual de alto nivel con el modelo de implementación de datos del sistema de bases de datos que se va a usar. El diseñador usa el esquema de bases de datos específico para el sistema resultante en la **fase de diseño físico** posterior, en la que se especifican las características físicas de la base de datos. Entre esas características están la forma de organización de los archivos y las estructuras de almacenamiento interno; se estudian en el Capítulo 10.

1.6.2. Diseño de la base de datos para una universidad

Para ilustrar el proceso de diseño, vamos a ver cómo se puede diseñar una base de datos para una universidad. La especificación de requisitos inicial puede basarse en un conjunto de entrevistas con los usuarios de la base de datos y el propio análisis del diseñador sobre la organización. La descripción que se obtiene de esta fase de diseño sirve como punto inicial para la especificación de la estructura conceptual de la base de datos. Estas son las principales características de la universidad:

- La universidad está organizada en departamentos. Cada departamento estará identificado por un nombre único *(nombre_dept)*, estará ubicado en un *edificio* y tendrá un *presupuesto*.

- Cada departamento tiene su propia lista de las asignaturas que oferta. Cada asignatura tiene asociado un *asignatura_id*, *nombre_asig*, *nombre_dept*, y puede tener un conjunto de *prerrequisitos*.
- Los profesores estarán identificados por su propio *ID* único. Los profesores tienen un *nombre*, el departamento *(nombre_dept)* al que están asociados y un *sueldo*.
- Los estudiantes estarán identificados por su propio *ID* único. Los estudiantes tendrán un *nombre,* un departamento *(nombre_dept)* principal al que estarán asociados y un *tot_cred* (número total de créditos que el estudiante haya conseguido).
- La universidad mantiene una lista de las aulas, indicando el nombre del *edificio*, el *número_aula* y la *capacidad* del aula.
- La universidad mantiene una lista de las clases (secciones) que se enseñan. Cada sección se identifica por su *asignatura_id*, *secc_id*, *año* y *semestre,* y tiene asociado un *semestre*, *año*, *edificio*, *número_aula* y *franja_horaria_id* (las horas a las que se imparte la clase).
- El departamento tiene una lista de la formación asignada, especificando, para cada profesor, las secciones que enseña.
- La universidad tiene una lista de todos los estudiantes matriculados, especificando, para cada estudiante, las asignaturas y las secciones asociadas en las que se ha matriculado.

La base de datos de una universidad real sería mucho más compleja que la que se ha indicado. Sin embargo, este modelo simplificado nos permitirá entender las ideas conceptuales sin perdernos en los detalles de un diseño complejo.

1.6.3. El modelo entidad-relación

El modelo de datos entidad-relación (E-R) se basa en un conjunto de objetos básicos, denominados *entidades*, y las *relaciones* entre esos objetos. Una entidad es una «cosa» u «objeto» del mundo real que es distinguible de otros objetos. Por ejemplo, cada persona es una entidad, y los profesores pueden considerarse entidades.

Las entidades se describen en las bases de datos mediante un conjunto de **atributos.** Por ejemplo, los atributos *nombre_dept*, *edificio* y *presupuesto* pueden describir un departamento concreto de una universidad y constituyen atributos del conjunto de entidades *departamento*. Análogamente, los atributos *ID*, *nombre* y *sueldo* pueden describir una entidad *profesor*[2].

Se usa un atributo extra, *ID*, para identificar unívocamente a un profesor (dado que es posible que haya dos profesores con el mismo nombre y el mismo sueldo). Se debe asignar un identificador de profesor único a cada profesor. En los Estados Unidos, muchas empresas usan el número de la seguridad social de cada persona (un número único que el Gobierno de Estados Unidos asigna a cada persona) como identificador.

Una **relación** es una asociación entre varias entidades. Por ejemplo, la relación *miembro* asocia un profesor con su departamento. El conjunto de todas las entidades del mismo tipo y el conjunto de todas las relaciones del mismo tipo se denominan, respectivamente, **conjunto de entidades** y **conjunto de relaciones.**

La estructura lógica general (esquema) de la base de datos se puede expresar gráficamente mediante un *diagrama entidad-relación (E-R)*. Existen distintas formas de dibujar este tipo de diagramas. Una de las más populares es utilizar el **lenguaje de modelado unificado** (UML: *unified modeling language*). En la notación que usaremos, basada en UML, un diagrama E-R se representa de la siguiente forma:

- Las entidades se representan mediante un rectángulo con el nombre de la entidad en la cabecera y la lista de atributos debajo.
- Las relaciones se representan mediante un rombo que conecta un par de entidades. El nombre de la relación se pone dentro del rombo.

Como ejemplo, suponga parte de la base de datos de una universidad, que consiste en profesores y departamentos con los que están asociados. En la Figura 1.3 se muestra el diagrama E-R correspondiente. Este diagrama indica que existen dos conjuntos de entidades, profesor y departamento, con los atributos indicados como se ha comentado anteriormente. El diagrama también muestra una relación entre profesor y departamento.

Además de entidades y relaciones, el modelo E-R representa ciertas restricciones que los contenidos de la base de datos deben cumplir. Una restricción importante es la **correspondencia de cardinalidades,** que expresa el número de entidades con las que otra entidad se puede asociar a través de un conjunto de relaciones. Por ejemplo, si un instructor solo se puede asociar con un único departamento, el modelo E-R puede expresar esta restricción.

El modelo entidad-relación se usa ampliamente en el diseño de bases de datos, y en el Capítulo 7 se explora en detalle.

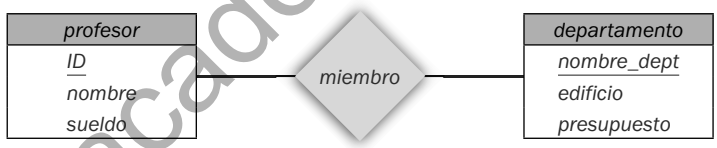


Figura 1.3. Diagrama E-R simple.

1.6.4. Normalización

Otro método de diseño de bases de datos es usar un proceso que suele denominarse normalización. El objetivo es generar un conjunto de esquemas de relaciones que permita almacenar información sin redundancias innecesarias, pero que también permita recuperar la información con facilidad. El enfoque es diseñar esquemas que se hallen en la *forma normal* adecuada. Para determinar si un esquema de relación se halla en una de las formas normales deseadas, hace falta información adicional sobre la empresa real que se está modelando con la base de datos. El enfoque más frecuente es usar **dependencias funcionales,** que se tratan en la Sección 8.4.

