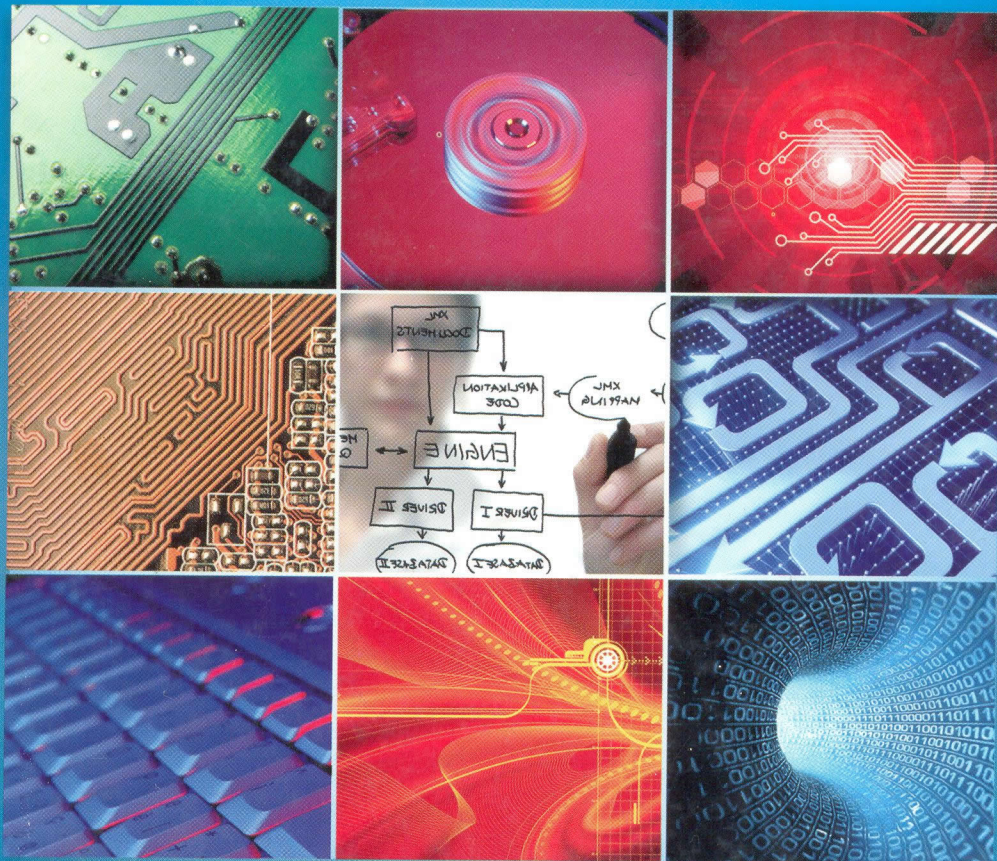


CORONEL / MORRIS / ROB

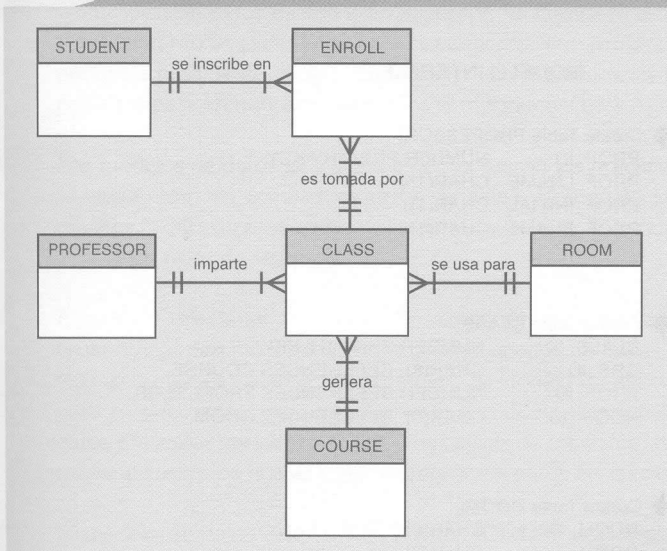
BASES DE DATOS

Diseño, implementación
y administración



NOVENA EDICIÓN

FIGURA 2.8 Modelo conceptual para Tiny College



Por tanto, los cambios en el hardware o el software del DBMS no tendrán efecto en el diseño de la base de datos en el nivel conceptual. En general, el término **diseño lógico** se usa para referirse a la tarea de crear un modelo conceptual de datos que podría ser implementado por cualquier DBMS.

2.6.3 EL MODELO INTERNO

Una vez seleccionado el DBMS específico, el modelo interno asocia el modelo conceptual con el DBMS. El **modelo interno** es la representación de la base de datos como es "vista" por el DBMS. En otras palabras, el modelo interno requiere que el diseñador equipare las características y restricciones del modelo conceptual con las del modelo de implementación seleccionado. Un **esquema interno** describe una representación específica de un modelo interno, usando los elementos de la base de datos soportados por la base de datos seleccionada.

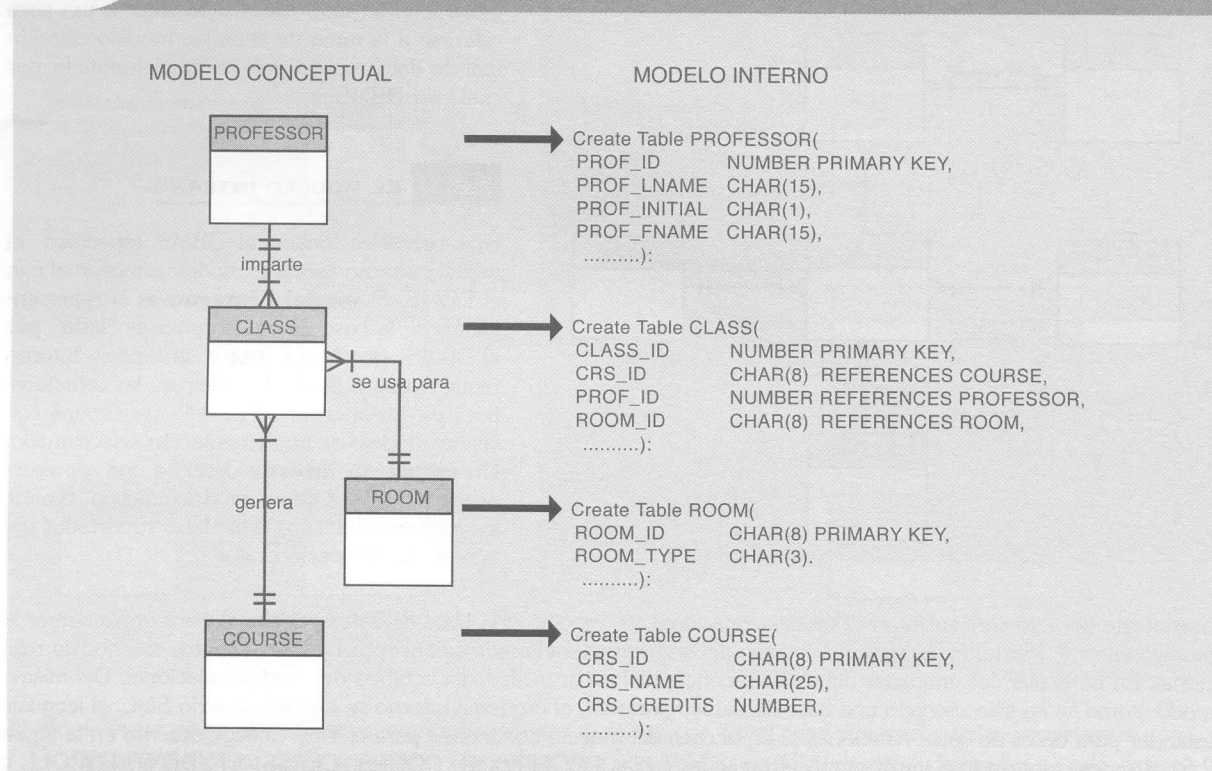
Como este libro se concentra en el modelo relacional, se escogió una base de datos relacional para implementar el modelo interno. Por tanto, el esquema interno debería trasladar el modelo conceptual a los elementos del modelo relacional. En particular, las entidades del modelo conceptual están trasladadas a tablas del modelo relacional. Del mismo modo, como se ha seleccionado una base de datos relacional, el esquema interno se expresa usando SQL, el lenguaje estándar para bases de datos relacionales. En el caso del modelo conceptual para el Tiny College descrito en la figura 2.8, el modelo interno se implementó al crear las tablas PROFESSOR, COURS, CLASS, STUDENT, ENROLL y ROOM. En la figura 2.9 se ve una versión simplificada del modelo interno para Tiny College.

El desarrollo de un modelo interno detallado es especialmente importante para diseñadores de bases de datos que trabajen con los modelos jerárquico o de red, porque estos modelos requieren especificación muy precisa del lugar de almacenamiento y de rutas de acceso de datos. En contraste, el modelo relacional requiere menos detalle en su modelo interno porque casi todos los RDBMS manejan *en forma transparente* la definición de ruta de acceso de datos; esto es, el diseñador no necesita estar consciente de los detalles de la ruta de acceso de datos. No obstante, incluso el software de base de datos relacional por lo general requiere la especificación del lugar de almacenamiento de datos, en especial en un ambiente de mainframes. Por ejemplo, DB2 requiere que se especifique el grupo de almacenamiento de datos, el lugar de la base de datos dentro del grupo de almacenamiento y el lugar de las tablas dentro de la base de datos.

