FUNDAMENTOS DE

Bases de datos

- Aprende técnicas de diseño y administración de bases de datos universales.
- Trabaja con SQL, Microsoft Access y Oracle.
- Asegura tu base de datos, conecta aplicaciones e integra documentos XML.

Andy Oppel

Mc Graw Hill

BIBLIOTECA DEL IT PRO

Fundamentos de
Bases de datos

Sobre el autor

Andrew J. (Andy) Oppel se graduó con orgullo de The Boys' Latin School of Maryland y la Transylvania University (Lexington, Kentucky), donde obtuvo una licenciatura en ciencias de la computación en 1974. Desde entonces, ha colaborado en una amplia variedad de cargos relacionados con la tecnología de información, entre ellos programador, programador/analista, arquitecto de sistemas, administrador de proyectos, administrador principal de base de datos, administrador de grupo de bases de datos, consultor, diseñador de bases de datos, modelador de datos y arquitecto de datos. Además, ha sido instructor de tiempo parcial en el área de extensión académica de la University of California, Berkeley, durante más de 20 años y recibió el premio Honored Instructor en el año 2000. Su actividad docente incluye la preparación de tres cursos para extensión académica en dicha universidad: "Concepts of database management systems", "Introduction to relational database management systems" y "Data modeling and database design". También obtuvo su certificación de Oracle 9*i* Database Associate en 2003. En la actualidad se desempeña como modelador principal de datos de Blue Shield en California. Además de su inclinación por los sistemas de computadoras, Andy tiene gusto por la música (toca guitarra y canta), la radio de aficionados (subdirector de la división Pacific, de la American Radio Relay League) y el fútbol soccer (instructor de árbitros, U. S. Soccer).

Andy ha diseñado e implementado cientos de bases de datos para una amplia variedad de aplicaciones, entre ellas investigación médica, operaciones bancarias, seguros, fabricación de prendas de vestir, telecomunicaciones, comunicaciones inalámbricas y recursos humanos. Es autor de *Databases demystified* (McGraw-Hill Professional, 2004) y *SQL demystified* (McGraw-Hill Professional, 2005), y es coautor de *Fundamentos de SQL* (McGraw-Hill Educación, 2010). Su experiencia con productos de bases de datos incluye IMS, DB2, Sybase ASE, Microsoft SQL Server, Microsoft Access, MySQL y Oracle (versiones 7, 8, 8*i*, 9*i*, y 10*g*).

Si tiene algunos comentarios, por favor comuníquese con el autor en andy@andyoppel.com.

Acerca del editor técnico

Todd Meisner ha colaborado en la utilización de tecnología de Microsoft durante más de diez años. Ha sido editor técnico de más de 50 libros con temas que cubren desde SQL Server hasta .NET Framework. Además de la edición de libros técnicos, colabora como director asistente de servicios de computación en la Ball State University, en Muncie, Indiana. Vive en la zona centro de Indiana con su esposa Kimberly, y cuatro hijos estupendos. Comuníquese con Todd en tmeisner@sycamoresolutions.com.

Fundamentos de **Bases de datos**

Andy Oppel

Traducción

Miguel Ángel Martínez Sarmiento

Traductor profesional

MÉXICO • BOGOTÁ • BUENOS AIRES • CARACAS • GUATEMALA • MADRID
NUEVA YORK • SAN JUAN • SANTIAGO • SÃO PAULO • AUCKLAND • LONDRES • MILÁN
MONTREAL • NUEVA DELHI • SAN FRANCISCO • SINGAPUR • ST. LOUIS • SIDNEY • TORONTO

Director editorial: Fernando Castellanos Rodríguez
Editor: Miguel Ángel Luna Ponce
Supervisor de producción: Zeferino García García

FUNDAMENTOS DE BASES DE DATOS

Corporativo Punta Santa Fe,
Prolongación Paseo de la Reforma 1015, Torre A
Piso 17, Colonia Desarrollo Santa Fe,
Delegación Álvaro Obregón
C.P. 01376, México, D.F.
Miembro de la Cámara Nacional de la Industria Editorial Mexicana, Reg. Núm. 736

ISBN: 978-607-15-0254-4

Translated from the 1st English edition of
Databases: A Beginner's Guide
By: Andrew Oppel

ISBN: 978-0-07-160846-6

1234567890 109876543210

Impreso en México *Printed in Mexico*

Contenido

Agradecimientos

Numerosas personas participaron en la preparación de *Fundamentos de Bases de datos*; y no me creo capaz de recordar por su nombre a todas ellas. En primer lugar, debo mencionar a los editores y el personal de McGraw-Hill que aportaron incontables horas de apoyo para este proyecto. En especial deseo agradecer a la directora editorial Wendy Rinaldi, quien me ha proporcionado sugerencias e inspiración durante la preparación de todos mis libros. En realidad, fue Wendy quien me inició como autor de McGraw-Hill. También deseo agradecer a Lisa Theobald por el excelente cuidado de la edición y a todos los demás editores, lectores de pruebas, especialistas en índices, diseñadores, ilustradores, y otros participantes. En especial quiero agradecer a Todd Meisner, el editor técnico, por su atención a los detalles y sus valiosas opiniones durante todo el proceso de edición. Por último, agradezco a mi familia su apoyo y comprensión por aceptar que incluyera los horarios de redacción de este libro en una agenda de negocios ya bastante saturada.

Introducción

Hace 35 años, sólo existían bases de datos en laboratorios de investigación especiales, donde los científicos computacionales buscaban modos de hacerlas eficientes y útiles, y publicaban sus hallazgos en numerosos documentos de investigación. En la actualidad, las bases de datos son una parte muy difundida de la industria de la tecnología de la información (TI) y los negocios en general. Cada día empleamos bases de datos de manera directa e indirecta; transacciones bancarias, reservaciones de viajes, listas de empleados, búsquedas en sitios Web, compras en línea y directas, y muchas otras transacciones se registran y son atendidas mediante bases de datos.

Tal como ocurre con muchas tecnologías de rápido crecimiento, los estándares de la industria se han quedado atrás en relación con el desarrollo de la tecnología de base de datos, lo que genera numerosos productos comerciales, cada uno con la visión específica de un vendedor de software. Además, han surgido varios modelos diferentes de bases de datos y, entre ellos, el más frecuente es el relacional. *Fundamentos de Bases de datos* examina los principales modelos de bases de datos, entre ellos jerárquico, de red, relacional, orientado a objetos y de objetos-relacional. No obstante, este libro se concentra en los modelos relacional y de objetos-relacional, porque son la corriente principal en la industria de TI y así se mantendrán en el futuro previsible.

El reto más significativo al implementar una base de datos consiste en diseñar correctamente su estructura. Si no se comprende al detalle la base de datos que se pretende resolver, y no se conocen las mejores prácticas para organizar los datos requeridos, la base de datos implementada se vuelve una pesada bestia que requiere atención constante.

Fundamentos de Bases de datos se concentra en transformar los requisitos en un modelo funcional de datos, con especial atención en un proceso llamado *normalización*, que se ha

demostrado como una técnica eficaz para diseñar bases de datos relacionales. En realidad, la normalización se puede aplicar con éxito a otros modelos de bases de datos. Y, de acuerdo con la noción de que no es posible diseñar un automóvil cuando no se ha conducido uno, se presenta al lector el lenguaje de consulta estructurada (SQL, Structured Quero Language) para que pueda "conducir" una base de datos antes de ahondar en los detalles de diseñar una.

Me he apoyado en mi extensa experiencia como diseñador, administrador e instructor de bases de datos para ofrecerle esta guía de autoayuda hacia el fascinante y complejo mundo de la tecnología de bases de datos. Se incluyen ejemplos con Microsoft Access y Oracle. En lo posible, en las figuras de ejemplo se utilizan las bases de datos de muestra de estos vendedores (Northwind de Microsoft Access y Human Resources de Oracle) para que pruebe los ejemplos directamente en su sistema de cómputo. Al final de cada capítulo se presenta una autoevaluación para que refuerce su aprendizaje.

Quién debe leer este libro

Fundamentos de Bases de datos es conveniente para cualquier persona que busque captar los fundamentos de diseño y administración de una base de datos, ya sea para uso personal o profesional. El libro se diseñó específicamente para quienes desconocen o saben poco de la tecnología de bases de datos; sin embargo, también será de utilidad para quienes necesitan un repaso sobre la normalización y el diseño y la administración de una base de datos. No importa si es un desarrollador experimentado, tiene cierta experiencia en ese proceso, es administrador de una base de datos o desconoce todo de programación y bases de datos, *Fundamentos de Bases de datos* aporta bases sólidas para cualquier persona que busque aprender más sobre la tecnología de las bases de datos. En realidad, este libro será de utilidad para cualquiera de las siguientes personas, con el fin de comprender y utilizar bases de datos:

- El novato que desconoce el diseño de una base de datos y la programación en SQL.
- El analista o administrador que pretende comprender mejor cómo diseñar, implementar y consultar bases de datos.
- El administrador de base de datos que quiere aprender más acerca de su diseño.
- El profesional de soporte técnico o ingeniero de pruebas o aseguramiento de la calidad que debe realizar consultas *ad hoc* contra bases de datos SQL.
- El desarrollador Web que crea aplicaciones que requieren bases de datos para persistencia de datos.
- El programador de lenguajes de tercera generación (3GL, Third Generation Languages) que incrusta SQL en el código fuente de una aplicación.
- Cualquier otra persona que quiera aprender a diseñar bases de datos y escribir código SQL para crear y consultar bases de datos dentro de un RDBMS.