Para comprender la necesidad de la normalización, obsérvese lo que puede fallar en un mal diseño de base de datos. Algunas de las propiedades no deseables de un mal diseño pueden ser:

- Repetición de la información.
- Imposibilidad de representar determinada información.

Se examinarán estos problemas con la ayuda de un diseño de bases de datos modificado para el ejemplo de nuestra universidad.

Suponga que, en lugar de tener las dos tablas, *profesor* y *departamento*, separadas, se tiene una sola tabla, *facultad*, que combina la información de las dos tablas (como puede verse en la Figura 1.4). Observe que hay dos filas de *facultad* que contienen información repetida del departamento de Historia, específicamente el edificio y el presupuesto. La repetición de la información en nuestro diseño alternativo no es deseable. Repetir la información desaprovecha espacio. Más aún, complica la actualización de la base de datos. Suponga que se desea modificar el presupuesto del departamento de Historia de 50.000 € a 46.800 €. Este cambio debe quedar reflejado en las dos filas; a diferencia del diseño original donde solo se necesita actualizar una única fila. Por tanto, las actualizaciones son más costosas en este diseño alternativo que en el original. Cuando se realiza una actualización en la base de datos alternativa, debemos asegurarnos de que se actualizan todas las filas que pertenecen al departamento de Historia. En caso contrario nuestra base de datos mostrará dos valores diferentes de presupuesto para dicho departamento.

2 El lector avispado se habrá dado cuenta que se ha eliminado el atributo *nombre_dept* del conjunto de atributos que describen la entidad *profesor*; no se trata de un error. En el Capítulo 7 se verá con detalle una explicación de por qué.

ID	*nombre*	*sueldo*	*nombre_dept*	*edificio*	*presupuesto*
22222	Einstein	95.000	Física	Watson	70.000
12121	Wu	90.000	Finanzas	Painter	120.000
32343	El Said	60.000	Historia	Painter	50.000
45565	Katz	75.000	Informática	Taylor	100.000
98345	Kim	80.000	Electrónica	Taylor	85.000
76766	Crick	72.000	Biología	Watson	90.000
10101	Srinivasan	65.000	Informática	Taylor	100.000
58583	Califieri	62.000	Historia	Painter	50.000
83821	Brandt	92.000	Informática	Taylor	100.000
15151	Mozart	40.000	Música	Packard	80.000
33456	Gold	87.000	Física	Watson	70.000
76543	Singh	80.000	Finanzas	Painter	120.000

Figura 1.4. La tabla *facultad.*

Examínese ahora el problema de la «imposibilidad de representar determinada información». Suponga que se desea crear un nuevo departamento en la universidad. En el diseño alternativo anterior no se puede representar directamente la información de un departamento (*nombre_dept*, *edificio*, *presupuesto*) a no ser que el departamento tenga al menos un profesor. Se debe a que en la tabla *facultad* se requiere que existan los valores *ID, nombre* y *sueldo*. Esto significa que no se puede registrar la información del nuevo departamento creado hasta haber contratado al primer profesor para el departamento.

Una solución para este problema es introducir valores **nulos.** Los valores *nulos* indican que el valor no existe (o es desconocido). Los valores desconocidos pueden ser valores *ausentes* (el valor existe, pero no se tiene la información) o valores *desconocidos* (no se sabe si el valor existe realmente o no). Como se verá más adelante, los valores nulos resultan difíciles de tratar, y es preferible no recurrir a ellos. Si no se desea tratar con valores nulos, se puede crear la información del departamento solo cuando se contrate al menos a un profesor en dicho departamento, Además, habría que eliminar esa información cuando el último profesor abandone el departamento. Claramente, esta situación no es deseable ya que, bajo el diseño original de la base de datos, la información de los departamentos debería existir haya o no profesores asociados, sin necesidad de recurrir a valores nulos.

Existe una extensa teoría de normalización que ayuda a definir formalmente qué bases de datos no son deseables y cómo obtener diseños apropiados. En el Capítulo 8 se trata el diseño de bases de datos relacionales, incluyendo el proceso de normalización.

1.7. 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, realmente 1.024 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. El procesador de consultas permite que los usuarios de la base de datos consigan un buen rendimiento a la vez que pueden trabajar en el nivel de vistas sin necesidad de comprender los detalles del diseño físico de la implementación del sistema. 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.

1.7.1. 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 sin 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, así como de 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 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 *profesor* con un *ID* determinado, o todos los registros *profesor* con un *nombre* dado. La asociación (*hashing*) 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 10. Los métodos de acceso eficiente a los datos mediante indexación o asociación se explican en el Capítulo 11.

1.7.2. 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 una **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 12, mientras que los métodos por los que el optimizador de consultas elige entre las estrategias de evaluación posibles se tratan en el Capítulo 13.

1.8. Gestión de transacciones

A menudo, varias operaciones sobre la base de datos forman una única unidad lógica de trabajo. Un ejemplo son las transferencias de fondos, como se vio en la Sección 1.2, en las que se realiza un cargo en la cuenta de un departamento (llámese *A*) y un abono en otra cuenta de otro departamento (llámese *B*). Evidentemente, resulta fundamental que, o bien tengan lugar tanto el cargo como el abono, o bien no se produzca ninguno. Es decir, la transferencia de fondos debe tener lugar por completo o no producirse en absoluto. Este requisito de todo o nada se denomina **atomicidad.** Además, resulta esencial que la ejecución de la transferencia de fondos preserve la consistencia de la base de datos. Es decir, el valor de la suma de los saldos de *A* y *B* se debe preservar. Este requisito de corrección se denomina **consistencia.** Finalmente, tras la ejecución correcta de la transferencia de fondos, los nuevos valores de las cuentas *A* y *B* deben persistir, a pesar de la posibilidad de un fallo del sistema. Este requisito de persistencia se denomina **durabilidad.**

Una **transacción** es un conjunto de operaciones que lleva a cabo una única función lógica en una aplicación de bases de datos. Cada transacción es una unidad de atomicidad y consistencia. Por tanto, se exige que las transacciones no violen ninguna restricción de consistencia de la base de datos. Es decir, si la base de datos era consistente cuando la transacción comenzó, debe ser consistente cuando la transacción termine con éxito. Sin embargo, durante la ejecución de una transacción puede ser necesario permitir inconsistencias temporalmente, ya que hay que hacer en primer lugar el cargo a *A* o el abono a *B*. Esta inconsistencia temporal, aunque necesaria, puede conducir a dificultades si se produce un fallo.