Como el modelo interno depende de software específico de base de datos, se dice que es dependiente de software. Por tanto, un cambio en el software del DBMS requiere que el modelo interno sea cambiado para ajustarse a las características y necesidades del modelo de base de datos de implementación. Cuando se pueda cambiar el modelo interno sin afectar el modelo conceptual, tenemos **independencia lógica**. Sin embargo, el modelo interno es todavía independiente del hardware porque no es afectado por la selección de la computadora en la que se instale el software. Por tanto, un cambio en dispositivos de almacenamiento o hasta un cambio en sistemas operativos no afectará el modelo interno.

2.6.4 EL MODELO FÍSICO

El **modelo físico** opera al más bajo nivel de abstracción, describiendo la forma en que los datos se guardan en medios de almacenamiento como discos o cintas. El modelo físico requiere la definición de los dispositivos de almacenamiento físico y los métodos de acceso (físicos) requeridos para llegar a los datos dentro de esos dispositivos, haciéndolos dependientes de software y hardware. Las estructuras de almacenamiento utilizadas son dependientes del software (el DBMS y el sistema operativo) y del tipo de dispositivos de almacenamiento que la computadora pueda manejar. La precisión requerida en la definición del modelo físico demanda que los diseñadores de bases de datos que trabajen a este nivel tengan un conocimiento detallado del hardware y software utilizados para implementar el diseño de bases de datos.

FIGURA 2.9 Modelo interno para Tiny College


Los primeros modelos de datos forzaban al diseñador de bases de datos a tomar en cuenta los detalles de los requisitos de almacenamiento de datos del modelo físico. No obstante, el ahora dominante modelo relacional está dirigido principalmente al nivel lógico en lugar del físico; por tanto, no requiere los detalles a nivel físico que son comunes a sus predecesores.

Aun cuando el modelo relacional no necesita que el diseñador se ocupe de las características de almacenamiento físico de datos, la *implementación* de un modelo relacional puede requerir afinación a nivel físico para mejor funcionamiento. La afinación es especialmente importante cuando en un ambiente de mainframes se instalan bases de datos muy grandes. Sin embargo, hasta esa afinación de operación al nivel físico no requiere conocimiento de características de almacenamiento de datos físicos.

Como ya se dijo, el modelo físico es dependiente del DBMS, métodos de acceso a archivos y tipos de dispositivos de almacenamiento soportados por el sistema operativo. Cuando se pueda cambiar el modelo físico sin afectar el modelo interno tenemos **independencia física**. Por tanto, un cambio en dispositivos o métodos de almacenamiento y hasta un cambio en el sistema operativo no afectará el modelo interno.

En la tabla 2.4 se da un resumen de los niveles de abstracción de datos.

TABLA 2.4 Niveles de abstracción de datos

MODELO	GRADO DE ABSTRACCIÓN	ENFOQUE	INDEPENDIENTE DE
Externo	Alto	Vistas de usuario final	Hardware y software
Conceptual	↕	Vista global de datos (independiente del modelo de base de datos)	Hardware y software
Interno		Modelo específico de base de datos	Hardware
Físico	Bajo	Métodos de almacenamiento y acceso	Ni hardware ni software

En ambos esquemas, una de las transacciones espera a que la otra termine y libere sus bloqueos. No obstante, en muchos casos, una transacción solicita múltiples bloqueos. ¿Cuánto tiene que esperar una transacción para la petición de cada bloqueo? Obviamente, esa situación puede causar que algunas transacciones esperen indefinidamente, causando un *interbloqueo*. Para evitarlo, cada solicitud de bloqueo tiene un valor asociado de tiempo. Si el bloqueo no es concedido antes que expire el tiempo, la transacción se retorna.

10.5 CONTROL DE CONCURRENCIA CON MÉTODOS OPTIMISTAS

El **método optimista** está basado en la suposición de que la mayor parte de las operaciones en bases de datos no tienen conflicto. El método optimista no requiere de bloqueo ni de técnicas para estampas de tiempo. En cambio, una transacción es ejecutada sin restricciones hasta que es registrada. Con el uso de un método optimista, cada transacción se mueve por dos o tres fases, conocidas como *lectura*, *validación* y *escritura*.³

- Durante la *fase de lectura*, la transacción lee la base de datos, ejecuta los cálculos necesarios y hace las actualizaciones a una copia privada de los valores de la base de datos. Todas las operaciones de actualización de la transacción se registran en un archivo de actualización temporal, al que no tienen acceso las transacciones restantes.
- Durante la *fase de validación*, la transacción es ratificada para asegurarse que los cambios hechos no afectarán la integridad y consistencia de la base de datos. Si la prueba de validación es positiva, la transacción pasa a la fase de escritura. Si la prueba de validación es negativa, la transacción se reinicia y se desechan los cambios.
- Durante la *fase de escritura*, los cambios son permanentemente aplicados a la base de datos.
- El método optimista es aceptable para casi todos los sistemas de lectura o consulta de una base de datos que requieran pocas transacciones de actualización.

En un ambiente de uso intenso de DBMS, la administración de *interbloqueos* (su prevención y detección) constituye una importante función del DBMS. Éste usará una o más de las técnicas aquí estudiadas, así como variaciones de ellas. No obstante, el interbloqueo es a veces peor que la enfermedad que se supone se cure con él. Por tanto, puede ser necesario emplear técnicas para recuperación de una base de datos a fin de restablecerla a un estado consistente.

10.6 ADMINISTRACIÓN DE LA RECUPERACIÓN DE UNA BASE DE DATOS

La **recuperación de una base de datos** restablece una base de datos de un estado determinado (por lo general inconsistente) a un estado previamente consistente. Las técnicas de recuperación están basadas en la **propiedad atómica de la transacción**: todas las partes de la transacción deben ser tratadas como una sola unidad de trabajo lógica en la que todas las operaciones se aplican y completan para producir una base de datos consistente. Si, por alguna razón, no puede completarse alguna operación de transacción, ésta debe ser abortada y deben deshacerse cualesquier cambios a la base de datos. En pocas palabras, la recuperación de una transacción invierte todos los cambios que la transacción hizo a la base de datos antes que fuera abortada.

Aunque este capítulo ha destacado la recuperación de *transacciones*, las técnicas de recuperación también se aplican a la *base de datos* y al *sistema* después que haya ocurrido algún tipo de error crítico. Los eventos críticos pueden causar que una base de datos se haga no operacional y comprometa la integridad de los datos. Ejemplos de eventos críticos son:

- *Fallas de hardware/software*. Por ejemplo, podría ser una falla del disco duro, un condensador o una tarjeta defectuosos o un banco de memoria en mal estado. Otras causas de errores bajo esta categoría incluyen errores en un programa de aplicación o sistema operativo que hagan que los datos sean sobrescritos, borrados o perdidos. Algunos administradores de bases de datos afirman que ésta es una de las fuentes más comunes de problemas en bases de datos.

³ El método optimista para control de concurrencia está descrito en el artículo de H.T. King y J.T. Robinson, "Optimistic Methods for Concurrency Control", *ACM Transactions on Database Systems* 6(2), junio de 1981, pp. 213-226. Incluso el software más actual está construido sobre normas conceptuales que fueron desarrolladas hace más de dos décadas.

- *Incidentes causados por humanos.* Este tipo de evento se puede clasificar como no intencional o intencional.
 - Una falla no intencional es causada por descuido del usuario final. Estos errores incluyen borrar los renglones equivocados de una tabla, presionar la tecla equivocada o por accidente apagar el servidor principal de la base de datos.
 - Los eventos intencionales son de naturaleza más severa y normalmente indican que los datos de la compañía están en grave riesgo. Bajo esta categoría están las amenazas de seguridad causadas por piratas cibernéticos (hackers) que tratan de ganar acceso no autorizado a recursos de datos y ataques de virus causados por empleados descontentos que tratan de comprometer la operación de la base de datos y dañar a la compañía.
- *Desastres naturales.* Esta categoría incluye incendios, terremotos, inundaciones y fallas de energía eléctrica.

Cualquiera que sea la causa, el error crítico puede inutilizar la base de datos y dejarla en estado inconsistente. La siguiente sección presenta las diversas técnicas usadas para recuperar la base de datos de un estado inconsistente a uno consistente.

10.6.1 RECUPERACIÓN DE TRANSACCIÓN

En la sección 10.1.4, aprendimos acerca de las bitácoras de transacciones y la forma en que éste contiene información para fines de recuperación de la base de datos. La recuperación de transacciones usa datos del registro para recuperar la base de datos de un estado inconsistente a uno consistente.

Antes de continuar, examinemos cuatro conceptos importantes que afectan el proceso de recuperación:

- El **protocolo de escritura adelantada en el registro** asegura que las bitácoras de transacción siempre se escriban *antes* que se actualicen realmente cualesquier datos de una base de datos. Este protocolo asegura que, en caso de una falla, la base de datos pueda ser recuperada más tarde a un estado consistente, usando los datos de la bitácora.
- Las **bitácoras de transacción redundante** (varias copias de la bitácora de transacción) aseguran que una falla del disco físico no dañará la capacidad del DBMS para recuperar datos.
- Las **memorias intermedias** de una base de datos son áreas de almacenamiento temporal en la memoria principal usadas para acelerar operaciones de disco. Para mejorar el tiempo de procesamiento, el software del DBMS lee los datos del disco físico y guarda una copia de ellos en una "memoria intermedia" en la memoria principal. Cuando una transacción actualiza datos, en realidad modifica la copia que está en la memoria intermedia porque ese proceso es mucho más rápido que tener acceso al disco físico cada vez. Después, todas las memorias intermedias que contengan datos actualizados se escriben en un disco físico durante una sola operación, con lo cual se ahorra un tiempo de procesamiento considerable.
- Los **puntos de verificación** son operaciones en las que el DBMS escribe todas sus memorias intermedias actualizadas en el disco. Mientras esto ocurre, el DBMS no ejecuta ninguna otra solicitud. Una operación de punto de verificación también se registra en la bitácora de transacciones. Como resultado de esta operación, la base de datos física y la bitácora de transacción estarán en sincronía. Esta sincronización se requiere porque operaciones de actualización renuevan la copia de los datos de las memorias intermedias y no en la base de datos física. Los puntos de verificación son programados automáticamente por el DBMS varias veces por hora. Como veremos a continuación, los puntos de verificación también desempeñan una importante función en la recuperación de transacciones.