No importa en cuál categoría se ubica usted, debe recordar que el libro está dirigido a quien busca conocer técnicas comunes de diseño de base de datos que funcionen en cualquier base de datos, no al producto de un vendedor en particular. Esto le permite aplicar las habilidades que aprenderá en este libro a situaciones reales, sin estar limitado a productos comunes. Claro que todavía necesitará conocer cómo se implementa en las bases de datos el producto con el que trabaja, en particular los dialectos de SQL, pero con los fundamentos que obtendrá en estas páginas, podrá pasar de un RDBMS al siguiente y, no obstante, tener una comprensión firme de la teoría del diseño de bases de datos. Como resultado, descubrirá que este libro es un recurso útil para quien desconozca las bases de datos, sobre todo las relacionales, sin tomar en cuenta el producto utilizado. Podrá adaptar fácilmente su conocimiento al RDBMS específico.

Lo que cubre el libro

Fundamentos de Bases de datos se divide en tres partes. En la parte I se presenta los conceptos básicos de bases de datos y explica cómo crear y consultar los objetos en el interior de su base de datos mediante SQL. En la parte II se ofrecen los fundamentos del desarrollo de una base de datos, entre ellos su ciclo de vida, el diseño lógico mediante un proceso de normalización, la transformación del diseño lógico a una base de datos física, y el modelado de datos y de procesos. La Parte III se concentra en la implementación de una base de datos, con hincapié en la seguridad, en temas avanzados de bases de datos como el procesamiento analítico en línea (OLAP) y la integración de objetos y documentos de XML en la base de datos, lo que permite ampliar lo aprendido en las partes I y II. Además de las tres partes, *Fundamentos de Bases de datos* contiene apéndices que incluyen las respuestas a las preguntas de autoevaluación y las soluciones a los ejercicios Pruebe esto que aparecen en todo el libro.

Descripción del contenido

El resumen siguiente describe el contenido del libro y muestra cómo se divide en capítulos orientados a tareas:

Parte I: Conceptos de bases de datos

En la parte I se presentan los conceptos básicos de una base de datos y se explica cómo crear y consultar los objetos dentro de ella mediante SQL.

Capítulo 1: Fundamentos de bases de datos En este capítulo se presentan conceptos y definiciones fundamentales acerca de las bases de datos, entre ellos las propiedades comunes, los modelos preferidos, una breve historia de las bases de datos y la razón primordial para concentrarse en el modelo relacional.

Capítulo 2: Exploración de los componentes de una base de datos relacional En este capítulo se exploran los componentes conceptual, físico y lógico que forman el modelo relacional. El *diseño conceptual de una base de datos* requiere el estudio y modelado de los

datos de manera independiente de la tecnología. El *diseño lógico de una base de datos* es el proceso de traducir, o *ubicar*, el diseño conceptual en un diseño lógico que se adapte al modelo de base de datos elegido (relacional, orientado a objetos, de objetos-relacional, etc.). El paso final del diseño es el *diseño físico de una base de datos*, que incluye la ubicación del diseño lógico en uno o más diseños físicos, cada uno adaptado al DBMS que administrará la base de datos y el sistema de computadoras específico en que funcionará la base de datos.

Capítulo 3: Consultas a bases de datos con formularios En este capítulo se ofrece un compendio de la preparación y la ejecución de consultas en una base de datos por medio de la herramienta de consulta mediante formularios de Microsoft Access, que proporciona los fundamentos para la consulta a bases de datos para la teoría de diseño de bases de datos que se presenta en los capítulos posteriores.

Capítulo 4: Introducción a SQL En este capítulo se presenta SQL, que se ha convertido en el lenguaje universal de las bases de datos relacionales que aceptan casi todos los DBMS en la vida moderna. Es evidente que su amplia aceptación se debe al tiempo y esfuerzo dedicados a desarrollar las funciones y las normas de ese lenguaje, que permiten trasladar fácilmente el SQL entre diferentes productos de RDBMS.

Parte II: Desarrollo de una base de datos

En la parte II se proporcionan los fundamentos para el desarrollo de una base de datos, entre ellos el ciclo de vida de una base de datos, el diseño lógico mediante un proceso de normalización, la transformación del diseño lógico en una base de datos física, y el modelado de datos y procesos.

Capítulo 5: El ciclo de vida de una base de datos En este capítulo se presenta la estructura en que ocurre el diseño de una base de datos, un útil precursor de los detalles de este proceso. *Ciclo de vida* de una base de datos (o sistema computacional) es el término utilizado para todos los eventos que ocurren entre el momento en que se reconoce la necesidad de una base de datos, seguido por el desarrollo y el despliegue, y que concluye el día en que se retira del servicio.

Capítulo 6: Diseño de una base de datos mediante normalización En este capítulo, aprenderá a efectuar el diseño lógico de una base de datos, mediante un proceso llamado *normalización*. En cuanto a la comprensión de la tecnología de una base de datos relacional, éste es el tema más importante del libro, porque la normalización le enseña cómo organizar mejor sus datos en tablas.

Capítulo 7: Modelado de datos y de procesos En este capítulo se analizan con mayor detalle los diagramas de entidades-relaciones (ERD) y el modelado de datos. En la segunda parte del capítulo se incluye una panorámica de alto nivel de los conceptos de diseño del proceso y técnicas de diagramación.

Capítulo 8: Diseño físico de una base de datos Este capítulo se dedica al trabajo de diseño físico de una base de datos, que consiste en transformar el diseño lógico en uno o más diseños físicos de una base de datos.

Parte III: Implementación de una base de datos

La parte III se concentra en la implementación de una base de datos, con hincapié en la seguridad, en temas avanzados de bases de datos como el procesamiento analítico en línea (OLAP) y la integración de objetos y documentos de Lenguaje de marcado extensible (XML) en la base de datos; esto permite ampliar lo aprendido en las partes I y II.

Capítulo 9: Conexión de bases de datos al mundo exterior Este capítulo comienza con un repaso de la evolución de los modelos de despliegue de una base de datos; es decir, los modos en que las bases de datos se han conectado con sus usuarios y los otros sistemas de computadoras dentro de la *infraestructura* de cómputo de una empresa (la estructura interna que organiza todos los recursos de cómputo de una empresa, entre ellos bases de datos, aplicaciones, hardware y una red). Luego se analizan los métodos empleados para conectar las bases de datos a aplicaciones que usan un navegador Web como principal interfaz de usuario, que es el modo en que se crean muchos sistemas de aplicaciones modernos. Concluye con una revisión de los métodos actuales para conectar bases de datos y aplicaciones, por ejemplo mediante conexiones ODBC (para casi todos los lenguajes de programación) y diversos métodos para enlazar bases de datos y aplicaciones escritas en Java (un lenguaje orientado a objetos de uso frecuente).

Capítulo 10: Seguridad de una base de datos En este capítulo se expone la necesidad de la seguridad, las consideraciones de seguridad para implementar servidores de base de datos, los temas relacionados con la seguridad de los clientes que consultan esos servidores, además de los métodos para implementar consultas seguras a la base de datos, y concluye con un análisis de monitoreo y auditoría de la seguridad.

Capítulo 11: Implementación de las bases de datos En este capítulo se cubren algunas consideraciones relacionadas con el desarrollo de las aplicaciones que emplea el sistema de base de datos. Entre ellas están el procesamiento mediante cursores, la administración de transacciones, la afinación del desempeño y el control de cambios.

Capítulo 12: Bases de datos para procesamiento analítico en línea En este capítulo se presentan los conceptos de bases de datos para procesamiento analítico, como almacenes y mercados de datos, un resumen de la minería de datos y otras técnicas de análisis, junto con las variaciones de diseño que requieren estos tipos de bases de datos.

Capítulo 13: Integración de documentos y objetos XML en bases de datos En este capítulo se exploran varias maneras de integrar contenido en XML y objetos en las bases de datos.

Parte IV: Apéndices

Los apéndices incluyen las respuestas a las preguntas de autoevaluación y las soluciones a los ejercicios Intente esto que aparecen en todo el libro.

Apéndice A: Respuestas a las autoevaluaciones Este apéndice proporciona las respuestas a las preguntas de autoevaluación presentadas al final de cada capítulo.

Apéndice B: Soluciones a los ejercicios Intente esto Este apéndice contiene las soluciones, entre ellas diagramas y código SQL aplicable, para los ejercicios Pruebe esto que aparecen en casi todos los capítulos del libro.

Contenido de los capítulos

Como se aprecia, *Fundamentos de Bases de datos* está organizado en capítulos. Cada capítulo se concentra en un conjunto de habilidades y conceptos importantes y contiene los antecedentes necesarios para comprender los conceptos, además de las habilidades requeridas para aplicar estos conceptos. Cada uno contiene elementos adicionales que ayudan a comprender mejor la información cubierta en ese capítulo:

Pregunta al experto

Cada capítulo contiene una o dos secciones Pregunta al experto, que responden preguntas que pueden surgir en relación con la información presentada en el capítulo.

Autoevaluación

Cada capítulo concluye con una Autoevaluación, un conjunto de preguntas para comprobar que asimiló la información y las habilidades presentadas en el capítulo. El apéndice A contiene las respuestas.

Ejercicios Pruebe esto

Casi todos los capítulos contienen uno o dos ejercicios Pruebe esto que le permiten aplicar la información aprendida en el capítulo. Cada ejercicio se divide en pasos que lo conducen por el proceso de concluir una tarea específica. En donde corresponde, los ejercicios incluyen archivos relacionados que puede descargar de nuestro sitio Web en www.mcgraw-hill-educacion.com. Haga una búsqueda por título y autor o ISBN. Haga clic en la portada de la obra y luego en el vínculo Descargue el código en el lado izquierdo de la página. Los archivos incluyen blocs de notas con las instrucciones de SQL o los diagramas utilizados en el ejercicio *Pruebe esto* (en inglés).