Es responsabilidad del programador definir adecuadamente las diferentes transacciones, de tal manera que cada una preserve la consistencia de la base de datos. Por ejemplo, la transacción para transferir fondos de la cuenta de un departamento *A* a la cuenta de un departamento *B* puede definirse como si estuviera compuesta de dos programas diferentes: uno que realiza el cargo en la cuenta *A* y otro que realiza el abono en la cuenta *B*. La ejecución de estos dos programas, uno después del otro, preservará realmente la consistencia. Sin embargo, cada programa en sí mismo no transforma la base de datos de un estado consistente a otro nuevo. Por tanto, estos programas no son transacciones.

Garantizar las propiedades de atomicidad y de durabilidad es responsabilidad del propio sistema de bases de datos; concretamente, del **componente de gestión de transacciones.** Si no existen fallos, todas las transacciones se completan con éxito y la atomicidad se consigue fácilmente. Sin embargo, debido a diversos tipos de fallos, puede que las transacciones no siempre completen su ejecución con éxito. Si se va a asegurar la propiedad de atomicidad, las transacciones fallidas no deben tener ningún efecto sobre el estado de la base de datos. Por tanto, esta debe restaurarse al estado en que estaba antes de que la transacción en cuestión comenzó a ejecutarse. El sistema de bases de datos, por tanto, debe realizar la **recuperación de fallos,** es decir, detectar los fallos del sistema y restaurar la base de datos al estado que tenía antes de que ocurriera el fallo.

Finalmente, cuando varias transacciones actualizan la base de datos de manera concurrente, puede que no se preserve la consistencia de los datos, aunque cada una de las transacciones sea correcta. Es responsabilidad del **gestor de control de concurrencia** controlar la interacción entre las transacciones concurrentes para garantizar la consistencia de la base de datos. El **gestor de transacciones** consta del gestor de control de concurrencia y del gestor de recuperación de fallos.

Los conceptos básicos del procesamiento de transacciones se tratan en el Capítulo 14. La gestión de las transacciones concurrentes se trata en el Capítulo 15. En el Capítulo 16 se estudiará con detalle la recuperación de fallos.

El concepto de transacción se ha manejado ampliamente en los sistemas de bases de datos y en sus aplicaciones. Aunque el empleo inicial de las transacciones se produjo en las aplicaciones financieras, el concepto se usa ahora en aplicaciones en tiempo real de telecomunicaciones, así como en la gestión de las actividades de larga duración, como el diseño de productos o los flujos de trabajo administrativo. Estas aplicaciones más amplias del concepto de transacción se estudian en el Capítulo 26.

1.9. Arquitectura de las bases de datos

Ya podemos ofrecer una visión única (Figura 1.5) de los diversos componentes de los sistemas de bases de datos y de las conexiones existentes entre ellos.

La arquitectura de los sistemas de bases de datos se ve muy influida por el sistema informático subyacente, sobre el que se ejecuta el sistema de bases de datos. Los sistemas de bases de datos pueden estar centralizados o ser del tipo cliente-servidor, en los que una máquina servidora ejecuta el trabajo en nombre de multitud de máquinas clientes. Los sistemas de bases de datos pueden dise-

ñarse también para aprovechar las arquitecturas de computadoras paralelas. Las bases de datos distribuidas se extienden por varias máquinas geográficamente separadas.

En el Capítulo 17 se trata la arquitectura general de los sistemas informáticos modernos. El Capítulo 18 describe el modo en que diversas acciones de las bases de datos, en especial el procesamiento de las consultas, pueden implementarse para aprovechar el procesamiento paralelo. El Capítulo 19 presenta varios problemas que surgen en las bases de datos distribuidas y describe el modo de afrontarlos. Entre los problemas se encuentran el modo de almacenar los datos, la manera de asegurar la atomicidad de las transacciones que se ejecutan en varios sitios, cómo llevar a cabo controles de concurrencia y el modo de ofrecer alta disponibilidad en presencia de fallos. El procesamiento distribuido de las consultas y los sistemas de directorio también se describen en ese capítulo.

Hoy en día la mayor parte de los usuarios de los sistemas de bases de datos no está presente en el lugar físico en que se encuentra el sistema de base de datos, sino que se conectan a él a través de una red. Por tanto, se puede diferenciar entre los equipos **clientes,** en los que trabajan los usuarios remotos de la base de datos, y los equipos **servidores,** en los que se ejecutan los sistemas de bases de datos.

Las aplicaciones de bases de datos suelen dividirse en dos o tres partes, como puede verse en la Figura 1.6. En una **arquitectura de dos capas,** la aplicación se divide en un componente que reside en la máquina cliente, que invoca la funcionalidad del sistema de bases de datos en la máquina servidora mediante instrucciones del lenguaje de consultas. Los estándares de interfaces de programas de aplicación, como ODBC y JDBC, se usan para la interacción entre el cliente y el servidor.

En cambio, en una **arquitectura de tres capas,** la máquina cliente actúa simplemente como una parte visible al usuario y no alberga llamadas directas a la base de datos. En vez de eso, el extremo cliente se comunica con un **servidor de aplicaciones,** generalmente mediante una interfaz de formularios. El servidor de aplicaciones, a su vez, se comunica con el sistema de bases de datos para tener acceso a los datos. La **lógica de negocio** de la aplicación, que establece las acciones que se deben realizar según las condiciones reinantes, se incorpora en el servidor de aplicaciones, en lugar de estar distribuida entre múltiples clientes. Las aplicaciones de tres capas resultan más adecuadas para aplicaciones de gran tamaño y para las aplicaciones que se ejecutan en la *World Wide Web*.