El proceso de recuperación de una base de datos involucra llevar ésta a un estado consistente después de una falla. Los procedimientos de recuperación de una base de datos generalmente hacen uso de técnicas de escritura diferida y de escritura directa o continua.

Cuando el procedimiento de recuperación usa una **técnica de escritura diferida** (también llamada **actualización diferida**), las operaciones de una transacción no actualizan de inmediato la base de datos física. En cambio, sólo se actualiza la bitácora de transacción. La base de datos se actualiza físicamente sólo después que la transacción llegue a su punto de registro, usando información de la bitácora de transacción. Si la transacción aborta antes de llegar a su punto de registro, no es necesario hacer cambios (no *ROLLBACK* o deshacer) a la base de datos porque ésta nunca fue actualizada. El proceso de recuperación para todas las transacciones iniciadas y registradas (antes de la falla) sigue estos pasos:

1. Identificar el último punto de verificación en la bitácora de transacción. Ésta es la última vez que los datos de una transacción se guardaron físicamente al disco.
2. Para una transacción que se inició y fue registrada antes del último punto de verificación, no es necesario hacer nada porque los datos ya están guardados.

3. Para una transacción que realizó una operación de registro después del último punto de verificación, el DBMS usa los registros de la bitácora de transacción para rehacer la transacción y actualizar la base de datos, usando los valores de "posterior" en la bitácora de transacción. Los cambios se hacen en orden descendente, del más viejo al más reciente.
4. Para cualquier transacción que tenía una operación ROLLBACK después del último punto de verificación, o que fue dejado en activo (ni con COMMIT ni un ROLLBACK) antes que ocurriera la falla, no es necesario hacer nada porque la base de datos nunca fue actualizada.

Cuando el procedimiento de recuperación usa una **técnica de escritura directa** (también llamada **actualización inmediata**), la base de datos es actualizada inmediatamente por operaciones de transacción durante la ejecución de ésta, o incluso antes que alcance su punto de registro. Si la transacción aborta antes de alcanzar su punto de registro, es necesario hacer una operación ROLLBACK o deshacer para restablecer la base de datos a un estado consistente. En ese caso, la operación ROLLBACK usará los valores de la bitácora de transacción "previos". El proceso de recuperación sigue estos pasos:

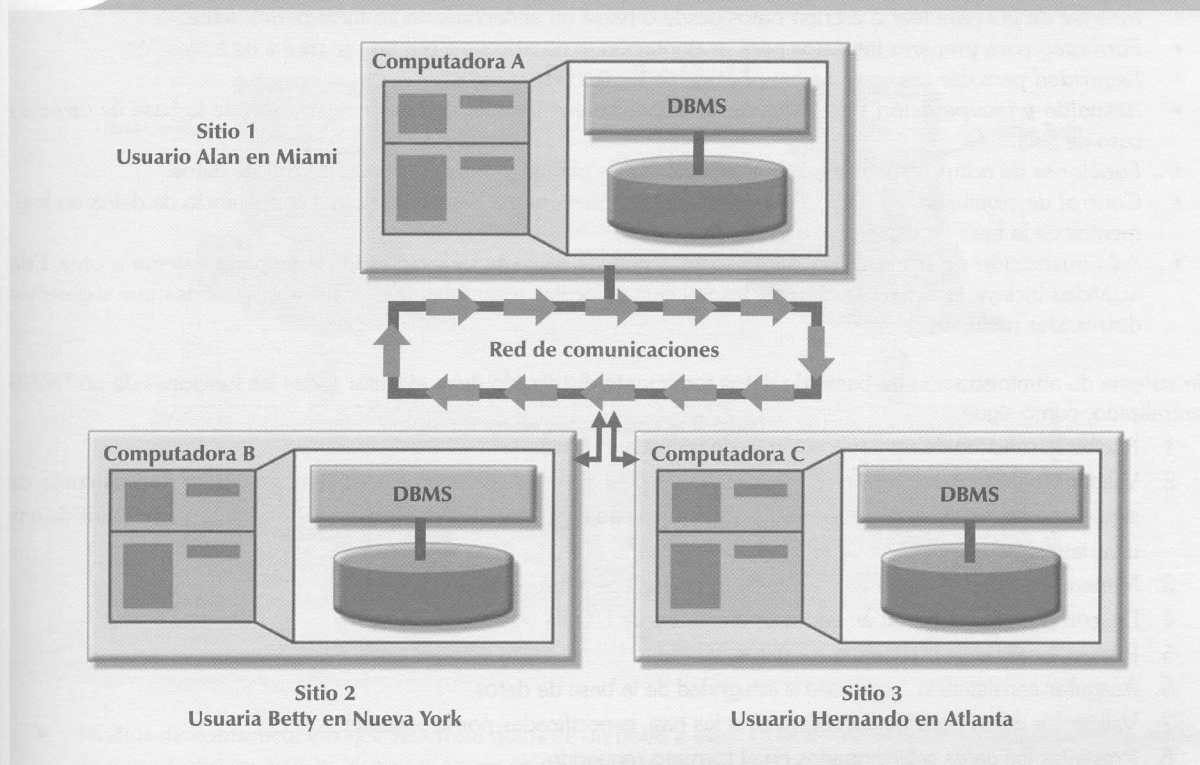
1. Identificar el último punto de verificación de la bitácora de transacción. Ésta es la última vez que los datos de transacción se guardaron físicamente en disco.
2. Para una transacción que se inició y fue registrada antes del último punto de verificación, no es necesario hacer nada porque los datos ya están guardados.
3. Para una transacción que fue registrada después del último punto de verificación, el DBMS la rehace usando los valores "posterior" de la bitácora de transacción. Se aplican cambios en orden ascendente, del más viejo al más reciente.
4. Para cualquier transacción que tuvo una operación ROLLBACK después del último punto de verificación, o que se dejó activa (sin COMMIT ni ROLLBACK) antes que ocurriera la falla, el DBMS usa los registros de la bitácora de transacción a ROLLBACK u operaciones de deshacer, usando los valores "previos" de la bitácora de transacción. Se aplican cambios en orden inverso, del más reciente al más viejo.

Use la bitácora de transacción de la tabla 10.5 para dar seguimiento a un proceso simple de recuperación de una base de datos. Para asegurarse de entenderlo, se emplea una simple bitácora de transacción que incluye tres transacciones y un punto de verificación. Esta bitácora comprende componentes de transacción empleados antes en el capítulo, de modo que el lector ya debe estar familiarizado con el proceso básico. Dada la transacción, la bitácora tiene las siguientes características:

- La transacción 101 consta de dos enunciados UPDATE que reducen la cantidad a mano para el producto 54778-2T y aumentan el saldo del cliente para el cliente 10011 por una venta a crédito de dos unidades del producto 54778-2T.
- La transacción 106 es el mismo evento de venta a crédito que vimos en la sección 10.1.1. Esta transacción representa la venta a crédito de una unidad del producto 89-WRE-Q al cliente 10016 por la cantidad de \$277.55. Esta transacción consta de cinco enunciados DML de SQL: tres enunciados INSERT y dos UPDATE.
- La transacción 155 representa una simple actualización de inventario. Esta transacción consta de un enunciado UPDATE que aumenta la cantidad disponible del producto 2232/QWE de seis a 26 unidades.
- Un punto de verificación de base de datos escribió en el disco todas las memorias intermedias actualizadas. El evento del punto de verificación escribe sólo los cambios para todas las transacciones previamente registradas. En este caso, el punto de verificación aplica todos los cambios hechos por la transacción 101 a los archivos de datos de la base de datos.

Usando la tabla 10.15, ahora se puede dar seguimiento al proceso de recuperación de una base de hecho por un DBMS, empleando el método de actualización diferida como sigue:

1. Identificar el último punto de verificación. En este caso, el último punto de verificación fue TRL ID 423. Ésta fue la última vez que las memorias intermedias de la base de datos se escribieron físicamente al disco.
2. Observe que la transacción 101 inicia y termina antes del último punto de verificación. Por tanto, todos los cambios ya se escribieron en el disco y no es necesario tomar ninguna acción adicional.
3. Por cada transacción que se registró después del último punto de verificación (TRL ID 423), el DBMS usará los datos del registro de transacción para escribir los cambios a disco, usando los valores "posterior". Por ejemplo, para la transacción 106:
 - a) Encuentre COMMIT (TRL ID 457).
 - b) Use los valores previos de apuntador para localizar el inicio de la transacción (TRL ID 397).
 - c) Use los valores siguientes de apuntador para localizar cada enunciado DML y aplique los cambios a disco, usando los valores "posterior". (Inicie con TRL ID 405, después 415, 419, 427 y 431.) Recuerde que TRL ID 457 fue el enunciado COMMIT para esta transacción.
 - d) Repita el proceso para la transacción 155.
4. Cualesquiera otras transacciones serán ignoradas. Por tanto, para transacciones que terminen con ROLLBACK o que se dejaron activas (las que no terminen con un COMMIT o ROLLBACK), nada se hace porque no se escribieron cambios en disco.