Para completar muchos de los ejercicios *Pruebe esto* del libro, necesita tener acceso a un RDBMS que le permita introducir y ejecutar instrucciones SQL de manera interactiva. Si accede a un RDBMS mediante una red, consulte al administrador de la base de datos para comprobar que inicia sesión con las credenciales necesarias para crear una base de datos y un esquema. Es posible que necesite permisos especiales para crear estos objetos. Verifique también si debe incluir parámetros específicos al crear la base de datos (por ejemplo, el tamaño del archivo de registro), y si existen restricciones sobre los nombres que puede usar o de otro tipo. No olvide la documentación adecuada antes de trabajar con cualquier producto de base de datos.

Parte I

Conceptos de bases de datos

Capítulo 1

Fundamentos de bases de datos

Habilidades y conceptos clave

- Propiedades de una base de datos
- Modelos prevalecientes de bases de datos
- Breve historia de las bases de datos
- ¿Por qué concentrarse en el modelo relacional?

Este capítulo introduce conceptos y definiciones fundamentales relacionados con las bases de datos, entre ellos sus propiedades comunes, los modelos frecuentes, una breve historia de las bases de datos y la razón de concentrarse en el modelo relacional.

Propiedades de una base de datos

Una *base de datos* es un conjunto de elementos de datos interrelacionados, administrados como unidad. Esta definición es deliberadamente amplia porque existe mucha variación entre los diferentes vendedores de software que ofrecen sistemas de bases de datos. Por ejemplo, Microsoft Access pone toda la base de datos en un solo archivo, de modo que una base de datos de Access puede definirse como el archivo que contiene los elementos de datos. Oracle Corporation define su base de datos como un conjunto de archivos físicos administrados por una instancia de su producto de software de base de datos. Una *instancia* es una copia del software de base de datos que se ejecuta en la memoria. Microsoft SQL Server y Sybase Adaptive Server Enterprise (ASE) definen una base de datos como un conjunto de elementos de datos que tienen un propietario común, y varias bases de datos suelen ser administradas por una sola instancia del software. Todo esto puede ser muy confuso si trabaja con varios productos; por ejemplo, una base de datos que se apega a la definición de Microsoft SQL Server o Sybase ASE es exactamente lo que Oracle Corporation llama un *esquema*.

Un *objeto de base de datos* es una estructura de datos con nombre que se guarda en una base de datos. Los tipos específicos de objetos permitidos varían de un vendedor a otro, y de un modelo a otro. El *modelo de base de datos* alude al modo en que una base de datos organiza su contenido para representar el mundo real. Los modelos más comunes de base de datos son presentados en la sección “Modelos frecuentes de bases de datos”, más adelante en el capítulo.

Un *archivo* es un conjunto de registros relacionados que un sistema operativo guarda como unidad. Debido a las definiciones desafortunadamente semejantes de *archivos* y *bases*

de datos, ¿cómo podemos diferenciarlos? Varios vendedores del sistema operativo Unix llaman "bases de datos" a sus archivos de contraseñas, pero los expertos se apresuran a señalar que en realidad no son bases de datos. Es evidente que requerimos más rigor en nuestras definiciones. La respuesta estriba en comprender ciertas características o propiedades que poseen las bases de datos y que no se encuentran en los archivos comunes, entre ellas:

- Control mediante un sistema de administración de bases de datos (DBMS, DataBase Management System).
- Abstracción de capas de datos.
- Independencia física de los datos.
- Independencia lógica de los datos.

Estas propiedades se analizan en las secciones siguientes.

El sistema de administración de bases de datos

El *sistema de administración de bases de datos (DBMS)* es el software proporcionado por el vendedor de la base de datos. Productos de software como Microsoft Access, Oracle, Microsoft SQL Server, Sybase ASE, DB2, Ingres y MySQL son DBMS. Si le parece extraño que se empleen las siglas DBMS en lugar de DMS, recuerde que, en inglés, el término *base de datos* (database) originalmente se escribía como dos palabras, y por convención se convirtió en una sola.

El DBMS ofrece todos los servicios básicos requeridos para organizar y conservar una base de datos, entre ellos:

- Mover los datos de archivos de datos físicos, según sea necesario.
- Administrar la opción de que varios usuarios consulten datos de manera concurrente, e incluir medidas que eviten que las actualizaciones simultáneas tengan conflictos entre sí.
- Controlar las transacciones para que los cambios en la base de datos de cada transacción sean una unidad de trabajo tipo todo o nada. En otras palabras, si la transacción tiene éxito, todos los cambios se registran en la base de datos; si la transacción fracasa, ninguno de los cambios se registra.
- Permitir un *lenguaje de consulta*, que es un sistema de comandos empleado por el usuario de la base de datos para recuperar sus datos.
- Proporcionar medidas para respaldar la base de datos y recuperarla después de una falla.
- Aportar mecanismos de seguridad para evitar la consulta y modificación no autorizadas de los datos.

Pregunta al experto

P: **He escuchado que se utiliza el término "banco de datos". ¿Cuál es la diferencia entre un banco y una base de datos?**

R: Un banco y una base de datos son lo mismo. *Banco de datos* es sólo un término más antiguo utilizado por los científicos que desarrollaron los primeros sistemas de base de datos. En realidad, el término *banco de datos* todavía se utiliza en algunos idiomas, como *banco de dados* en portugués.

Capas de abstracción de los datos

Las bases de datos son únicas por su capacidad para presentar a varios usuarios vistas propias y diferentes de los datos, al tiempo que conservan una sola vez los datos relacionados. En conjunto, se denominan *vistas de usuarios*. En este contexto, un *usuario* es cualquier persona o aplicación que se registra en la base de datos con el propósito de guardar datos, recuperarlos, o ambas opciones. Una *aplicación* es un conjunto de programas de computación diseñado para resolver un problema de negocios específico, como un sistema de recepción de pedidos, uno de procesamiento de nóminas o uno contable.

Cuando se utiliza una hoja de cálculo electrónica, como Microsoft Excel, todos los usuarios deben compartir una vista común de los datos, y esa vista debe coincidir con la manera en que los datos se guardan físicamente en el archivo correspondiente. Si un usuario oculta algunas columnas en una hoja de cálculo, reorganiza las filas y guarda la hoja de cálculo, el siguiente usuario que abra la hoja de cálculo verá los datos de la manera en que los guardó el primer usuario. Por supuesto, una opción es que cada usuario guarde su propia copia en archivos físicos separados, pero después cuando un usuario realiza actualizaciones, se desactualizan los datos de los demás usuarios. Los sistemas de bases de datos presentan a cada usuario una vista de los mismos datos, pero las vistas se pueden *ajustar* a las necesidades de los usuarios individuales, aunque todas provengan de una copia de los datos guardada de manera común. Debido a que las vistas no guardan los datos reales, reflejan automáticamente cualquier cambio realizado en los objetos relacionados de la base de datos. Todo esto es posible mediante las *capas de abstracción*, que se observan en la figura 1-1.

La arquitectura presentada en la figura 1-1 fue desarrollada en primer lugar por ANSI/SPARC (American National Standards Institute/Standards Planning and Requirements Committee) en la década de 1970 y pronto se convirtió en la base para casi todos los esfuerzos de investigación y desarrollo posteriores. Casi todos los DBMS modernos siguen esta arquitectura, integrada por tres capas primarias: física, lógica y externa. La arquitectura original incluía una capa conceptual, que se ha omitido aquí porque ningún vendedor de bases de datos modernas la implementa.

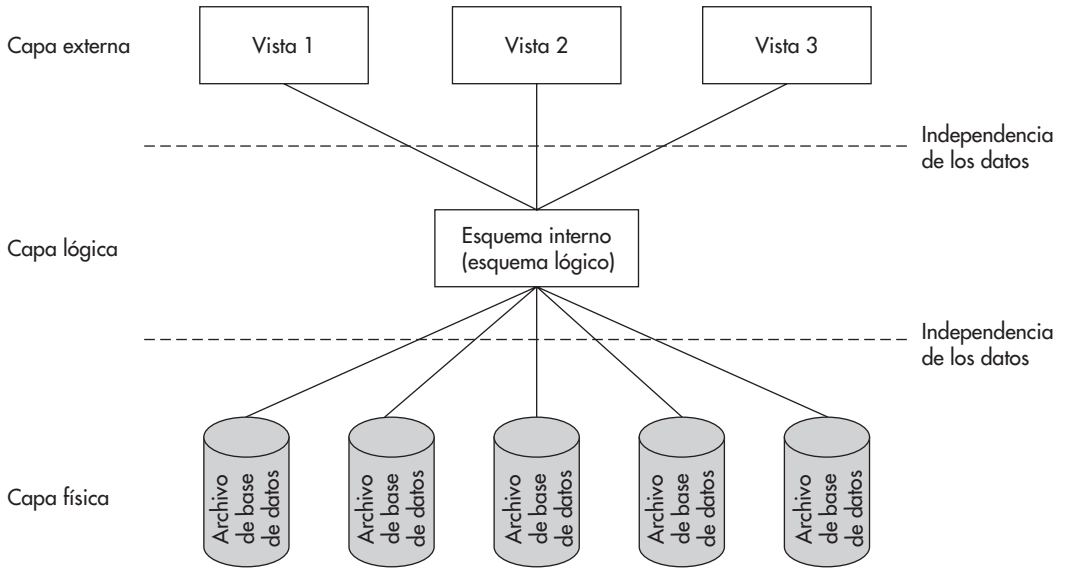


Figura 1-1 Capas de abstracción de una base de datos.