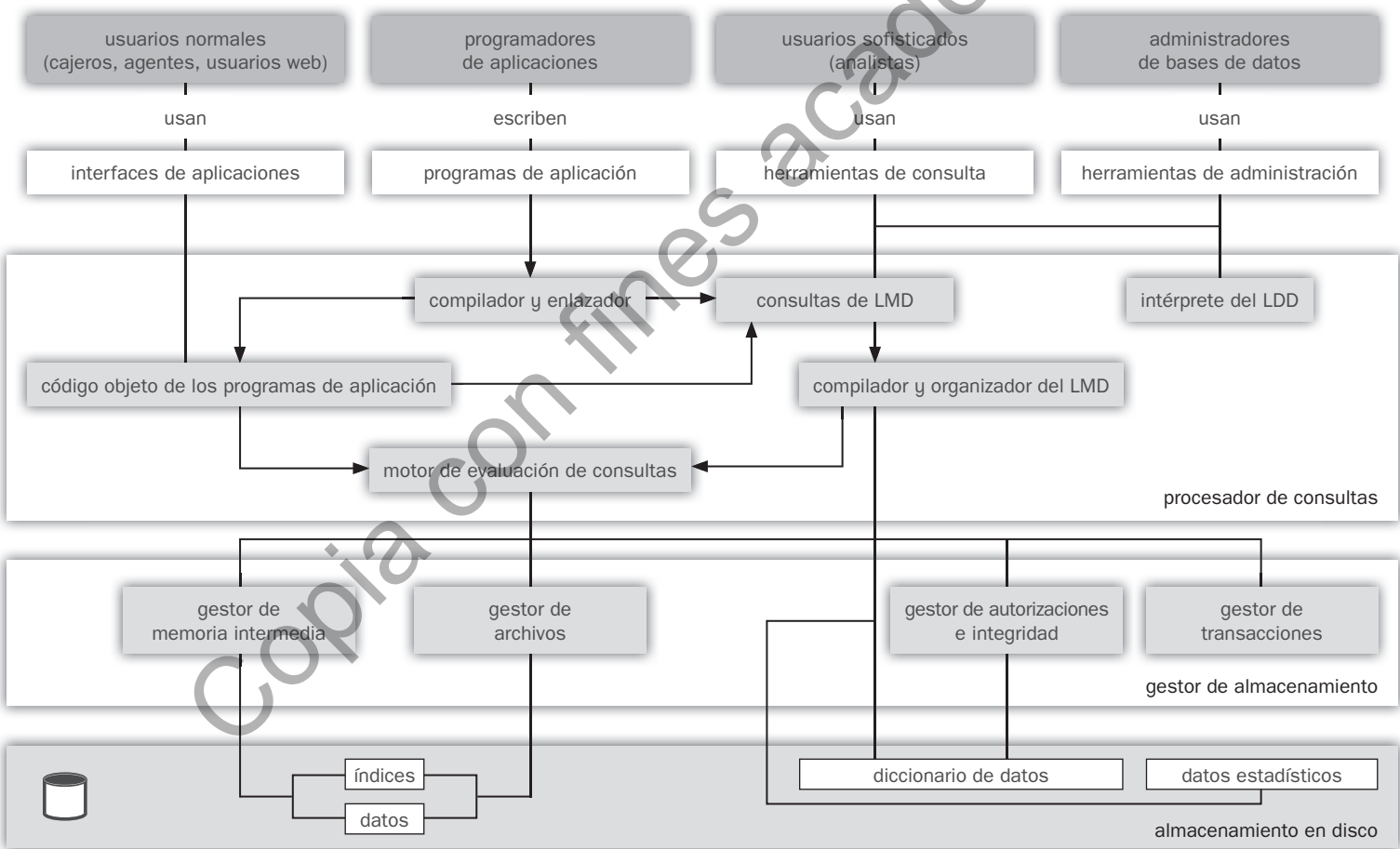
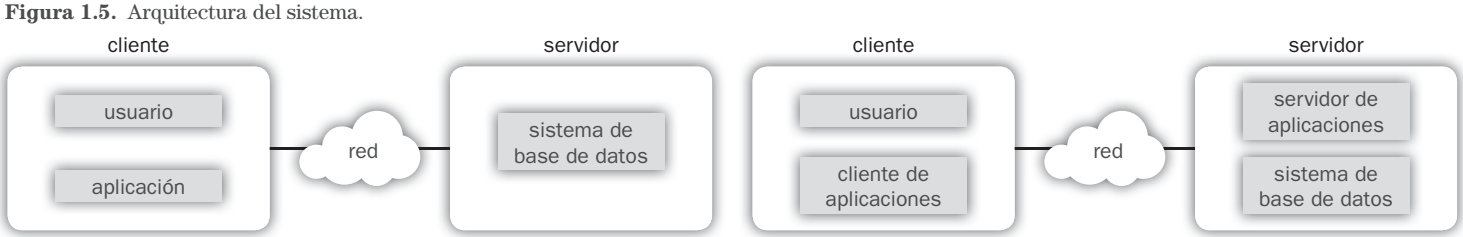


Figura 1.5. Arquitectura del sistema.



(a) Arquitectura de dos capas

(b) Arquitectura de tres capas

Figura 1.6. Arquitecturas de dos y tres capas.

1.10. Minería y análisis de datos

El término **minería de datos** se refiere, en líneas generales, al proceso de análisis semiautomático de grandes bases de datos para descubrir patrones útiles. Al igual que el descubrimiento de conocimientos en la inteligencia artificial (también denominado **aprendizaje de la máquina**) o el análisis estadístico, la minería de datos intenta descubrir reglas y patrones en los datos. Sin embargo, la minería de datos se diferencia del aprendizaje de la máquina y de la estadística en que maneja grandes volúmenes de datos, almacenados principalmente en disco. Es decir, la minería de datos trata del «descubrimiento de conocimientos en las bases de datos».

Algunos tipos de conocimientos descubiertos en las bases de datos pueden representarse mediante un conjunto de **reglas.** Lo que sigue es un ejemplo de regla, definida informalmente: «las mujeres jóvenes con ingresos anuales superiores a 50.000 € son las personas con más probabilidades de comprar coches deportivos pequeños». Por supuesto, esas reglas no son universalmente ciertas, sino que tienen grados de «apoyo» y de «confianza». Otros tipos de conocimiento se representan mediante ecuaciones que relacionan diferentes variables, o mediante otros mecanismos para la predicción de los resultados cuando se conocen los valores de algunas variables.