**FIGURA
12.3****Ambiente de base de datos distribuida**

La base de datos de la figura 12.3 está dividida en tres fragmentos de base de datos (E1, E2 y E3) situados en lugares diferentes. Las computadoras están conectadas mediante un sistema de red. En una base de datos completamente distribuida, los usuarios Alan, Betty y Hernando no necesitan saber el nombre o ubicación de cada fragmento de la base de datos para tener acceso a esta última. También, los usuarios podrían estar ubicados en sitios que no sean Miami, Nueva York o Atlanta y aun así tener acceso a la base de datos como una sola unidad lógica.

Al examinar las figuras 12.2 y 12.3, deben recordarse los siguientes puntos:

- El procesamiento distribuido no requiere una base de datos distribuida, sino que una base de datos distribuida requiere procesamiento distribuido (cada fragmento de una base de datos es administrado por su propio proceso local de base de datos).
- El procesamiento distribuido puede estar basado en una sola base de datos ubicada en una sola computadora. Para que ocurra la administración de datos distribuidos, copias o partes de las funciones de procesamiento de la base de datos deben estar distribuidas a todos los sitios de almacenamiento de datos.
- Tanto el procesamiento distribuido como las bases de datos distribuidas requieren de una red para conectar todos los componentes.

12.4 CARACTERÍSTICAS DE LOS SISTEMAS DE ADMINISTRACIÓN DE BASES DE DATOS DISTRIBUIDAS

Un DDBMS gobierna el almacenamiento y procesamiento de datos lógicamente relacionados, mediante sistemas interconectados computarizados en los que los datos y funciones de procesamiento están distribuidos entre varios sitios. Un DBMS debe tener al menos las siguientes funciones para ser clasificado como distribuido:

- *Interfaz de aplicación* para interactuar con el usuario final, programas de aplicación y otros DBMS dentro de la base de datos distribuida.
- *Validación* para analizar solicitudes de datos para corrección de la sintaxis.
- *Transformación* para descomponer las solicitudes complejas en componentes atómicos de solicitud de datos.

14.1 CONECTIVIDAD DE UNA BASE DE DATOS

La *conectividad de una base de datos* se refiere a los mecanismos por medio de los cuales los programas de aplicación se conectan y comunican con depósitos de datos. El software de conectividad de bases de datos también se conoce como **middleware de la base de datos** porque proporciona una interfaz entre el programa de aplicación y la base de datos. El depósito de datos, también conocido como *fuentes de datos*, representa la aplicación de la administración de datos, por ejemplo Oracle RDBMS, SQL Server DBMS, o IBM DBMS, que se usarán para guardar los datos generados por el programa de aplicación. En el ideal, una fuente o depósito de datos podría encontrarse en cualquier parte y tener cualquier tipo de datos. Por ejemplo, la fuente de datos podría ser una base de datos relacional, una jerárquica, una hoja de cálculo o un archivo de datos de texto.

La necesidad de interfaces de conectividad de bases de datos estándar no se puede exagerar. Así como SQL se ha convertido en el lenguaje de facto para manipulación de datos, hay necesidad de una interfaz estándar para conectividad de bases de datos que permita que aplicaciones se conecten a depósitos de datos. Hay numerosas formas de obtener conectividad de bases de datos. Esta sección estudia sólo las siguientes interfaces:

- Conectividad nativa de SQL (proporcionada por el vendedor).
- Conectividad de base de datos abierta de Microsoft (ODBC), objetos de acceso a datos (DAO) y objetos de datos remotos (RDO).
- Vinculación e incrustación de objetos para base de datos de Microsoft (OLE-DB).
- Objetos de datos ActiveX de Microsoft (ADO.NET)
- Conectividad de base de datos Java de Sun (JDBC).

No debe sorprender que la mayor parte de las interfaces que se encuentre sean productos de Microsoft. Después de todo, las aplicaciones cliente se conectan a bases de datos y casi todas esas aplicaciones se ejecutan en computadoras que son activadas por alguna versión de Microsoft Windows. Las interfaces de conectividad de datos ilustradas aquí son participantes dominantes en el mercado y, lo más importante, disfrutan del apoyo de la mayoría de vendedores de bases de datos. De hecho, las ODBC, OLE-DB y ADO.NET forman la espina dorsal de la arquitectura **Universal Data Access (UDA)** de Microsoft, que es un conjunto de tecnologías que se emplean para tener acceso a cualquier tipo de fuente de datos y manejar éstos a través de una interfaz común. Como se observa, las interfaces de conectividad de base de datos de Microsoft han evolucionado con el tiempo: cada una se construye sobre otra, dando así funcionalidad, características, flexibilidad y mejor soporte.

14.1.1 CONECTIVIDAD SQL NATIVA

La mayoría de los vendedores de DBMS cuentan con sus propios métodos para conectarse a sus bases de datos. La conectividad SQL nativa se refiere a la interfaz de conexión que es proporcionada por el vendedor de la base de datos y que es única. El mejor ejemplo de este tipo de interfaz nativa es el RDBMS de Oracle. Para conectar una aplicación cliente a una base de datos de Oracle, es necesario instalar y configurar la interfaz SQL*Net de Oracle en la computadora cliente. La figura 14.1 muestra la configuración de la interfaz SQL*NET de Oracle en la computadora cliente.

Las interfaces para conectividad de base de datos nativa se optimizan para “sus” DBMS y esas interfaces dan soporte al acceso a casi todas, pero no a todas, las funciones de la base de datos. No obstante, mantener múltiples interfaces nativas para diferentes bases de datos puede convertirse en carga para el programador. Por tanto, surge la necesidad de una conectividad “universal” de bases de datos. Por lo general, la interfaz nativa para conectividad de base de datos proporcionada por el vendedor no es la única manera de conectarse a una base de datos; casi todos los productos actuales de los DBMS soportan otros estándares para conectividad de bases de datos, siendo la más común la ODBC.

14.1.2 ODBC, DAO Y RDO

Ideada a principios de 1990, la **conectividad de base de datos abierta (ODBC)** es la implementación de Microsoft de un superconjunto del estándar **interfaz a nivel de llamada (CLI)** del grupo de acceso de SQL para acceder a bases de datos. Es probable que la ODBC sea la interfaz más soportada para conectividad a bases de datos. La ODBC permite que cualquier aplicación para Windows tenga acceso a fuentes de datos relacionales, usando SQL mediante una **interfaz de programación de aplicación (API)** estándar. El diccionario Webopedia en línea (www.webopedia.com) define una API como “un conjunto de rutinas, protocolos y herramientas para construir aplicaciones de software”. Una buena API facilita el desarrollo de un programa al suministrar todos los elementos de construcción; el programador

al cliente (incluido el controlador de base de datos JDBC y toda la información de configuración) y a continuación se ejecutan con seguridad en el ambiente de tiempo de ejecución del cliente.

Todos los días, más y más empresas están invirtiendo recursos en el desarrollo y expansión de su presencia en la web y buscando formas para hacer más negocio en internet. Esas empresas van a generar cantidades crecientes de datos que se guardarán en bases de datos. Java y .NET son parte de la tendencia hacia una creciente seguridad en internet como recurso crítico de negocios. De hecho, es probable que internet sea la plataforma de desarrollo del futuro. En la siguiente sección aprenderemos más acerca de bases de datos de internet y cómo se usan.

14.2 BASES DE DATOS EN INTERNET

Millones de personas en todo el mundo tienen acceso a internet, conectándose a bases de datos por medio de navegadores web o servicios de datos (por ejemplo, usando un applet de teléfono inteligente para obtener información del clima). La conectividad de bases de datos de internet abre la puerta a nuevos e innovadores servicios que:

- Permiten respuestas rápidas a presiones de la competencia llevando con celeridad nuevos servicios y productos al mercado.
- Aumentan la satisfacción de clientes mediante la creación de servicios de soporte basados en la web.
- Permiten acceso a datos en cualquier lugar/a cualquier hora usando equipos inteligentes móviles vía internet.
- Dan diseminación de información rápida y eficaz por medio de un acceso universal desde la acera de enfrente o desde el otro lado del mundo.

Dadas esas ventajas, muchas organizaciones se apoyan en sus departamentos de servicios de internet (IS) para crear arquitecturas de acceso a datos universales basadas en normas de esa red. La tabla 14.3 presenta una muestra de características de tecnología de internet y los beneficios que brindan.

TABLA 14.3

Características y beneficios de tecnologías de internet

CARACTERÍSTICA DE INTERNET	BENEFICIO
Independencia de hardware y software	Ahorros en adquisición de equipo/software Capacidad para ejecutarse en casi todos los equipos existentes Independencia y portabilidad de plataforma No hay necesidad de desarrollo multiplataformas
Interfaz de usuario común y sencilla	Tiempo y costo reducidos en capacitación Costo reducido para soporte a usuario final No hay necesidad de desarrollo multiplataformas
Independencia de lugar	Acceso mundial vía infraestructura de internet y dispositivos inteligentes móviles Necesidades (y costos) reducidas para conexiones dedicadas
Rápido desarrollo a costos manejables	Disponibilidad de múltiples herramientas de desarrollo Herramientas de desarrollo de conectar y operar (normas abiertas) Desarrollo más interactivo Tiempos reducidos de desarrollo Herramientas relativamente baratas Herramientas gratis para acceso a clientes (navegadores web) Bajo costo de entrada. Disponibilidad frecuente de servidores web gratis. Costos reducidos de mantenimiento de redes privadas Procesamiento y escalabilidad distribuidas, usando múltiples servidores

En el actual ambiente de información de negocios y mundial, es fácil ver por qué muchos profesionales de bases de datos consideran que la conexión DBMS a internet es un elemento de importancia decisiva en el desarrollo de servicios de internet (IS). Como se verá en las siguientes secciones, el desarrollo de aplicaciones de bases de datos, en particular la creación, administración de interfaces de usuario y conectividad de bases de datos, son profundamente afectadas por la web. No obstante, tener una interfaz de base de datos basada en la web no cancela los problemas de diseño e implementación de una base de datos que se abordaron en los capítulos previos. En el análisis final, si usted hace una compra en línea o si hace fila en una tienda, los detalles de transacción a nivel de sistemas son esencialmente iguales y requieren las mismas estructuras y relaciones básicas de base de datos. Si cualquier lección inmediata ha de

aprenderse, es ésta: los efectos de un diseño, implementación y administración deficientes en una base de datos se multiplican en un ambiente en el que las transacciones podrían medirse en cientos de miles por día y no en cientos por día.