La capa física

La *capa física* incluye los archivos que contienen toda la información de la base de datos. Casi todos los DBMS modernos permiten que la base de datos se guarde en varios archivos, que se suelen distribuir en varias unidades de disco físicas. Con esta distribución, las unidades de disco trabajan en paralelo para ampliar al máximo el desempeño. Microsoft Access es una excepción notable entre los DBMS utilizados como ejemplo en este libro, porque almacena la base de datos completa en un solo archivo físico. Si bien la simplificación de una base de datos sirve en una computadora personal de un solo usuario, esta distribución limita la capacidad del DBMS para crecer y alojar muchos usuarios concurrentes, lo que hace que la base de datos sea una solución inadecuada como sistema para una empresa grande. Para ser justos, Microsoft Access no fue diseñado para ser un DBMS sólido de nivel empresarial. No se le ha incluido en los análisis del libro porque compita con productos como Oracle y SQL Server, sino porque es un estupendo ejemplo de un DBMS personal con una interfaz de usuario que hace fácil y divertido aprender los conceptos de una base de datos.

El usuario de una base de datos no necesita comprender cómo se guardan los datos dentro de los archivos ni cuáles archivos contienen los elementos de datos que le interesan. En casi todas las organizaciones, un técnico conocido como *administrador de base de datos* (DBA, DataBase Administrator) maneja los detalles de instalación y configuración del software y los archivos de la base de datos que permiten presentarla a los usuarios. El DBMS funciona con el sistema operativo de la computadora para ilustrar los archivos de datos de manera automá-

tica, lo que incluye todas las operaciones de abrir, cerrar, leer y escribir archivos. No debe ser obligatorio que el usuario haga referencia a los archivos físicos de datos al utilizar la base de datos, lo que está en marcado contraste con las hojas de cálculo y los procesadores de textos, en que el usuario debe guardar conscientemente los documentos y elegir nombres de archivo y lugares de almacenamiento. Muchos de los DBMS para computadoras personales son excepciones a este dogma, porque se requiere que el usuario localice y abra un archivo físico como parte del proceso de inicio de sesión en el DBMS. Por el contrario, con DBMS de nivel empresarial (como Oracle, Sybase ASE, Microsoft SQL Server y MySQL), los archivos físicos son administrados automáticamente y nunca es necesario que el usuario haga referencia a ellos al utilizar la base de datos.

La capa lógica

La *capa lógica* (o *modelo lógico*) está formada por las primeras dos *capas de abstracción* en la base de datos: la capa física tiene una existencia concreta en los archivos del sistema operativo, mientras que la capa lógica sólo existe como estructuras abstractas de datos integradas en la capa física, según se requiera. El DBMS transforma los datos de los archivos en una estructura común. En ocasiones, a esta capa se le denomina *esquema*, término utilizado para el conjunto de todos los elementos de datos guardados en una base de datos específica o que pertenecen a un usuario en particular. Dependiendo del DBMS, esta capa contiene un conjunto de tablas bidimensionales, estructura jerárquica similar al organigrama de una compañía, o alguna otra estructura. En la sección "Modelos frecuentes de bases de datos", presentada en páginas posteriores de este capítulo, se describen con mayor detalle las estructuras posibles.

La capa externa

La *capa externa* (o *modelo externo*) es la segunda capa de abstracción de la base de datos. Esta capa está formada por las vistas de usuarios analizadas antes, a las que en conjunto se les denomina *esquema secundario*. En esta capa, los usuarios (los programas de aplicaciones y las personas) que la consultan se conectan y plantean consultas contra la base de datos. Lo ideal es que sólo el DBA administre las capas física y lógica. El DBMS controla la transformación de los elementos seleccionados de una o más estructuras de datos en la capa lógica para formar la vista de cada usuario. Las vistas de usuario de esta capa se pueden redefinir y almacenar en la base de datos para reutilización, o pueden ser elementos temporales creados por el DBMS para contener los resultados de una sola consulta *ad hoc*, hasta que ya no sean necesarios para el usuario. Una consulta *ad hoc* no se prepara con anticipación y no es probable que se reutilice. Las vistas se analizan con mayor detalle en el capítulo 2.

Independencia física de los datos

La capacidad para modificar la estructura de los archivos físicos de una base de datos sin afectar a los usuarios y los procesos existentes es la *independencia física de los datos*. Como se aprecia en la figura 1-1, la separación entre las capas física y lógica aporta la independen-

cia física de los datos en un DBMS. Es esencial que se comprenda que la independencia física de los datos no es una propiedad que "se tiene o no se tiene", sino una en que un DBMS específico puede tener una independencia de datos mayor o menor que otro. La medida, también llamada *grado* de independencia física de los datos, representa la cantidad de cambios que se pueden hacer al sistema de archivos sin afectar a la capa lógica. Antes de los sistemas que ofrecen independencia de los datos, hasta el cambio más insignificante en el modo en que se guardaban los datos requería que el personal de programación hiciera cambios en todos los programas que utilizaban los datos, lo que era un proceso costoso y prolongado.

Todas las computadoras modernas tienen cierto grado de independencia física de los datos. Por ejemplo, una hoja de cálculo en una computadora personal seguirá funcionando adecuadamente si se copia de un disco duro a un disquete o a una memoria USB portátil. El hecho de que el desempeño (la velocidad) de estos dispositivos varíe mucho no es lo esencial, sino el hecho de que los dispositivos tienen una construcción física completamente diferente y, no obstante, el sistema operativo de la computadora personal maneja automáticamente las diferencias y presenta los datos del archivo para la aplicación (es decir, el programa de hoja de cálculo, como Microsoft Excel) y por lo tanto al usuario, exactamente de la misma manera. Sin embargo, en casi todos los sistemas personales, el usuario todavía debe recordar dónde colocó el archivo para poder localizarlo cuando lo necesite.

Los DBMS aumentaron en gran medida la independencia física de los datos ofrecida por las computadoras, porque permiten a los usuarios de una base de datos consultar *objetos* (por ejemplo, las tablas en un DBMS relacional) sin tener que hacer referencia a los archivos de datos físicos. El *catálogo* del DBMS guarda las definiciones de objetos y conserva un registro del lugar físico donde se guardan. Éstos son algunos ejemplos de los cambios físicos que pueden hacerse de manera independiente de los datos:

- Mover un archivo de base de datos de un dispositivo o de un directorio a otro.
- Dividir o combinar archivos de una base de datos.
- Cambiar el nombre de los archivos de una base de datos.
- Mover un objeto de base de datos de un archivo a otro.
- Agregar objetos o archivos de datos nuevos a la base de datos.

Observe que no se menciona la eliminación de cosas. Por lo tanto, debe ser obvio que eliminar un objeto de una base de datos provocará la falla de todo lo que utiliza ese objeto. Sin embargo, todo lo demás debe quedar intacto, excepto tal vez la disponibilidad; algunos DBMS requerirán que se desactive la base de datos o un servicio del DBMS mientras se hacen ciertos cambios en la capa física.

Independencia lógica de los datos

La capacidad para hacer cambios en la capa lógica sin afectar a los usuarios y procesos existentes es la *independencia lógica de los datos*. En la figura 1-1 se muestra que la transformación entre las capas lógica y externa proporciona la independencia lógica de los datos. Al igual que con la independencia física de los datos, existen grados de independencia lógica de los datos. Es importante que comprenda que casi todos los cambios *lógicos* también incluyen un cambio *físico*. Por ejemplo, no puede agregar un nuevo objeto a la base de datos (como una tabla en un DBMS relacional) sin guardar físicamente los datos en algún lugar; por lo tanto, se efectúa un cambio correspondiente en la capa física. Además, la eliminación de objetos en la capa lógica hará que falle todo lo que utiliza esos objetos, pero no debe afectar otras cosas.

Éstos son algunos ejemplos de cambios en la capa lógica que pueden hacerse de manera segura gracias a la independencia lógica de los datos:

- Agregar un objeto nuevo a la base de datos.
- Incorporar elementos de datos a un objeto existente.
- Hacer cualquier cambio en que se pueda colocar una vista en el modelo externo que reemplaza al objeto original en la capa lógica (y que se procesa igual que éste), como combinar o dividir objetos existentes.

Modelos prevalecientes de bases de datos

En esencia, un *modelo de bases de datos* es la arquitectura que utiliza el DBMS para guardar los objetos dentro de la base de datos y relacionarlos entre sí. Aquí se presentan los modelos más frecuentes en orden de evolución. En la siguiente sección aparece una breve historia de las bases de datos relacionales para que adquiera una perspectiva cronológica.