Una gran variedad de tipos posibles de patrones pueden resultar útiles, y para descubrir tipos de patrones diferentes se emplean diversas técnicas. En el Capítulo 20 se estudian unos cuantos ejemplos de patrones y se ve la manera en que pueden obtenerse de las bases de datos de forma automática.

Generalmente, en la minería de datos hay un componente manual, que consiste en el preprocesamiento de los datos de una manera aceptable para los algoritmos, y el posprocesamiento de los patrones descubiertos para descubrir otros nuevos que puedan resultar útiles. Lo normal es que más de un tipo de patrón pueda ser descubierto en una base de datos dada, y quizá sea necesaria la interacción manual para escoger los tipos de patrones útiles. Por este motivo, la minería de datos en la vida real es, en realidad, un proceso semiautomático. No obstante, en nuestra descripción se concentra la atención en el aspecto automático de la minería.

Las empresas han comenzado a explotar la creciente cantidad de datos en línea para tomar mejores decisiones sobre sus actividades, como los artículos de los que hay que tener existencias y la mejor manera de llegar a los clientes para incrementar las ventas. Muchas de sus consultas son, sin embargo, bastante complicadas, y hay cierto tipo de información que no puede extraerse ni siquiera usando SQL.

Se dispone de varias técnicas y herramientas para ayudar a la toma de decisiones. Algunas herramientas para el análisis de datos permiten a los analistas ver los datos de diferentes maneras. Otras herramientas de análisis realizan cálculos previos de resúmenes de grandes cantidades de datos, con objeto de dar respuestas rápidas a las preguntas. El estándar SQL contiene actualmente constructores adicionales para soportar el análisis de datos.

Las grandes compañías disponen de distintas fuentes de datos que necesitan utilizar para tomar decisiones empresariales. Si desean ejecutar consultas eficientes sobre datos tan diversos, las compañías han de construir *almacenes de datos*. Los almacenes de datos recogen datos de distintas fuentes bajo un esquema unificado, en un único punto. Por tanto, proporcionan al usuario una interfaz única y uniforme con los datos.

Los datos textuales también han crecido de manera explosiva. Estos datos carecen de estructura, a diferencia de los datos rígidamente estructurados de las bases de datos relacionales. La consulta de datos textuales no estructurados se denomina *recuperación de la información*. Los sistemas de recuperación de la información tienen mucho en común con los sistemas de bases de datos; en especial, el almacenamiento y la recuperación de datos en medios de almacenamiento secundarios. Sin embargo, el énfasis en el campo de los sistemas de información es diferente del de los sistemas de bases de datos, y se concentra en aspectos como las consultas basadas en palabras clave; la relevancia de los documentos para la consulta; y el análisis, clasificación e indexación de los documentos. En los Capítulos 20 y 21 se trata la ayuda a la toma de decisiones, incluyendo el procesamiento analítico en línea, la minería de datos, los almacenes de datos y la recuperación de la información.

1.11. Bases de datos específicas

Varias áreas de aplicaciones de los sistemas de bases de datos están limitadas por las restricciones del modelo de datos relacional. En consecuencia, los investigadores han desarrollado varios modelos de datos para tratar con estos dominios de aplicación, entre otros los modelos basados en objetos y los modelos de datos semiestructurados.

1.11.1. Modelos de datos basados en objetos

El modelo de datos orientado a objetos se ha convertido en la metodología dominante de desarrollo de software. Ello ha llevado al desarrollo del **modelo de datos orientado a objetos**, que se puede ver como una ampliación del modelo E-R con las nociones de encapsulación, métodos (funciones) e identidad de objetos. La herencia, la identidad de objetos y la encapsulación (ocultamiento de información), junto con otros métodos que aportan una interfaz para con los objetos, se encuentran entre los conceptos clave de la programación orientada a objetos que han encontrado aplicación en el modelado de datos. El modelo de datos orientado a objetos también enriquece el sistema de tipos, incluyendo tipos estructurados y colecciones. En los años ochenta, se desarrollaron varios sistemas de bases de datos siguiendo un modelo de datos orientado a objetos.

Los principales proveedores de bases de datos apoyan actualmente el **modelo de datos relacional orientado a objetos** que combina las características del modelo de datos orientado a objetos con el modelo de datos relacional. Amplía el modelo relacional tradicional con distintas características como los tipos estructurados y las colecciones, así como la orientación a objetos. El Capítulo 22 trata el modelo de datos relacional orientado a objetos.

1.11.2. Modelos de datos semiestructurados

Los modelos de datos semiestructurados permiten la especificación de los datos, de modo que cada elemento de datos del mismo tipo puede tener conjuntos de atributos diferentes. Esto los diferencia de los modelos de datos mencionados anteriormente, en los que todos los elementos de datos de un tipo dado deben tener el mismo conjunto de atributos.

El lenguaje XML se diseñó inicialmente como un modo de añadir información de marcas a los documentos de texto, pero se ha vuelto importante debido a sus aplicaciones en el intercambio de datos. XML ofrece un modo de representar los datos que tienen una estructura anidada y, además, permite una gran flexibilidad en la estructuración de los datos, lo cual es importante para ciertas clases de datos no tradicionales. En el Capítulo 23 se describe el lenguaje XML, diferentes maneras de expresar las consultas sobre datos representados en XML y la transformación de los datos XML de una forma a otra.

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 diversos 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 empleado de la universidad que tiene que añadir un nuevo profesor a un departamento A invoca un programa llamado *nueva_contratacion*. Este programa le pide al empleado el nombre del nuevo profesor, su nuevo *ID*, el nombre del departamento (es decir, A) y el sueldo.

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

  Como ejemplo adicional, considérese un estudiante que durante el periodo de matrícula quiere matricularse en una asignatura utilizando una interfaz web. Ese usuario puede acceder a una aplicación web que se ejecuta en un servidor web. La aplicación comprueba en primer lugar la identidad del usuario y le permite acceder a un formulario en el que introduce la información solicitada. La información del formulario se envía de vuelta a la aplicación web en el servidor, se comprueba si hay espacio en la asignatura (realizando una consulta a la base de datos) y si lo hay, añade la información del estudiante a la asignatura en 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 o con herramientas como el software de análisis de datos. 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 22 se estudian varias de estas aplicaciones.