Internet está cambiando rápidamente la forma en que se genera, se tiene acceso y se distribuye información. En el corazón de este cambio está la capacidad de la web para tener acceso a datos en bases de datos (locales y remotas), la sencillez de la interfaz y la funcionalidad de plataformas cruzadas (heterogéneas). La web ha ayudado a crear un nuevo estándar para diseminación de información.

Las siguientes secciones examinan la forma en que el middleware de la web a la base de datos hace posible que los usuarios finales interactúen con bases de datos en la web.

14.2.1 MIDDLEWARE DE LA WEB A LA BASE DE DATOS: EXTENSIONES EN EL LADO DEL SERVIDOR

En general, el servidor web es el centro a través del cual se tiene acceso a todos los servicios de internet. Por ejemplo, cuando un usuario final emplea un navegador web para consultar dinámicamente una base de datos, el navegador cliente solicita una página web. Cuando el servidor web recibe la petición de página, la busca en el disco duro; cuando la encuentra (por ejemplo, una cotización de acciones, información de un catálogo de productos o una lista de tarifas aéreas), la envía al cliente.



CONTENIDO EN LÍNEA

Los sistemas cliente/servidor se estudian en detalle en el **Apéndice F** que se encuentra en el sitio web Premium para este libro.

Las páginas web dinámicas están en el corazón de los actuales sitios web. En esta situación de consulta a la base de datos, el servidor web genera el contenido de la página web antes de enviarlo al navegador web cliente. El único problema con la precedente situación de consulta es que el servidor web debe incluir el resultado de la consulta a la base de datos en la página *antes* de enviarla al cliente. Desafortunadamente, ni el navegador web ni el servidor web saben cómo conectarse y leer los datos de la base de datos. Por tanto, para soportar este tipo de consulta (consulta a la base de datos), la capacidad del servidor web debe extenderse para que pueda entender y procesar las solicitudes de la base de datos. Este trabajo es realizado mediante una extensión del lado del servidor.

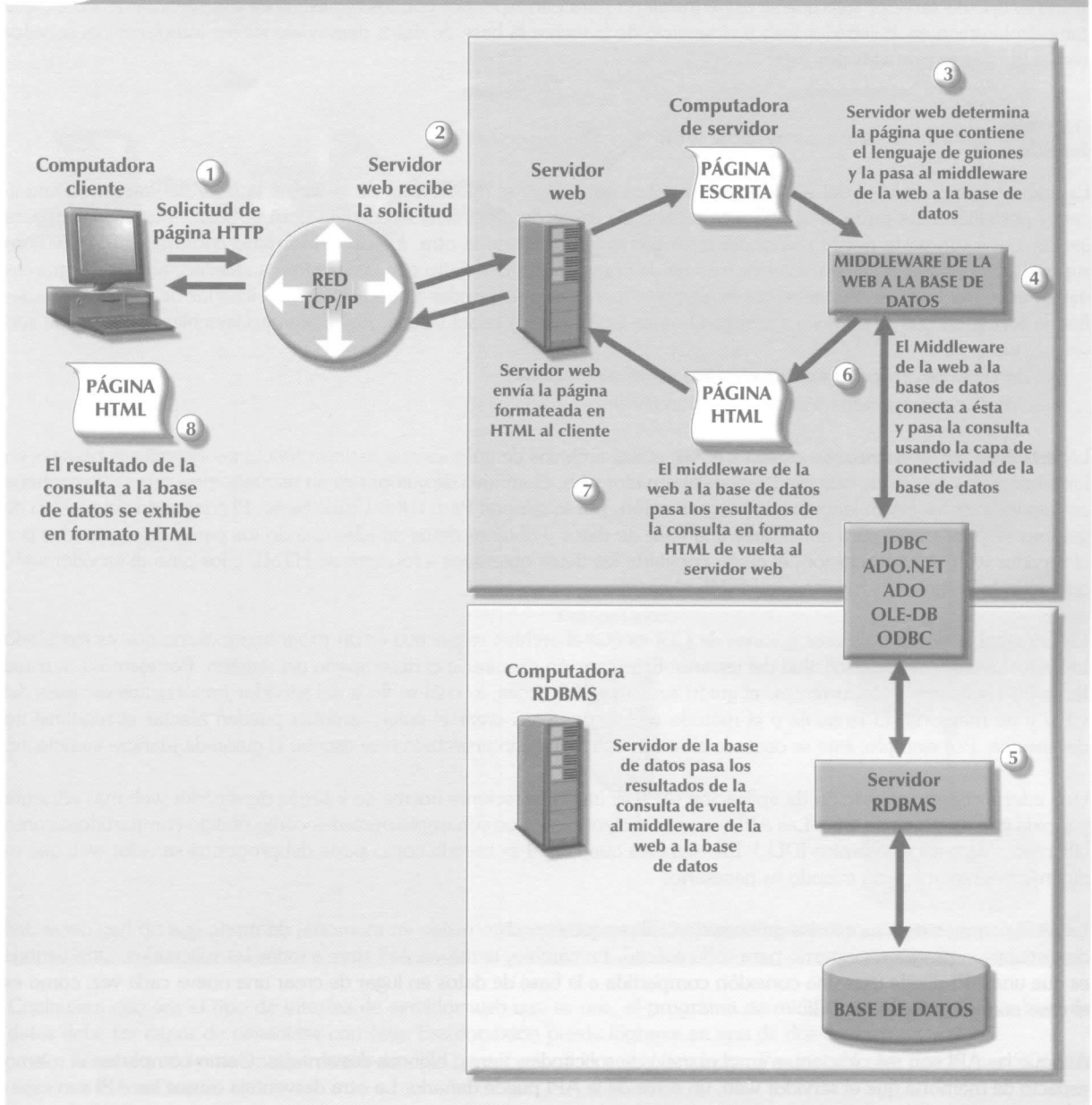
Una **extensión del lado de servidor** es un programa que interactúa directamente con el servidor web para manejar tipos específicos de solicitudes. En el ejemplo precedente de consulta a la base de datos, el programa de extensión del lado de servidor obtiene los datos de las bases de datos y los pasa al servidor web, que, a su vez, los envía al navegador cliente para fines de presentación. La extensión de lado del servidor hace posible obtener y presentar los resultados de la consulta, pero lo que es más importante es que *proporciona sus servicios al servidor web en una forma totalmente transparente al navegador cliente*. En pocas palabras, la extensión del lado de servidor agrega suficiente funcionalidad al servidor web y, por tanto, a internet.

Un programa de extensión del lado del servidor de la base de datos también se conoce como **middleware de la web a la base de datos**. La figura 14.18 muestra la interacción entre el navegador, el servidor web y el middleware de la web a la base de datos.

Siga las acciones del middleware de la web a la base de datos de la figura 14.8:

1. El navegador cliente envía una solicitud de página al servidor web.
2. El servidor recibe y valida la solicitud. En este caso, el servidor la pasará al middleware de la web a la base de datos para procesarla. En general, la página pedida contiene algún tipo de lenguaje de guiones para hacer posible la interacción con la base de datos.

FIGURA 14.8 Middleware de la web a la base de datos



3. El middleware de la web a la base de datos lee, valida y ejecuta el guión. En este caso, conecta a la base de datos y pasa la consulta usando la capa de conectividad de la base de datos.
4. El servidor de la base de datos ejecuta la consulta y pasa el resultado al middleware de la web a la base de datos.
5. El middleware de la web a la base de datos compila el conjunto de resultado, dinámicamente genera una página con formato de HTML que incluye los datos obtenidos de la base de datos y la envía al servidor web.
6. El servidor web entrega la página de HTML recién creada, que ahora incluye el resultado de la consulta, al navegador cliente.
7. El navegador cliente exhibe la página en la computadora local.

La interacción entre el servidor web y el middleware de la web a la base de datos es crucial para el desarrollo de una implementación exitosa de la base de datos de internet. Por tanto, el middleware debe estar bien integrado con los otros servicios y los componentes de internet que intervienen en su uso. Por ejemplo, al instalar el middleware éste debe verificar el tipo de servidor web que se use e instalarlo para corresponder con los requisitos de ese servidor. Además, qué también interactúen el servidor web y el servicio de la web a la base de datos dependerá de las interfaces del servidor web que están soportadas por éste.