Archivos simples

Los *archivos simples* son archivos comunes de un sistema operativo; en ellos, los registros de un archivo no contienen información para comunicar su estructura, o cualquier relación entre los registros, a las aplicaciones que utiliza el archivo. Cualquier información acerca de la estructura o el significado de los datos en el archivo deben incluirse en cada aplicación que lo utilice o debe conocerla cada persona que lo lee. En esencia, los archivos simples no son bases de datos, porque no cumplen ninguno de los criterios analizados antes. No obstante, es importante que los comprenda por dos razones. En primer lugar, suelen ser utilizados para guardar información de bases de datos. En este caso, el sistema operativo todavía desconoce el contenido y la estructura de los archivos, pero el DBMS tiene *metadatos* que le permiten convertir los archivos simples de la capa física en las estructuras de base de datos de la capa lógica. *Metadatos*, que literalmente significa “datos acerca de los datos”, es el término utili-

zado para la información que conserva la base de datos en su catálogo para describir los datos guardados en ella y las relaciones entre los datos. Por ejemplo, los metadatos de un cliente pueden incluir todos los elementos de datos recopilados acerca del cliente (como nombre, dirección y estado de cuenta) junto con la longitud, los valores mínimo y máximo de los datos y una breve descripción de cada elemento. En segundo lugar, los archivos simples existieron antes que las bases de datos, y los primeros sistemas de bases de datos *evolucionaron* a partir de los sistemas de archivo simple que les precedieron.

En la figura 1-2 se muestra un sistema de archivo simple, un subconjunto de los datos de la compañía ficticia Northwind Traders, proveedor internacional de artículos alimenticios (y la base de datos de ejemplo de Microsoft). Recuerde que los títulos de las columnas (Id de cliente, Nombre de compañía, etc.) sólo se incluyen como ejemplo; únicamente los registros de datos se guardarían en los archivos reales. Los datos del cliente se guardan en un archivo Clientes, y cada registro representa un cliente de Northwind. Cada empleado de Northwind tiene un registro en el archivo Empleados y cada producto vendido por Northwind tiene un registro en el archivo Productos. Los datos de un pedido (los pedidos hechos a Northwind por sus clientes) se guardan en otros dos archivos simples. El archivo Pedidos contiene un registro para cada pedido de cliente con los datos correspondientes, como la identificación del cliente que hizo el pedido y el nombre del empleado que aceptó el pedido del cliente. El archivo Detalles de pedido contiene un registro de cada artículo de un pedido (un pedido puede contener varios artículos de línea, uno para cada producto solicitado) incluidos datos como precio unitario y cantidad.

Un *programa de aplicación* es una unidad de lógica del programa computacional que efectúa una función específica dentro de un sistema de aplicación. Northwind tiene un programa de aplicación que imprime una lista de todos los pedidos. Esta aplicación debe correlacionar los datos entre los cinco archivos, al leer un pedido y realizar los pasos siguientes:

1. Emplear Id de cliente para encontrar el nombre del cliente en el archivo Clientes.
2. Usar Id de cliente para hallar el nombre del empleado relacionado en el archivo Empleados.
3. Utilizar Id de pedido para ubicar los artículos de línea correspondientes en el archivo Detalles de pedido.
4. Para cada artículo de línea, aplicar Id de producto con el fin de hallar el nombre de producto correspondiente en el archivo Productos.

Esto es muy complicado, porque sólo intentamos imprimir una simple lista de todos los pedidos; sin embargo, éste es el mejor diseño de datos posible de un sistema de archivo simple.

Un diseño alterno sería combinar toda la información en un solo archivo con todos los datos acerca de clientes, empleados y pedidos, en un solo registro para cada pedido. Aunque esto simplificaría mucho la recuperación de los datos, considere las repercusiones de repetir todos los datos del cliente en cada artículo de línea de un pedido. Es posible que no pueda agregar un cliente nuevo hasta que éste tenga un pedido preparado. Asimismo, si alguien elimina el último pedido de un cliente, se perdería toda la información de éste. Pero lo peor

Archivo Clientes

Id de cliente	Compañía	Apellidos	Nombre	Cargo	Ciudad	Estado o provincia
6	Compañía F	Pérez-Olaeta	Francisco	Jefe de compras	Milwaukee	WI
26	Compañía Z	Pinto	Armando	Ayudante de contabilidad	Miami	FL

Archivo Empleados

Id	Apellidos	Nombre	Cargo
2	Escolar	Jesús	Vicepresidente de ventas
5	San Juan	Patricia	Jefe de ventas
9	Chaves	Francisco	Representante de ventas

Archivo Productos

Id	Código de producto	Nombre del producto	Categoría	Cantidad por unidad	Precio listado
5	NWTO-5	Aceite de oliva Northwind Traders	Aceite	36 cajas	$21.35
7	NWTDFN-7	Peras secas Northwind Traders	Frutos secos	12 paq. de 1/2 kg	$30.00
40	NWTCM-40	Carne de cangrejo Northwind Traders	Carne enlatada	24 latas de 120 g	$18.40
41	NWTSO-41	Sopa de almejas Northwind Traders	Sopas	12 latas de 360 g	$ 9.65
48	NWTCA-48	Chocolate Northwind Traders	Golosinas	10 paq.	$12.75
51	NWTDFN-51	Manzanas secas Northwind Traders	Frutos secos	50 paq. de 300 g	$53.00

Archivo Pedidos

Id de pedido	Id de cliente	Id de empleado	Fecha de pedido	Fecha de envío	Gastos de envío
51	26	9	05/04/2006	05/04/2006	$60.00
56	6	2	03/04/2006	03/04/2006	$ 0.00
79	6	2	23/06/2006	23/06/2006	$ 0.00

Archivo Detalles de pedido

Id de pedido	Id de producto	Precio	Cantidad
51	5	$21.35	25
51	41	$ 9.65	30
51	40	$18.40	30
56	48	$12.75	10
79	7	$30.00	30
79	51	$53.00	30

Figura 1-2 Sistema de pedidos de archivo simple.

sucedería cuando cambiara la información del cliente, porque tendría que encontrar y actualizar cada registro en que están repetidos sus datos. Se explorarán estos problemas con mayor detalle cuando se analice el diseño lógico de una base de datos en el capítulo 7.

Otro método alterno que se suele emplear en los sistemas basados en archivo simple consiste en combinar los archivos estrechamente relacionados, como Pedidos y Detalles de pedido, en uno solo, con los artículos de línea para cada pedido después de cada registro del encabezado de pedido y un elemento de datos Tipo de registro agregado, que ayuda a que la aplicación diferencie los dos tipos de registro. En este método, la Id de pedido se omitiría del registro Detalles de pedido, porque la aplicación sabe a cuál pedido pertenece el registro Detalles de pedido, por su posición en el archivo (después del registro Pedidos). Aunque este método facilita la correlación de los datos del pedido, lo consigue al incorporar la complejidad de mezclar diferentes tipos de registros en el mismo archivo, de modo que no aporta una ganancia neta a la simplicidad o al desarrollo más rápido de la aplicación.

En general, el mayor problema con el método de archivo simple es que la definición del contenido de cada archivo y la lógica requerida para correlacionar los datos de varios archivos simples deben incluirse en cada programa de aplicación que requiera esos archivos, lo que aumenta el costo y la complejidad de los programas de aplicación. Este mismo problema sirvió como incentivo para que los científicos computacionales hallaran un mejor modo de organizar los datos.

El modelo jerárquico

Las primeras bases de datos seguían el modelo jerárquico, que evolucionó a partir de los sistemas de archivos que reemplazaron las bases de datos, con los registros ordenados en una jerarquía similar a un organigrama. Cada archivo del sistema de archivo simple se convirtió en un *tipo de registro*, o *nodo* en terminología jerárquica, pero aquí se utiliza el término *registro* por simplicidad. Los registros se conectaban mediante *apuntadores* que contenían la dirección del registro relacionado. Los apuntadores indicaban a la computadora dónde se ubicaba físicamente el registro relacionado; igual que la dirección de una calle lo dirige a un edificio específico en una ciudad, un URL lo dirige a una página Web específica en Internet, o las coordenadas de un GPS apuntan a una ubicación determinada en el planeta. Cada apuntador establece una relación primario-secundario, también denominada *relación uno a varios*, en que un elemento principal puede tener muchos elementos secundarios, pero cada uno de éstos sólo puede tener un elemento primario. Esto es similar a la situación en una organización de negocios tradicional, donde cada gerente puede tener muchos empleados bajo su mando, pero cada empleado sólo tiene un gerente. El problema obvio del modelo jerárquico es que algunos datos no se ajustan exactamente a esta estructura jerárquica estricta, como cuando un pedido debe tener como elemento primario al cliente que lo generó y como otro elemento primario al empleado que capturó el pedido. (Las relaciones de los datos se presentan con mayor detalle en el capítulo 2.) La base de datos jerárquica más popular fue el Information Management System (IMS) de IBM.

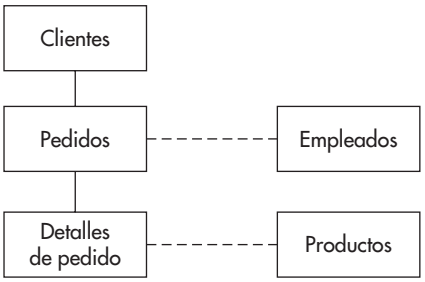


Figura 1-3 Estructura del modelo jerárquico de Northwind.