1.12.2. Administrador de bases de datos

Una de las principales razones de usar un SGBD 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 de un administrador de bases de datos son:
  - Copia de seguridad periódica de la base de datos, bien sobre cinta o bien sobre servidores remotos, para impedir la pérdida de datos en casos de desastre como inundaciones.
  - Asegurarse de que se dispone de suficiente espacio libre en disco para las operaciones normales y aumentar el espacio en el 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.

1.13. Historia de los sistemas de bases de datos

El procesamiento de datos impulsa el crecimiento de las computadoras, como lo ha hecho desde los primeros días de las computadoras comerciales. De hecho, la automatización de las tareas de procesamiento de datos precede a las computadoras. Las tarjetas perforadas, inventadas por Herman Hollerith, se emplearon a principios del siglo XX para registrar los datos del censo de Estados Unidos, y se usaron sistemas mecánicos para procesar las tarjetas y para tabular los resultados. Las tarjetas perforadas se utilizaron posteriormente con profusión como medio para introducir datos en las computadoras.

Las técnicas de almacenamiento y de procesamiento de datos han evolucionado a lo largo de los años:

- **Años cincuenta y primeros años sesenta:** se desarrollaron las cintas magnéticas para el almacenamiento de datos. Las tareas de procesamiento de datos, como la elaboración de nóminas, se automatizaron con los datos almacenados en cintas. El procesamiento de datos consistía en leer datos de una o varias cintas y escribirlos en una nueva cinta. Los datos también se podían introducir desde paquetes de tarjetas perforadas e imprimirse en impresoras. Por ejemplo, los aumentos de sueldo se procesaban introduciéndolos en las tarjetas perforadas y leyendo el paquete de cintas perforadas de manera sincronizada con una cinta que contenía los detalles principales de los sueldos.

Los registros debían estar en el mismo orden. Los aumentos de sueldo se añadían a los sueldos leídos de la cinta maestra y se escribían en una nueva cinta; esa nueva cinta se convertía en la nueva cinta maestra.

Las cintas (y los paquetes de tarjetas perforadas) solo se podían leer secuencialmente, y el tamaño de los datos era mucho mayor que la memoria principal; por tanto, los programas de procesamiento de datos se veían obligados a procesar los datos en un orden determinado, leyendo y mezclando datos de las cintas y de los paquetes de tarjetas perforadas.

- **Finales de los años sesenta y años setenta:** el empleo generalizado de los discos duros a finales de los años sesenta modificó en gran medida la situación del procesamiento de datos, ya que permitieron el acceso directo a los datos. La ubicación de los datos en disco no era importante, ya que se podía tener acceso a cualquier posición del disco en solo unas decenas de milisegundos. Los datos se liberaron así de la tiranía de la secuencialidad. Con los discos pudieron crearse las bases de datos de red y las bases de datos jerárquicas, que permitieron que las estructuras de datos, como las listas y los árboles, pudieran almacenarse en disco. Los programadores pudieron crear y manipular estas estructuras de datos.

  El artículo histórico de Codd [1970] definió el modelo relacional y las formas no procedimentales de consultar los datos en el modelo relacional, y así nacieron las bases de datos relacionales. La simplicidad del modelo relacional y la posibilidad de ocultar completamente los detalles de implementación a los programadores resultaron realmente atractivas. Codd obtuvo posteriormente el prestigioso premio Turing de la ACM (*Association of Computing Machinery*: asociación de maquinaria informática) por su trabajo.

- **Años ochenta:** aunque académicamente interesante, el modelo relacional no se usó inicialmente en la práctica debido a sus inconvenientes en cuanto a rendimiento; las bases de datos relacionales no podían igualar el rendimiento de las bases de datos de red y jerárquicas existentes. Esta situación cambió con System R, un proyecto innovador del centro de investigación de IBM que desarrolló técnicas para la construcción de un sistema de bases de datos relacionales eficiente. En Astrahan et ál. [1976] y Chamberlin et ál. [1981] se pueden encontrar excelentes visiones generales de System R. El prototipo System R completamente funcional condujo al primer producto de bases de datos relacionales de IBM: SQL/DS. Al mismo tiempo, se estaba desarrollando el sistema Ingres en la Universidad de California, lo que condujo a un producto comercial del mismo nombre. Los primeros sistemas comerciales de bases de datos relacionales, como DB2 de IBM, Oracle, Ingres y Rdb de DEC, desempeñaron un importante papel en el desarrollo de técnicas para el procesamiento eficiente de las consultas declarativas. En los primeros años ochenta las bases de datos relacionales habían llegado a ser competitivas frente a los sistemas de bases de datos jerárquicas y de red, incluso en cuanto a rendimiento. Las bases de datos relacionales eran tan sencillas de usar que finalmente reemplazaron a las bases de datos jerárquicas y de red; los programadores que usaban esas bases de datos se veían obligados a tratar muchos detalles de implementación de bajo nivel y tenían que codificar sus consultas de forma procedimental, y lo que era aún más importante, tenían que tener presente el rendimiento durante el diseño de los programas, lo que suponía un gran esfuerzo. En cambio, en las bases de datos relacionales, casi todas estas tareas de bajo nivel las realiza de manera automática el sistema de bases de datos, lo que libera al programador para que se centre en el nivel lógico. Desde que alcanzara su liderazgo en los años ochenta, el modelo relacional ha reinado sin discusión entre todos los modelos de datos.

  Los años ochenta también fueron testigos de profunda investigación en las bases de datos paralelas y distribuidas, así como del trabajo inicial en las bases de datos orientadas a objetos.

- **Primeros años noventa:** el lenguaje SQL se diseñó fundamentalmente para las aplicaciones de ayuda a la toma de decisiones, que son intensivas en consultas, mientras que el objetivo principal de las bases de datos en los años ochenta eran las aplicaciones de procesamiento de transacciones, que son intensivas en actualizaciones. La ayuda a la toma de decisiones y las consultas volvieron a emerger como una importante área de aplicación para las bases de datos. El uso de las herramientas para analizar grandes cantidades de datos experimentó un gran crecimiento.