14.2.2 INTERFACES DE SERVIDOR WEB

Extender la funcionalidad del servidor web implica que éste y el middleware de la web a la base de datos se comuniquen perfectamente entre sí. (Los profesionales de bases de datos con frecuencia usan la palabra *interoperar* para indicar que cada parte puede responder a las comunicaciones de la otra. El uso del término *comunicar* en este libro supone interoperación.) Si un servidor web ha de comunicarse con éxito con un programa externo, ambos programas deben usar una forma estándar para intercambiar mensajes y responder a solicitudes. Una interfaz de servidor web define la forma en que se comunica con programas externos. En la actualidad, hay dos interfaces bien definidas de servidores web:

- Interfaz de compuerta común (CGI).
- Interfaz de programación de aplicación (API)

La **interfaz de compuerta común (CGI)** utiliza archivos de guiones que realizan funciones específicas basadas en los parámetros del cliente que son pasados al servidor web. El archivo de guiones es un pequeño programa que contiene comandos escritos en un lenguaje de programación, por lo general Perl, C# o Visual Basic. El contenido del archivo de guiones se puede usar para conectarse a la base de datos y obtener datos de ella, usando los parámetros pasados por el servidor web. A continuación, el guión convierte los datos obtenidos a formato de HTML y los pasa al servidor web, que envía la página con formato de HTML al cliente.

La principal desventaja de usar guiones de CGI es que el archivo respectivo es un programa externo que es ejecutado individualmente por cada solicitud del usuario. Esa situación disminuye el desempeño del sistema. Por ejemplo, si usted tiene 200 solicitudes concurrentes, el guión se carga 200 veces, lo cual se lleva del servidor importantes recursos del CPU y de memoria. El lenguaje y el método empleados para crear el guión también pueden afectar el rendimiento del sistema. Por ejemplo, éste se degrada si se usa un lenguaje interpretado o se escribe el guión de manera ineficiente.

Una interfaz de programación de aplicación (API) es una más reciente norma de interfaz de servidor web más eficiente y rápida que un guión de CGI. Las API son más eficientes porque son implementadas como código compartido o como bibliotecas de enlace dinámico (DLL). Eso significa que la API es tratada como parte del programa servidor web que es dinámicamente invocado cuando es necesario.

Las API son más rápidas que los guiones de CGI porque el código reside en memoria, de modo que no hay necesidad de ejecutar un programa externo para cada solicitud. En cambio, la misma API sirve a todas las solicitudes. Otra ventaja es que una API puede usar una conexión compartida a la base de datos en lugar de crear una nueva cada vez, como es el caso con los guiones de CGI.

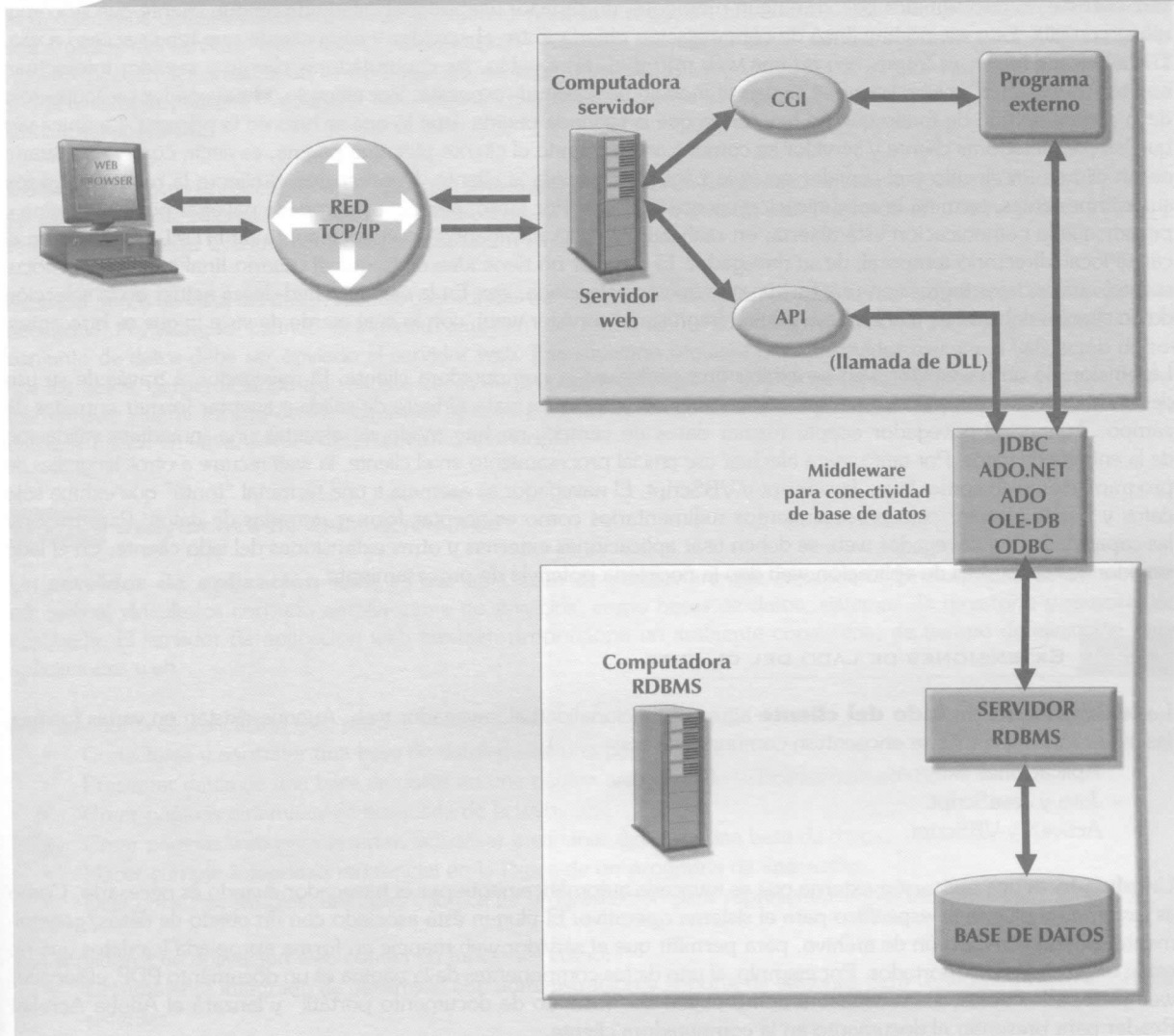
Aunque las API son más eficientes en el manejo de solicitudes, tienen algunas desventajas. Como comparten el mismo espacio de memoria que el servidor web, un error de la API puede dañarlo. La otra desventaja es que las API son específicas para el servidor web y para el sistema operativo.

Al momento de escribir este libro, hay cuatro API bien establecidas del servidor web:

- API de servidor de internet (ISAPI) para servidores web Microsoft Windows.
- WebSite API (WSAPI) para servidores web O'Reilly.
- JDBC para proporcionar conectividad a base de datos para aplicaciones Java.

Los diversos tipos de interfaces web se ilustran en la figura 14.9.

FIGURA 14.9 Interfaces de CGI y API para servidor web



Cualquiera que sea el tipo de interfaz de servidor web que se use, el programa de middleware de la web a la base de datos debe ser capaz de conectarse con ésta. Esa conexión puede lograrse en una de dos formas:

- Usar el middleware nativo de acceso SQL proporcionado por el vendedor. Por ejemplo, se puede usar SQL*Net si se usa Oracle.
- Usar los servicios de estándares generales de conectividad de base de datos como conectividad de base de datos abierta (ODBC), vinculación e incrustación de objetos para base de datos (OLE-DB), objetos de datos de ActiveX para interfaz .NET (ADO.NET), o conectividad de base de datos Java (JDBC).

14.2.3 EL NAVEGADOR WEB

El navegador web es el software de aplicación en la computadora cliente, por ejemplo Microsoft Internet Explorer, Apple Safari, o Mozilla Firefox, que permite a usuarios finales navegar por la web. Cada vez que el usuario dé un clic en un hipervínculo, el navegador genera una solicitud de página HTTP GET que es enviada al servidor web designado, usando el protocolo TCP/IP de internet.

El trabajo del navegador web es *interpretar* el código de HTML que reciba del servidor web y presentar los diversos componentes de la página en un formato estándar. Por desgracia, las capacidades de interpretación y presentación del navegador no son suficientes para desarrollar aplicaciones basadas en la web. Esto es porque la web es un **sistema sin estado**, lo cual significa que en ningún momento, un servidor web sabe el estado de ningún cliente que se comunique con ella. Esto es, no hay línea de comunicación abierta entre el servidor y cada cliente que tenga acceso a ella, lo cual, desde luego, es impráctico en una web *mundial*. En cambio, las computadoras cliente y servidor interactúan en “conversaciones” muy cortas que siguen el modelo de solicitud-respuesta. Por ejemplo, el navegador se ocupa sólo de la página *actual*, de modo que no hay forma que la segunda página sepa lo que se hizo en la primera. La única vez que las computadoras cliente y servidor se comunican es cuando el cliente pide una página, es decir, cuando el usuario da un clic en un vínculo y el servidor envía la página respectiva al cliente. Una vez que el cliente la reciba junto con sus componentes, termina la comunicación cliente/servidor. Por tanto, aunque usted pueda navegar por una página y *pensar* que la comunicación está abierta, en realidad sólo está navegando en el documento de HTML guardado en el caché local (directorio temporal) de su navegador. El servidor no tiene idea de lo que el usuario final haga con el documento, cuáles datos ingresa en una forma, qué opción selecciona, etc. En la web, si usted desea actuar en la selección de un cliente, debe saltar a una nueva página (regresar al servidor web), con lo cual pierde de vista lo que se hizo antes.