En la figura 1-3 se muestra la estructura jerárquica del modelo jerárquico de la base de datos Northwind Traders. Reconocerá los tipos de registros Clientes, Empleados, Productos, Pedidos y Detalles de pedido presentados antes. Al comparar la estructura jerárquica con el sistema de archivo simple presentado en la figura 1-2, observará que los registros Empleados y Productos se exhiben en la estructura jerárquica con líneas de guiones, porque no pueden conectarse a los otros registros mediante apuntadores. Ilustran la mayor limitación del modelo jerárquico, que fue la principal razón para su temprana desaparición: ningún registro puede tener más de un elemento primario. Por lo tanto, *no es posible* conectar los registros Empleados con los registros Pedidos, porque estos últimos ya tienen el registro Clientes como elemento principal. Asimismo, los registros Productos no pueden relacionarse con los registros Detalles de pedido porque éstos ya tienen al registro Pedidos como elemento principal. Los técnicos de bases de datos tenían que superar este defecto al relacionar los registros primarios "adicionales" en los programas de aplicación, como se había hecho con los sistemas de archivo simple, o al repetir todos los registros bajo cada elemento principal, lo que, por supuesto, desperdiciaba el entonces valioso espacio en disco, sin mencionar los desafíos de mantener sincronizados los datos redundantes. Ninguna de éstas era en realidad una solución aceptable, de modo que IBM modificó IMS para permitir varios elementos principales por registro. Al modelo de base de datos resultante se le llamó modelo *jerárquico extendido*, y tenía un gran parecido con el modelo de bases de datos de red actual, tal como se analiza en la sección siguiente.

En la figura 1-4 se presenta el contenido de registros seleccionados dentro del diseño del modelo jerárquico de Northwind. Por simplicidad se eliminaron algunos elementos de datos, pero una comparación con la figura 1-2 debe aclarar el contenido completo de cada registro, si es necesario. El registro del cliente 6 tiene un apuntador hacia su primer pedido (Id 56) y ese pedido tiene un apuntador al siguiente pedido (Id 79). Sabe que el pedido 79 es el último del cliente porque no tiene un apuntador a un pedido posterior. Al analizar la siguiente etapa de la jerarquía, el pedido 79 tiene un apuntador hacia su primer registro de Detalles de pedido (para el producto 7), y ese registro tiene un apuntador al siguiente registro del detalle (para el producto 51). Como se aprecia, en cada capa de la jerarquía, una cadena de apuntadores

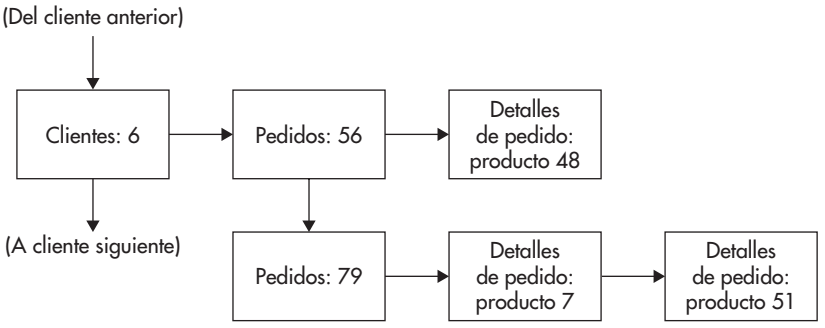


Figura 1-4 Contenido de un registro del modelo jerárquico de Northwind.

conecta los registros en la secuencia adecuada. Existe una diferencia adicional importante entre el sistema de archivo simple y el modelo jerárquico: en el modelo jerárquico, la clave (el identificador) del registro principal se elimina de los registros secundarios, porque los apuntadores manejan las relaciones entre los registros. Por lo tanto, Id de cliente e Id de empleado se eliminan del registro Pedidos, e Id de producto se elimina del registro Detalles de pedido. No es una buena idea dejarlos, porque esto permitiría que en las bases de datos apareciera información contradictoria, como que un cliente apunta a un pedido y, no obstante, el pedido contiene la identificación de un cliente distinto.

El modelo de red

El modelo de base de datos de red evolucionó casi al mismo tiempo que el modelo jerárquico. En esencia, se formó un comité de representantes de la industria para crear un modelo mejorado. Un cínico diría que un camello es un caballo diseñado por un comité, y sería correcto decir eso en este caso. El modelo de base de datos más popular basado en el modelo de red fue el Integrated Database Management System (IDMS), desarrollado en un principio por Cullinane (que cambió su nombre a Cullinet). El producto fue mejorado con extensiones relacionales, se denominó IDMS/R y terminó por ser vendido a Computer Associates.

Al igual que con el modelo jerárquico, los *tipos de registros* (o sencillamente los *registros*) representan lo que serían archivos separados en un sistema de archivo simple, y esos registros se enlazan mediante relaciones uno a varios, también llamadas *relaciones propietario-miembro* o *conjuntos*, en la terminología del modelo de red. Nos apegaremos una vez más a los términos *elemento primario* y *elemento secundario*, para simplificar la exposición. Al igual que con el modelo jerárquico, se utilizan apuntadores a la dirección física para conectar los registros relacionados, y se elimina cualquier identificación de los registros principales a partir de cada registro secundario, para evitar posibles inconsistencias. En contraste con el modelo jerárquico, se da nombre a las relaciones para que el programador pueda dar instruc-

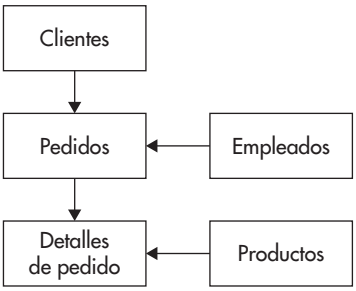


Figura 1-5 Estructura del modelo de red de Northwind.

ciones al DBMS de que utilice una relación específica para desplazarse de un registro a otro en la base de datos, permitiendo que un tipo de registro participe como elemento secundario de varias relaciones. El modelo de red aportó mayor flexibilidad, pero (como suele ocurrir con las computadoras) a costa de pérdida de la simplicidad.

La estructura del modelo de red de Northwind, mostrada en la figura 1-5, tiene los mismos registros que la estructura equivalente del modelo jerárquico de la figura 1-3. Por convención, el extremo de las líneas apunta del elemento primario al secundario. Observe también que los registros Clientes y Empleados ahora tienen líneas continuas en el diagrama de la estructura, porque se pueden implementar directamente en la base de datos.

En el ejemplo del contenido del modelo de red presentado en la figura 1-6, cada relación elemento principal-secundario se muestra con un tipo de línea diferente, lo que indica que cada relación tiene un nombre distinto. Esta diferencia es importante porque señala la más grande desventaja del modelo de red: su complejidad. En lugar de emplear una sola trayectoria para procesar los registros, ahora se utilizan muchas rutas. Por ejemplo, al comenzar con el registro para el empleado 2 (vicepresidente de ventas Jesús Escolar) y utilizarlo para hallar el primer pedido (Id 56), se llega a la cadena de pedidos que pertenecen al cliente 6 (la Compañía F). Aunque en realidad llegó al primer pedido de ese cliente, no tiene manera de saberlo. Con el fin de hallar los demás pedidos de este cliente, primero debe encontrar un modo de avanzar desde donde está hasta el final de la cadena, y luego volver al inicio y avanzar desde ahí hasta hallar el pedido con el que comenzó. En las bases de datos con el modelo de red, para satisfacer esta necesidad de procesamiento, todas las cadenas de apuntadores son circulares. Por lo tanto, puede seguir los apuntadores desde el pedido 56 hasta el siguiente (Id 79) y después al registro del cliente (Id 6) y por último regresar al pedido 56. Como imaginará, estas cadenas circulares de apuntadores pueden provocar fácilmente un *bucle infinito* (un proceso que nunca concluye) si un usuario de la base de datos no tiene cuidado de registrar su ubicación en ella y cómo llegó ahí. A grandes rasgos, la estructura de World Wide Web se parece a la de una base de datos de red en que cada página Web tiene vínculos con otras páginas relacionadas, y son frecuentes las referencias circulares.

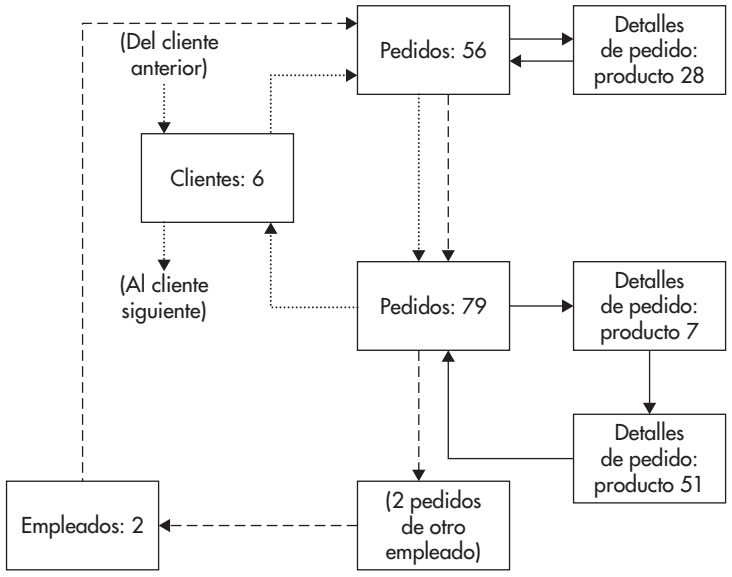


Figura 1-6 Registro en el modelo de red de Northwind.

Al proceso de desplazamiento por una base de datos de red se le denomina "recorrer el conjunto", porque requiere que se elijan trayectorias a través de la estructura de la base de datos de un modo muy parecido a como se eligen rutas en un bosque, cuando existen varias trayectorias hacia el mismo destino. Sin un mapa actualizado, es fácil perderse o, peor aún, llegar a un callejón sin salida donde es imposible arribar al destino buscado sin deshacer el camino. La complejidad de este modelo y el costo de un pequeño ejército de técnicos requerido para darle mantenimiento fueron factores importantes en su desaparición.