  En esta época muchas marcas de bases de datos introdujeron productos de bases de datos paralelas. Las diferentes marcas de bases de datos comenzaron también a añadir soporte relacional orientado a objetos a sus bases de datos.

- **Finales de los años noventa:** el principal acontecimiento fue el crecimiento explosivo de la *World Wide Web*. Las bases de datos se implantaron mucho más ampliamente que nunca hasta ese momento. Los sistemas de bases de datos tenían que soportar tasas de procesamiento de transacciones muy elevadas, así como una fiabilidad muy alta, y disponibilidad 24×7 (disponibilidad 24 horas al día y 7 días a la semana, lo que impedía momentos de inactividad debidos a actividades de mantenimiento planificadas). Los sistemas de bases de datos también tenían que soportar interfaces web para los datos.

- **Años 2000:** la primera mitad de los años 2000 ha sido testigo del crecimiento de XML y de su lenguaje de consultas asociado, XQuery, como nueva tecnología de las bases de datos. Aunque XML se emplea ampliamente para el intercambio de datos, así como para el almacenamiento de tipos de datos complejos, las bases de datos relacionales siguen siendo el núcleo de la inmensa mayoría de las aplicaciones de bases de datos muy grandes. En este periodo también se ha podido presenciar el crecimiento de las técnicas de «informática autónoma/administración automática» para la minimización del esfuerzo de administración. Este periodo también ha vivido un gran crecimiento del uso de sistemas de bases de datos de fuente abierta, en particular PostgreSQL y MySQL.

  En la última parte de la década se ha visto un crecimiento de las bases de datos especializadas en el análisis de datos, en particular el almacén de columnas, que guardan realmente cada columna de una tabla como *array* (matriz) separado, y sistemas de bases de datos masivamente paralelos diseñados para el análisis de grandes conjuntos de datos. Se han construido varios sistemas de almacenamiento distribuido novedosos para el manejo de los requisitos de gestión de datos de sitios web de grandes dimensiones, como Amazon, Facebook, Google, Microsoft y Yahoo!, y algunos de ellos se ofertan como servicios web que pueden utilizar desarrolladores de aplicaciones. También se ha realizado un importante trabajo en la gestión y análisis de flujos de datos, como los datos del mercado de valores o los datos de gestión de las redes de computadores. Se han desarrollado ampliamente técnicas de minería de datos; entre las aplicaciones de ejemplo se encuentran los sistemas de recomendación de productos para la web y la selección automática de anuncios relevantes en páginas web.

1.14. Resumen

- Un **sistema gestor de bases de datos (SGBD)** consiste en un conjunto de datos interrelacionados y un conjunto de programas para tener acceso a esos datos. Los datos describen una empresa concreta.
- El objetivo principal de un SGBD es proporcionar un entorno que sea tanto conveniente como eficiente para las personas que lo usan, para la recuperación y almacenamiento de información.
- Los sistemas de bases de datos resultan ubicuos hoy en día, y la mayor parte de la gente interactúa, directa o indirectamente, con bases de datos muchas veces al día.
- Los sistemas de bases de datos se diseñan para almacenar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para el almacenamiento de la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben preocuparse de la seguridad de la información almacenada, en caso de caídas del sistema o de intentos de acceso sin autorización. Si los datos deben compartirse entre varios usuarios, el sistema debe evitar posibles resultados anómalos.
- Uno de los propósitos principales de los sistemas de bases de datos es ofrecer a los usuarios una visión abstracta de los datos. Es decir, el sistema oculta ciertos detalles de la manera en que los datos se almacenan y mantienen.
- Por debajo de la estructura de la base de datos se halla el **modelo de datos:** un conjunto de herramientas conceptuales para describir los datos, las relaciones entre ellos, la semántica de los datos y las restricciones de estos.
- El modelo de datos relacional es el modelo más atendido para el almacenamiento de datos en bases de datos. Otros modelos son el modelo orientado a objetos, el modelo relacional orientado a objetos y los modelos de datos semiestructurados.
- Un **lenguaje de manipulación de datos (LMD)** es un lenguaje que permite a los usuarios tener acceso a los datos o manipularlos. Los LMD no procedimentales, que solo necesitan que el usuario especifique los datos que necesita, sin aclarar exactamente la manera de obtenerlos, se usan mucho hoy en día.
- Un **lenguaje de definición de datos (LDD)** es un lenguaje para la especificación del esquema de la base de datos y otras propiedades de los datos.
- El diseño de bases de datos supone sobre todo el diseño del esquema de la base de datos. El modelo de datos entidad-relación (E-R) es un modelo de datos muy usado para el diseño de bases de datos. Proporciona una representación gráfica conveniente para ver los datos, las relaciones y las restricciones.
- Un sistema de bases de datos consta de varios subsistemas:
  - El subsistema **gestor de almacenamiento** proporciona la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación, así como las consultas remitidas al sistema.
  - El subsistema **procesador de consultas** compila y ejecuta instrucciones LDD y LMD.
- El **gestor de transacciones** garantiza que la base de datos permanezca en un estado consistente (correcto) a pesar de los fallos del sistema. El gestor de transacciones garantiza que la ejecución de las transacciones concurrentes se produzca sin conflictos.
- La arquitectura de un sistemas de base de datos se ve muy influida por el sistema computacional subyacente en que se ejecuta el sistema de la base de datos. Pueden ser centralizados o cliente-servidor, en el que una computadora servidora ejecuta el trabajo para múltiples computadoras cliente. Los sistemas de bases de datos también se diseñan para explotar arquitecturas computacionales paralelas. Las bases de datos distribuidas pueden extenderse geográficamente en múltiples computadoras separadas.
- Las aplicaciones de bases de datos suelen dividirse en un *front-end*, que se ejecuta en las máquinas clientes, y una parte que se ejecuta en *back-end*. En las arquitecturas de dos capas, el *front-end* se comunica directamente con una base de datos que se ejecuta en el *back-end*. En las arquitecturas de tres capas, la parte en *back-end* se divide a su vez en un servidor de aplicaciones y un servidor de bases de datos.
- Las técnicas de descubrimiento de conocimientos intentan descubrir automáticamente reglas y patrones estadísticos a partir de los datos. El campo de la minería de datos combina las técnicas de descubrimiento de conocimientos desarrolladas por los investigadores de inteligencia artificial y analistas estadísticos con las técnicas de implementación eficiente que les permiten utilizar bases de datos extremadamente grandes.
- Existen cuatro tipos diferentes de usuarios de sistemas de bases de datos, diferenciándose por las expectativas de interacción con el sistema. Se han desarrollado diversos tipos de interfaces de usuario diseñadas para los distintos tipos de usuarios.