La función de un navegador web es exhibir una página en la computadora cliente. El navegador, a través de su uso del HTML, no tiene capacidad computacional más allá de dar formato al texto de salida y aceptar formar entradas de campo. Aunque el navegador acepte formar datos de campo, no hay modo de ejecutar una inmediata validación de la entrada de datos. Por tanto, para efectuar ese crucial procesamiento en el cliente, la web recurre a otros lenguajes de programación web como Java, JavaScript y VBScript. El navegador se asemeja a una terminal “tonta” que exhibe sólo datos y puede ejecutar sólo procesamientos rudimentarios como es aceptar formar entradas de datos. Para mejorar las capacidades del navegador web, se deben usar aplicaciones externas y otras extensiones del lado cliente. En el lado servidor, los servidores de aplicación web dan la necesaria potencia de procesamiento.

14.2.4 EXTENSIONES DE LADO DEL CLIENTE

Las **extensiones de lado del cliente** agregan funcionalidad al navegador web. Aunque existen en varias formas, las extensiones que más se encuentran comúnmente son:

- Aplicaciones externas.
- Java y JavaScript.
- ActiveX y VBScript.

Un **plug-in** es una aplicación externa que es invocada automáticamente por el navegador cuando es necesario. Como es *externa*, el plug-in es específico para el sistema operativo. El plug-in está asociado con un objeto de datos, generalmente usando la extensión de archivo, para permitir que el servidor web maneje en forma apropiada los datos que no están originalmente soportados. Por ejemplo, si uno de los componentes de la página es un documento PDF, el servidor web recibirá los datos, los reconocerá como objeto de “formato de documento portátil” y lanzará el Adobe Acrobat Reader para presentar el documento en la computadora cliente.

Como ya vimos, Java se ejecuta sobre el software del navegador web. Las aplicaciones de Java se compilan y guardan en el servidor web. (En muchos aspectos, Java se asemeja a C++.) Las llamadas a rutinas de Java se insertan dentro de la página de HTML. Cuando el navegador encuentra esta llamada, descarga las clases de Java (código) del servidor web y ejecuta ese código en la computadora cliente. La principal ventaja de Java es que hace posible que los desarrolladores desplieguen sus aplicaciones una vez y las ejecuten en muchos ambientes. (Para desarrollar aplicaciones web, la interoperabilidad es algo muy importante. Desafortunadamente, diferentes navegadores no son 100% interoperables, con lo cual se limita la portabilidad.)

JavaScript es un lenguaje de guiones (que hace posible ejecutar una serie de comandos o macros) que permite a autores web diseñar sitios interactivos. Como JavaScript es más sencillo para generarse que Java, es más fácil de aprender. El código de JavaScript está insertado en las páginas web. Se descarga con ellas y se activa cuando un evento específico tiene lugar, por ejemplo, un clic de ratón en un objeto o página que sea cargado del servidor en la memoria.

ActiveX es la alternativa de Microsoft a Java. ActiveX es una especificación para escribir programas que se ejecutarán dentro del navegador de Microsoft (Internet Explorer). Como ActiveX está orientado principalmente hacia aplicaciones para Windows, tiene baja portabilidad. ActiveX extiende el navegador web al agregar "controles" a las páginas web. (Ejemplos de esos controles son listas descendentes, una regla, un calendario y una calculadora.) Esos controles, descargados del servidor web cuando sea necesario, permiten manipular datos dentro del navegador. Los controles ActiveX pueden ser creados en varios lenguajes de programación; C++ y Visual Basic son los que más se usan. .NET de Microsoft permite una interoperabilidad más amplia de aplicaciones de ActiveX (por ejemplo ADO.NET) en diversos ambientes operativos.

VBScript es otro producto de Microsoft que se usa para extender la funcionalidad del navegador. VBScript se deriva de Visual Basic de Microsoft. Al igual que JavaScript, el código de VBScript está insertado en una página de HTML y se acciona al activar eventos como dar un clic en un vínculo.

Desde el punto de vista del desarrollador, usar rutinas que permitan validación de datos en el lado cliente es una absoluta necesidad. Por ejemplo, cuando se ingresan datos en una forma web y no se hace validación en el lado cliente, todo el conjunto de datos debe ser enviado al servidor web. Esa situación requiere que el servidor ejecute la validación de todos los datos, desperdiciando así valiosos ciclos de procesamiento del CPU. Por tanto, la validación de entrada de datos del lado cliente es uno de los requisitos básicos para las aplicaciones web. Casi todas las rutinas de validación de datos se hacen en Java, JavaScript, ActiveX o VBScript.

14.2.5 SERVIDORES DE APLICACIÓN WEB

Un **servidor de aplicación web** es una aplicación de middleware que expande la funcionalidad de los servidores web al vincularlos con una amplia gama de servicios, como bases de datos, sistemas de directorio y motores de búsqueda. El servidor de aplicación web también proporciona un ambiente consistente de tiempo de ejecución para aplicaciones web.

Los servidores de aplicación web pueden usarse para:

- Conectarse y consultar una base de datos desde una página web.
- Presentar datos de una base de datos en una página web, usando varios formatos.
- Crear páginas dinámicas de búsqueda de la web.
- Crear páginas web para insertar, actualizar y eliminar datos de una base de datos.
- Hacer cumplir integridad referencial en la lógica de un programa de aplicación.
- Usar consultas simples, anidadas y lógica de programación para representar reglas de negocios.

Los servidores de aplicación web contienen funciones como:

- Un ambiente integrado de desarrollo con administración y soporte de sesiones para variables de aplicación persistentes.
- Seguridad y autenticación de usuarios mediante ID y contraseñas.
- Lenguajes computacionales para representar y guardar lógica de negocios en el servidor de aplicación.
- Generación automática de páginas de HTML integrada con Java, JavaScript, VBScript, ASP, etcétera.
- Funciones de desempeño y tolerancia a fallas.
- Acceso a base de datos con capacidad de administrar transacciones.
- Acceso a múltiples servicios, por ejemplo, transferencias de archivos (FTP), conectividad de base de datos, e-mail y servicios de directorio.

Al tiempo de escribir esto, los servidores de aplicación web preferidos por usuarios incluyen ColdFusion/JRun de Adobe, WebSphere Application Server de IBM, WebLogic Server de Oracle, Fusion de NetObjects, Visual Studio .NET de Microsoft y WebObjects de Apple. Todos los servidores de aplicación web ofrecen la capacidad de conectar servidores web a múltiples fuentes de datos y otros servicios. Varían en términos del alcance de sus funciones disponibles, robustez, escalabilidad, facilidad de uso, compatibilidad con otras herramientas web, de bases de datos y extensión del ambiente de desarrollo.



CONTENIDO EN LÍNEA

Para ver y probar una interfaz de la web a la base de datos, consulte el **Apéndice J** en el sitio web Premium para este libro. Este apéndice lo guía para el proceso de crear y usar una sencilla interfaz de la web a la base de datos y da información más detallada sobre el desarrollo de bases de datos web con el middleware ColdFusion de Adobe.

Los sistemas de la generación actual suponen más que sólo el desarrollo de aplicaciones de bases de datos activadas por la web. También requieren aplicaciones capaces de comunicarse entre sí y con otros sistemas no basados en la web. Es evidente que los sistemas deben ser capaces de intercambiar datos en un formato basado en estándares. Éste es el papel de XML.

14.3 LENGUAJE DE MARCADO EXTENSIBLE

Internet ha originado nuevas tecnologías que facilitan el intercambio de datos de negocios entre socios y consumidores de negocios. Las compañías la están usando para crear nuevos tipos de sistemas que integran sus datos para aumentar eficiencia y reducir costos. El comercio electrónico hace posible que todo tipo de organizaciones vendan y compren productos y servicios en un mercado mundial de millones de usuarios. Las transacciones por internet, es decir, la venta de productos o servicios, puede tener lugar entre negocios (negocio a negocio, o B2B) o entre un negocio y un consumidor (negocio a consumidor, o B2C).

La mayor parte de las transacciones de comercio electrónico tiene lugar entre negocios. Como el comercio electrónico B2B integra procesos de negocios entre compañías, requiere la transferencia de información de negocios entre diferentes entidades. Pero la forma en que las empresas representan, identifican y usan datos tiende a diferir de manera importante de una a otra ("código de producto" vs "ID de artículo").

Hasta hace poco, la expectativa era que una orden de compra que viajara por la web estaría en la forma de un documento de HTML. La página web de HTML mostrada en el navegador web incluiría formateo y detalles del pedido. Las **etiquetas** de HTML describen la forma en que algo se ve en la página web, por ejemplo, tipo en letras negritas o estilo de encabezado y casi siempre vienen en pares para iniciar y terminar funciones de formato. Por ejemplo, las siguientes etiquetas encerradas en picocorchetes muestran la leyenda EN VENTA en fuente Arial negrita:

```
<strong><font face=Arial>EN VENTA</font></strong>
```

Si una aplicación desea obtener los datos del pedido de la página web, no hay forma fácil de extraerlos (por ejemplo, número del pedido, fecha, número de cliente, el artículo, la cantidad, el precio o detalles de pagos) de un documento de HTML. El documento de HTML puede sólo describir la forma de exhibir el pedido en un navegador web; no permite la manipulación de los elementos de datos del pedido, es decir, fecha, información de embarque, detalles de pago, información del producto, etc. Para resolver ese problema se ideó un nuevo lenguaje de marcado, conocido como Lenguaje de marcado extensible (XML, Extensible Markup Language).

El **Lenguaje de marcado extensible (XML)** es un metalenguaje que se usa para representar y manipular elementos de datos. XML está diseñado para facilitar el intercambio de documentos estructurados, por ejemplo, pedidos y facturas, a través de internet. El World Wide Web Consortium (W3C)¹ publicó en 1998 la primera definición de la norma XML 1.0. Esa norma preparó el escenario para dar a XML el atractivo práctico de ser una verdadera plataforma independiente del vendedor. Por tanto, no es de sorprender que XML se haya convertido rápidamente en la norma de intercambio de datos para aplicaciones de ventas por internet.