El modelo relacional

Además de la complejidad, los modelos de base de datos de red y jerárquico comparten otro problema común: son inflexibles. Deben seguirse las rutas preconcebidas a través de los datos para procesarlos con eficiencia. Las consultas *ad hoc*, como hallar todos los pedidos enviados en un mes específico, requieren que se explore toda la base de datos para localizarlos. Los científicos computacionales todavía buscan un mejor modo. Sólo unos cuantos elementos en la historia del desarrollo de las computadoras fueron verdaderamente revolucionarios, y el trabajo de investigación de E. F. (Ted) Codd que condujo al modelo relacional lo es.

El *modelo relacional* se basa en la noción de que cualquier ruta preconcebida a través de la estructura de datos es una solución demasiado restrictiva, sobre todo ante las cada vez mayores demandas para dar soporte a solicitudes *ad hoc* de información. Los usuarios de una base de datos no pueden pensar en cada uso posible de los datos antes de que se cree la base

de datos; por lo tanto, la imposición de rutas predefinidas a través de los datos sólo genera una "prisión de datos". El modelo relacional permite a los usuarios relacionar los registros *según se requiera* y no de manera predefinida, cuando se guardan los registros por primera vez en la base de datos. Además, el modelo relacional está creado de modo que las consultas funcionan con *conjuntos* de datos (por ejemplo, todos los clientes que tienen un saldo sobresaliente) en lugar de un registro a la vez, como ocurre con los modelos de red y jerárquico.

El modelo relacional presenta los datos en las familiares tablas bidimensionales, como lo hace una hoja de cálculo. Pero a diferencia de lo que sucede en ésta, no es necesario que los datos se guarden en forma tabular; además, el modelo también permite combinar (*unir* en la terminología relacional) tablas para formar *vistas*, que también se presentan como tablas bidimensionales. En resumen, sigue el modelo ANSI/SPARC y, por lo tanto, aporta dosis saludables de independencia física y lógica de los datos. En lugar de vincular los registros relacionados con apuntadores a una dirección física, como ocurre en los modelos jerárquico y de red, se guarda en cada tabla un elemento de datos común, como se hace en los sistemas de archivo simple.

En la figura 1-7 se muestra el diseño del modelo relacional de Northwind. Al repasar la figura 1-2 se confirma que cada archivo del sistema de archivo simple ha sido ubicado como una tabla en el modelo relacional. Como aprenderá en el capítulo 6, esta correspondencia uno a uno entre los archivos simples y las tablas relacionales no siempre resultará verdadera, pero es muy frecuente. En la figura 1-7, se incluyeron líneas entre las tablas para mostrar las relaciones uno a varios; la línea sencilla representa el lado "uno" y la línea dividida en tres partes (denominada "pata de gallo") designa el lado "varios". Por ejemplo, puede apreciar que "un" cliente se relaciona con "varios" pedidos y que "un" pedido se relaciona con "varios" Detalles de pedido con sólo inspeccionar las líneas que conectan a estas tablas. La técnica de diagramación presentada aquí, llamada *diagrama entidad-relación (ERD)*, se cubre con mayor detalle en el capítulo 7.

En la figura 1-8, tres de las cinco tablas se han representado con datos de muestra de las columnas seleccionadas. En particular, observe que la columna Id de cliente se guarda en las tablas Clientes y Pedidos. Cuando Id de cliente de una fila de la tabla Pedidos coincide con Id

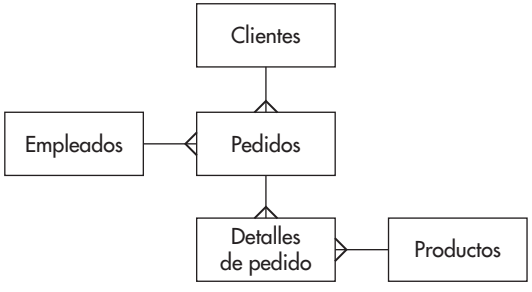


Figura 1-7 Estructura del modelo relacional de Northwind.

Tabla Clientes

Id de cliente	Compañía	Apellidos	Nombre	Cargo	Ciudad	Estado o provincia
6	Compañía F	Pérez-Olaeta	Francisco	Jefe de compras	Milwaukee	WI
26	Compañía Z	Pinto	Armando	Ayudante de contabilidad	Miami	FL

Tabla Pedidos

Id de pedido	Id de cliente	Id de empleado	Fecha de pedido	Fecha de envío	Gastos de envío
51	26	9	05/04/2006	05/04/2006	$60.00
56	6	2	03/04/2006	03/04/2006	$0.00
79	6	2	23/06/2006	23/06/2006	$0.00

Tabla Empleados

Id	Apellidos	Nombre	Cargo
2	Escolar	Jesús	Vicepresidente de ventas
5	San Juan	Patricia	Jefe de ventas
9	Chaves	Francisco	Representante de ventas

Figura 1-8 Contenido de la tabla relacional de Northwind.

de cliente de una fila de la tabla Clientes, se sabe qué pedido pertenece a ese cliente específico. Asimismo, la columna Id de empleado se guarda en las tablas Empleados y Pedidos, para indicar al empleado que aceptó cada pedido.

La elegante sencillez del modelo relacional y la facilidad con que las personas lo aprenden y comprenden han sido factores importantes para su aceptación universal. El modelo relacional es el tema principal de este libro debido a su uso extendido en los sistemas de tecnología de la información actuales y es probable que permanezca así durante los años futuros.

El modelo orientado a objetos

El modelo orientado a objetos (OO) en realidad comenzó en la década de 1970, pero no tuvo un uso comercial importante hasta la década de 1990. Este auge súbito provino de la incapacidad de los sistemas de administración de bases de datos relacionales de la época para manejar tipos de datos complejos como archivos de imágenes, dibujos complicados, y de audio y video. La súbita explosión de Internet y de World Wide Web creó una intensa demanda de transmisión de datos complejos.

Un *objeto* es un agrupamiento lógico de datos relacionados y de lógica de programa que representa algo real, como un cliente, empleado, pedido o producto. En el modelo OO, a los elementos de datos individuales, como la identificación y el nombre del cliente, se les denomina *variables* y se guardan dentro de cada objeto. También puede ver que a las variables se les denomina *variables de instancia* o *propiedades* pero, por consistencia, seguiremos utili-

zando el término *variables*. En la terminología de OO, un *método* es un segmento de lógica del programa de una aplicación que opera sobre un objeto específico y proporciona una función finita, como comprobar el límite de crédito de un cliente o actualizar la dirección de un cliente. Entre las numerosas diferencias entre el modelo OO y los modelos presentados antes, la más importante es que las variables *sólo* pueden ser consultadas mediante métodos. A esta propiedad se le llama *encapsulado*.

La definición estricta de *objeto* empleada aquí sólo se aplica al modelo OO. El término general *objeto de bases de datos*, utilizado al inicio del capítulo, se refiere a cualquier elemento con nombre que puede guardarse en una base de datos que no es OO (como una tabla, índice o vista). Así como los conceptos de OO se han abierto paso en las bases de datos relacionales, también lo ha hecho su terminología, aunque a menudo con definiciones menos precisas.

En la figura 1-9 se exhibe el objeto Clientes, ejemplo de una implementación de OO. El círculo de métodos que rodea al núcleo de variables nos recuerda el encapsulado. En realidad, puede considerar que un objeto es como un átomo con un campo de electrones de métodos y un núcleo de variables. Cada cliente de Northwind tendría su propia copia de la estructura de objetos, llamada *instancia de objetos*, de manera muy similar a cuando un cliente individual tiene una copia de la estructura del registro del cliente en un sistema de archivo simple.

A primera vista, el modelo OO parece horriblemente ineficiente porque sugiere que cada instancia requiere que los métodos y la definición de las variables se guarden de manera re-

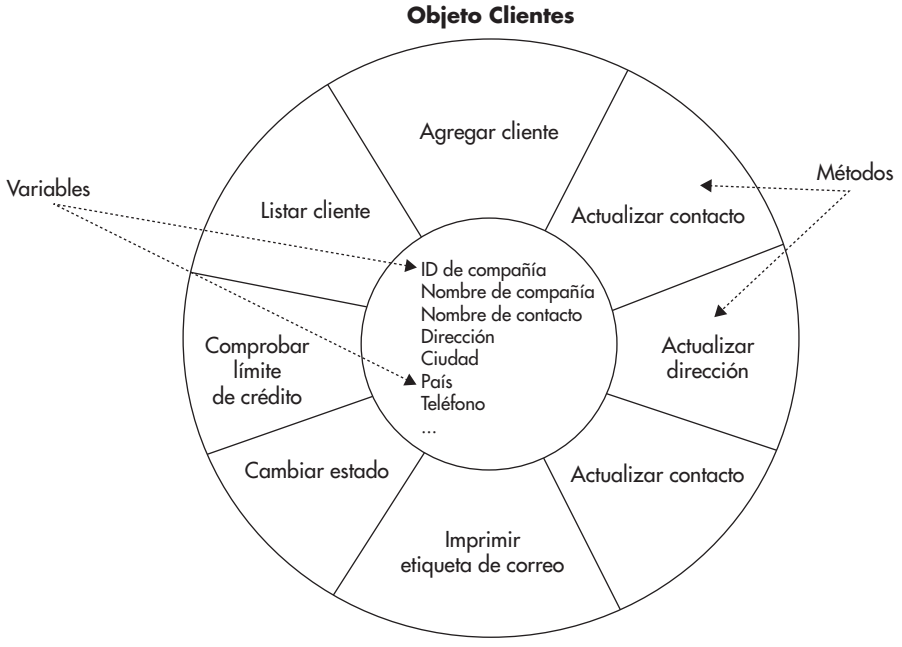


Figura 1-9 Anatomía de un objeto.

dundante. Sin embargo, no ocurre así por completo. Los objetos se organizan en una *jerarquía de clases*, de modo que los métodos y las definiciones de variables comunes sólo deben definirse una vez y luego son *heredados* por los otros miembros de la misma clase. Las variables también pertenecen a las clases y, por lo tanto, es fácil incorporar tipos de datos nuevos con sólo definir una clase nueva de ellos.