Términos de repaso

- Sistema gestor de bases de datos (SGBD).
- Aplicaciones de sistemas de bases de datos.
- Sistemas procesadores de archivos.
- Inconsistencia de datos.
- Restricciones de consistencia.
- Abstracción de datos.
- Ejemplar de la base de datos.
- Esquema.
  - Esquema físico.
  - Esquema lógico.
- Independencia física de los datos.
- Modelos de datos.
  - Modelo entidad-relación.
  - Modelo de datos relacional.
  - Modelo de datos basado en objetos.
  - Modelo de datos semiestructurados.
- Lenguajes de bases de datos.
  - Lenguaje de definición de datos.
  - Lenguaje de manipulación de datos.
  - Lenguaje de consultas.
- Metadatos.
- Programa de aplicación.
- Normalización.
- Diccionario de datos.
- Gestor de almacenamiento.
- Procesador de consultas.
- Transacciones.
  - Atomicidad.
  - Recuperación de fallos.
  - Control de concurrencia.
- Arquitecturas de bases de datos de dos y tres capas.
- Minería de datos.
- Administrador de bases de datos (ABD).

Ejercicios prácticos

1.1. En este capítulo se han descrito varias ventajas importantes de los sistemas gestores de bases de datos. ¿Cuáles son sus dos inconvenientes?

1.2. Indique cinco formas en que los sistemas de declaración de tipos de lenguajes, como Java o C++, difieren de los lenguajes de definición de datos utilizados en una base de datos.

1.3. Indique seis pasos importantes que se deben dar para configurar una base de datos para una empresa dada.

1.4. Indique al menos tres tipos diferentes de información que una universidad debería mantener, además de la indicada en la Sección 1.6.2.

1.5. Suponga que quiere construir un sitio web similar a YouTube. Tenga en cuenta todos los puntos indicados en la Sección 1.2, como las desventajas de mantener los datos en un sistema de procesamiento de archivos. Justifique la importancia de cada uno de esos puntos para el almacenamiento de datos de vídeo real y de los metadatos del vídeo, como su título, el usuario que lo subió y los usuarios que lo han visto.

1.6. Las consultas por palabras clave que se usan en las búsquedas web son muy diferentes de las consultas de bases de datos. Indique las diferencias más importantes entre las dos, en términos de las formas en que se especifican las consultas y cómo son los resultados de una consulta.

Ejercicios

1.7. Indique cuatro aplicaciones que se hayan usado y que sea muy posible que utilicen un sistema de bases de datos para almacenar datos persistentes.

1.8. Indique cuatro diferencias significativas entre un sistema de procesamiento de archivos y un SGBD.

1.9. Explique el concepto de independencia física de los datos y su importancia en los sistemas de bases de datos.

1.10. Indique cinco responsabilidades del sistema gestor de bases de datos. Para cada responsabilidad, explique los problemas que surgirían si no se asumiera esa responsabilidad.

1.11. Indique al menos dos razones para que los sistemas de bases de datos soporten la manipulación de datos mediante un lenguaje de consultas declarativo, como SQL, en vez de limitarse a ofrecer una biblioteca de funciones de C o de C++ para llevar a cabo la manipulación de los datos.

1.12. Explique los problemas que causa el diseño de la tabla de la Figura 1.4.

1.13. ¿Cuáles son las cinco funciones principales del administrador de bases de datos?

1.14. Explique la diferencia entre la arquitectura en dos capas y en tres capas. ¿Cuál se acomoda mejor a las aplicaciones web? ¿Por qué?

1.15. Describa al menos tres tablas que se puedan utilizar para almacenar información en un sistema de redes sociales como Facebook.

Herramientas

Hay gran número de sistemas de bases de datos comerciales actualmente en uso. Entre los principales están: DB2 de IBM (www.ibm.com/software/data/db2), Oracle (www.oracle.com), SQL Server de Microsoft (www.microsoft.com/SQL), Sybase (www.sybase.com) e IBM Informix (www.ibm.com/software/data/informix). Algunos de estos sistemas están disponibles gratuitamente para uso personal o no comercial, o para desarrollo, pero no para su implantación real.

También hay una serie de sistemas de bases de datos gratuitos o de dominio público; los más usados son MySQL (www.mySQL.com) y PostgreSQL (www.postgreSQL.org).

Una lista más completa de enlaces a sitios web de fabricantes y a otras informaciones se encuentra disponible en la página inicial de este libro, en www.db-book.com.

Notas bibliográficas

A continuación se ofrece una relación de libros de propósito general, colecciones de artículos de investigación y sitios web sobre bases de datos. Los capítulos siguientes ofrecen referencias a materiales sobre cada tema descrito en ese capítulo.

Codd [1970] es el artículo histórico que introdujo el modelo relacional.

Entre los libros de texto que tratan los sistemas de bases de datos están: Abiteboul et ál. [1995], O'Neil y O'Neil [2000], Ramakrishnan y Gehrke [2002], Date [2003], Kifer et ál. [2005], Elmasri y Navathe [2006] y García-Molina et ál. [2008]. Un libro con artículos de investigación sobre gestión de base de datos se puede encontrar en Hellerstein and Stonebraker [2005].

Un repaso de los logros en la gestión de bases de datos y una valoración de los desafíos en la investigación futura aparecen en Silberschatz et ál. [1990], Silberschatz et ál. [1996], Bernstein et ál. [1998], Abiteboul et ál. [2003] y Agrawal et ál. [2009]. La página inicial del grupo de interés especial de la ACM en gestión de datos (www.acm.org/sigmod) ofrece gran cantidad de información sobre la investigación en bases de datos. Los sitios web de los fabricantes de bases de datos (véase *Herramientas* a la izquierda de estas *Notas*) proporciona detalles acerca de sus respectivos productos.