El metalenguaje XML permite la definición de noticias, por ejemplo <ProdPrice>, para describir los elementos de datos que se usan en un documento de XML. Esta capacidad de *extender* el lenguaje explica la X en el acrónimo XML; se dice que el lenguaje es *extensible*. El XML se deriva del Lenguaje de marcado General Estándar (SGML), una norma

¹ Se puede visitar la página web del W3C en www.w3.org para obtener información adicional acerca del trabajo realizado para desarrollar la norma XML.

- *Seguridad e integridad de la base de datos.* El DBA debe definir las políticas que rigen la seguridad e integridad. La seguridad de una base de datos es especialmente crucial. Las normas de seguridad deben estar definidas con toda claridad e impuestas en forma estricta. Los procedimientos de seguridad deben estar diseñados para manejar una multitud de situaciones para cerciorarse que los problemas de seguridad son mínimos. Aunque no hay sistema que pueda estar por completo seguro, deben diseñarse procedimientos de seguridad para satisfacer normas críticas. El creciente uso de interfaces de internet a bases de datos abre la puerta a nuevas amenazas de seguridad que son mucho más complejas y difíciles de manejar, que las que se encuentran con interfaces más tradicionales, internamente generadas y controladas. Por tanto, el DBA debe trabajar estrechamente con especialistas en seguridad en internet para cerciorarse que las bases de datos están debidamente protegidas de ataques lanzados en forma inadvertida o deliberada.
- *Respaldo y recuperación de una base de datos.* Los procedimientos de respaldo y recuperación de una base de datos deben incluir la información necesaria, para garantizar la correcta ejecución y administración de los respaldos.
- *Mantenimiento y operación de una base de datos.* Las operaciones diarias de un DBMS deben estar claramente documentadas. Los operadores deben mantener bitácoras de trabajo, escribir instrucciones y notas al operador. Esas notas son útiles para identificar las causas y soluciones de problemas. Los procedimientos operacionales también deben incluir información precisa respecto a procedimientos de respaldo y recuperación.
- *Capacitación del usuario final.* Debe establecerse un programa de capacitación, con la funcionalidad más avanzada posible dentro de la organización y deben especificarse con toda claridad los procedimientos que rijan la capacitación. El objetivo es indicar claramente quién hace qué, cuándo y cómo. Cada usuario final debe estar informado del tipo y magnitud de la metodología de capacitación disponible.

Los procedimientos y normas deben modificarse, al menos cada año, para mantenerlos actualizados y asegurar que la organización puede adaptarse rápidamente a cambios en el ámbito del trabajo. Naturalmente, la introducción de nuevo software del DBMS, el descubrimiento de violaciones de seguridad o integridad, la reorganización de la compañía, así como los cambios similares, requieren de la modificación de los procedimientos y normas.

Seguridad, privacidad e integridad de datos

La seguridad, privacidad e integridad de los datos de la base de datos son de la mayor preocupación para los DBA que administren instalaciones actuales de DBMS. La tecnología ha señalado el camino para una mayor productividad por medio de administración de la información. La tecnología también ha resultado en la distribución de datos en múltiples sitios, haciendo así más difícil mantener el control, seguridad e integridad de datos. La configuración de datos, en múltiples sitios, ha hecho imperativo que el DBA utilice mecanismos de seguridad e integridad proporcionados por el DBMS, para hacer cumplir políticas de administración de bases de datos definidas en la sección previa. Además, los DBA deben hacer equipo con expertos en seguridad de internet para formar mecanismos de seguridad y salvaguardar datos contra posibles ataques o acceso no autorizado. La sección 15.6 estudia con más detalle problemas de seguridad.

Respaldo y recuperación de datos

Cuando los datos no sean fácilmente disponibles, las compañías se arriesgan a pérdidas potencialmente cuantiosas. Por tanto, los procedimientos de respaldo y recuperación de datos son de importancia decisiva en todas las instalaciones de bases de datos. El DBA debe asegurar que los datos de la base de datos pueden ser recuperados por completo en caso de pérdida física de datos o pérdida de integridad de la base de datos.

La pérdida de datos puede ser parcial o total. La parcial es causada por una pérdida física de parte de la base de datos o cuando parte de la base de datos ha perdido integridad. Una pérdida total podría significar que la base de datos continúe existiendo, pero su integridad está enteramente perdida o que toda la base de datos esté físicamente perdida. En cualquier caso, los procedimientos de respaldo y recuperación son el seguro más barato que se pueda pagar por una base de datos.

La administración de la seguridad, integridad, respaldo y recuperación de una base de datos es tan crítica que numerosos departamentos de DBA han creado una posición llamada **oficial de seguridad de base de datos (DSO)**. El único trabajo del DSO es garantizar la seguridad e integridad de la base de datos. En organizaciones grandes, las actividades del DSO con frecuencia son calificadas como *administración de desastres*.

La **administración de desastres** incluye todas las actividades del DBA diseñadas para asegurar la disponibilidad de datos, después de un desastre físico o una falla de integridad de la base de datos. La administración de desastres comprende toda la planeación, organización, prueba de planes de contingencia de bases de datos y de procedimientos de recuperación. Las medidas de respaldo y recuperación pueden incluir al menos:

- *Respallos periódicos de datos y aplicaciones.* Algunos DBMS incluyen herramientas para asegurar el respaldo y recuperación de los datos de la base de datos. El DBA debe usarlas para hacer automáticas las tareas de respaldo y recuperación. Productos como DB2 de IBM permiten la creación de diferentes tipos de respaldo: completos, incrementales y concurrentes. Un **respaldo completo**, también conocido como **depósito de base de datos**, produce una copia completa de toda la base de datos. Un **respaldo incremental** produce un respaldo de todos los datos desde la última fecha de respaldo; un **respaldo concurrente** ocurre cuando el usuario está trabajando en la base de datos.
- *Identificación correcta de respaldo.* Los respaldos deben ser identificados claramente por medio de descripciones detalladas e información de fecha, haciendo posible así que el DBA asegure que se usen los respaldos correctos para recuperar la base de datos. El medio más común de respaldo ha sido tradicionalmente la cinta; el almacenamiento y marcado de cintas deben ser hechos de manera diligente por los operadores de la computadora y el DBA debe dar seguimiento a la actualización y ubicación de la cinta. No obstante, las organizaciones que sean lo suficiente grandes para contratar un DBA por lo general no usan discos compactos (CD) o DVD para respaldos de sus empresas. Otras soluciones de respaldo de reciente aparición incluyen equipos de respaldo ópticos y de discos. Estas soluciones de respaldo incluyen almacenamiento en línea basado en el *Network Attached Storage* (NAS) y el *Storage Area Networks* (SAN). Las soluciones de respaldo de empresas utilizan un método de respaldo de capas en el que los datos primero se respaldan a medios de disco rápido para almacenamiento intermedio y rápida restauración. Después, los datos son transferidos a cinta para almacenamiento en archivo.
- *Almacenamiento cómodo y seguro de respaldos.* Debe haber múltiples respaldos de los mismos datos y cada copia de respaldo debe ser guardada en un lugar diferente. Los lugares de almacenamiento deben incluir sitios dentro y fuera de la organización. (Mantener respaldos diferentes en el mismo lugar derrota al propósito de tener múltiples respaldos en el primer lugar.) Los lugares de almacenamiento deben estar debidamente preparados y pueden contar con cajas fuertes y bóvedas de seguridad a prueba de terremotos, así como controles de humedad y temperatura. El DBA debe establecer una política para responder a dos preguntas: 1) ¿Dónde han de guardarse los respaldos? 2) ¿Cuánto tiempo han de guardarse los respaldos?
- *Protección física de hardware y software.* La protección puede incluir el uso de instalaciones cerradas con acceso restringido, así como preparación de los sitios de la computadora para proporcionar acondicionamiento de aire, energía eléctrica para el respaldo y protección contra incendio. La protección física también incluye la provisión de una computadora y DBMS para respaldo para uso en caso de emergencia. Por ejemplo, cuando el huracán Katrina golpeó la costa del Golfo de México en 2005, Nueva Orleans sufrió la destrucción casi total de su infraestructura de comunicaciones. La tormenta sirvió como “llamada de atención” para numerosas organizaciones e instituciones educacionales, que no tenían planes adecuados para recuperarse del desastre para ese nivel de interrupción del servicio.
- *Control personal de acceso al software de una instalación de base de datos.* Se pueden usar símbolos de desafío/respuesta para contraseñas de multiniveles y privilegios, así como hardware y software para identificar debidamente a usuarios autorizados de recursos.
- *La cobertura de seguro para los datos en la base de datos.* El oficial DBA o de seguridad deben garantizar una política de seguros para dar protección financiera en el caso de una falla de la base de datos. El seguro puede ser costoso, pero es menos caro que el desastre creado por una pérdida masiva de datos.

Dos puntos adicionales merecen la pena ser citados.

- Los planes para recuperación y contingencia de datos deben ser probados, evaluados perfectamente y practicados con frecuencia. Las prácticas llamadas “simulacro de incendio” no deben menospreciarse y requieren de apoyo e imposición de parte de la alta dirección de la empresa.
- No es probable que un programa de respaldo y recuperación abarque todos los componentes de un sistema de información, por lo cual es apropiado establecer prioridades respecto a la naturaleza y extensión del proceso de recuperación de datos.

Distribución y uso de datos

Los datos son útiles sólo cuando llegan a los usuarios apropiados en forma oportuna. El DBA es responsable de asegurar que los datos estén distribuidos a las personas apropiadas, en el momento oportuno y en el formato apropiado. Las ta-