El modelo OO también permite *objetos complejos*, que son objetos formados por uno o más objetos adicionales. Esto suele implementarse mediante una *referencia* a objeto, en donde un objeto contiene el identificador de uno o más objetos adicionales. Por ejemplo, el objeto Clientes puede contener una lista de objetos Pedidos que haya hecho un cliente, y cada objeto Pedidos puede contener el identificador del cliente que hizo el pedido. Al identificador único de un objeto se le llama *identificador de objeto* (Objecto IDentifier, OID), cuyo valor se asigna automáticamente a cada objeto a medida que se crea y luego es invariable (es decir, su valor nunca cambia). La combinación de objetos complejos y la jerarquía de clases hace que las bases de datos OO sean muy convenientes para administrar datos no escalares, como dibujos y diagramas.

Los conceptos de OO tienen la ventaja de que se han abierto paso hacia casi todos los aspectos de las computadoras modernas. Por ejemplo, el Registro de Microsoft Windows (el directorio que guarda las especificaciones y las opciones de algunos sistemas operativos Windows) tiene una jerarquía de clases, y casi todas las aplicaciones de diseño asistido por computadora (Computer Aided Design, CAD) emplean una base de datos OO para guardar sus datos.

El modelo de objetos-relacional

Aunque el modelo OO aporta algunos beneficios significativos debido al encapsulado de los datos para reducir los efectos de las modificaciones del sistema, la falta de capacidad para consultas *ad hoc* lo ha relegado a un pequeño mercado en que se requieren datos complejos, pero no capacidad para consultas *ad hoc*. No obstante, algunos vendedores de bases de datos relacionales observaron los importantes beneficios del modelo OO, sobre todo su capacidad para ubicar con facilidad tipos de datos complejos, de modo que incorporaron este tipo de capacidad a sus productos DBMS relacionales con la esperanza de capitalizar lo mejor de ambos modelos. Aunque los puristas de los objetos nunca han adoptado este método, la táctica parece haber funcionado en buena medida, porque las bases de datos OO puras sólo ganan terreno en mercados específicos. El nombre original dado a la combinación fue *base de datos universal*, y aunque a los publicistas les encantó el término, nunca se estableció en los círculos técnicos, de modo que el nombre preferido para el modelo fue *de objetos-relacional (OR)*. Durante su evolución, se puede decir que las bases de datos de Oracle, DB2 e Informix han sido OR DMBS en diferentes grados.

Para comprender por completo el modelo OR, necesita un conocimiento más detallado de los modelos relacional y OO. Sin embargo, recuerde que un OR DMBS proporciona una combinación de características convenientes del mundo de los objetos, como el almacenamiento de tipos de datos complejos, con la relativa sencillez y facilidad de uso del modelo relacional.

Casi todos los expertos de la industria creen que la tecnología de objetos-relacional seguirá aumentando su participación en el mercado.

Breve historia de las bases de datos

Los proyectos de exploración espacial condujeron a muchos descubrimientos importantes en las industrias de ciencia y tecnología, entre ellas la tecnología de la información. Como parte del proyecto lunar Apolo de la NASA, la North American Aviation (NAA) creó un sistema jerárquico de archivos llamado Método de acceso generalizado a actualizaciones (GUAM, Generalized Update Access Method) en 1964. IBM se unió a NAA para desarrollar GUAM como la primera base de datos de modelo jerárquico disponible para venta; la llamó Information Management System (IMS) y la distribuyó en 1966.

Por esa época, General Electric creó de manera interna la primera base de datos basada en el modelo de red, bajo la dirección del destacado científico computacional Charles W. Bachman, y la nombró Integrated Data Store (IDS). En 1967, la Conference on Data Systems Languages (CODASYL), un grupo industrial, formó el Database Task Group (DBTG) y comenzó a trabajar en un conjunto de normas para el modelo de red. Como respuesta a las críticas de la restricción de "elemento principal único" del modelo jerárquico, IBM introdujo una versión de IMS que eliminaba el problema al permitir que los registros tuvieran un elemento principal "físico" y varios "lógicos".

En junio de 1970, E. F. (Ted) Codd, un investigador de IBM (y luego socio de la empresa), publicó un documento de investigación titulado "A relational model of data for large shared data banks" (Un modelo relacional de datos para grandes bancos de datos compartidos) en *Communications of the ACM*, la publicación periódica de la Association for Computing Machinery, Inc. (La publicación se obtiene fácilmente en Internet.) En 1971, el DBTG de CODASYL publicó sus normas, que tardaron más de tres años en elaborar. Esto inició cinco años de debate acalorado acerca de cuál modelo era el mejor.

Los defensores del DBTG de CODASYL afirmaban lo siguiente:

- El modelo relacional era demasiado matemático.
- No era posible crear una implementación eficiente del modelo relacional.
- Los sistemas de aplicación necesitaban procesar los datos de registro en registro.

Los promotores del modelo relacional argumentaban lo siguiente:

- Nada tan complicado como la propuesta del DBTG podía realmente ser el modo correcto de administrar datos.
- Las consultas orientadas a los conjuntos eran demasiado difíciles en el lenguaje del DBTG.
- El modelo de red no tenía un sustento formal en la teoría matemática.

El debate alcanzó su punto más alto en la conferencia ACM SIGMOD (Special Interest Group on Management of Data) de 1975. Codd y otros dos debatieron contra Bachman y otros dos acerca de los méritos de los dos modelos. Al final, los asistentes estaban más confundidos que nunca. En retrospectiva, esto ocurrió porque cada argumento expresado por las dos partes era completamente correcto. Sin embargo, el interés en el modelo de red se desvaneció notoriamente a fines de la década de 1970. La evolución de la tecnología de bases de datos computacionales que ocurrió a continuación demostró que el modelo relacional era la mejor opción, al ofrecer estos descubrimientos importantes:

- Surgieron lenguajes como el lenguaje estructurado de consultas (Structured Query Language, SQL) que no eran tan matemáticos.
- Las implementaciones experimentales del modelo relacional demostraron que podía conseguirse una eficiencia razonable, aunque nunca fue tan eficiente como una base de datos equivalente con el modelo de red. Asimismo, el precio de los equipos de cómputo se mantuvo a la baja, y se consideró que la flexibilidad era más importante que la eficiencia.
- Se incorporaron provisiones al SQL para permitir el procesamiento de un conjunto de datos mediante un método de un registro a la vez.
- Las herramientas avanzadas facilitaron todavía más el uso del modelo relacional.
- La investigación de Codd condujo al descubrimiento de una nueva disciplina en las matemáticas, conocida como *cálculo relacional.*

A mediados de la década de 1970, la investigación y el desarrollo de bases de datos estaban en su apogeo. Un equipo de 15 investigadores de IBM en San José, California, bajo la dirección de Frank King, trabajó de 1974 a 1978 para desarrollar un prototipo de base de datos relacional llamado System R, que se distribuyó comercialmente y se convirtió en la base de HP ALLBASE e IDMS/SQL. Larry Ellison y una compañía que después fue conocida como Oracle implementaron de manera independiente las especificaciones externas del System R. Ahora se sabe que el primer cliente de Oracle fue la Agencia Central de Inteligencia (CIA, Central Intelligence Agency). Con algunas modificaciones, IBM convirtió el System R en SQL/DS y luego en DB2, que es su base de datos insignia hasta la actualidad.

Un equipo de recuperación de la University of California, Berkeley, formado por estudiantes bajo la dirección de Michael Stonebraker y Eugene Wong, colaboró de 1973 a 1977 para desarrollar el Ingres DBMS. Ingres también se convirtió en un producto comercial y tuvo mucho éxito. Después fue vendido a Computer Associates, pero emergió de nuevo como una compañía independiente en 2005.

En 1976, Peter Chen presentó el modelo entidad-relación (ERD). Su obra destacó las debilidades de modelado del modelo relacional y se convirtió en la base de muchas técnicas de modelado posteriores. Si Codd es considerado el "padre" del modelo relacional, Chen debe ser considerado el del ERD. Los modelos ERD también se analizan en el capítulo 7.

Sybase, que fue un RDBMS exitoso desplegado en los servidores Unix, estableció un acuerdo conjunto con Microsoft para desarrollar la siguiente generación de Sybase (que luego se llamaría System 10) con una versión disponible en servidores Windows. Por razones que desconoce el público, la relación se complicó antes de que se concluyeran los productos, y cada parte siguió su camino con todo el trabajo desarrollado hasta ese momento. Microsoft concluyó la versión para Windows y vendió el producto como Microsoft SQL Server, mientras que Sybase se apresuró a entrar en el mercado con Sybase System 10. Los productos eran tan similares que se supo que los instructores de SQL Server empleaban en clase los manuales de Sybase, en lugar de la documentación de primera generación de Microsoft. Con los años, las líneas de productos se han diferenciado bastante, pero las raíces en Sybase de Microsoft SQL Server todavía son evidentes en el producto.

La tecnología relacional tomó el mercado por asalto en la década de 1980. Las bases de datos orientadas a objetos, que aparecieron en la década de 1970, también tuvieron éxito comercial en la década siguiente. En la década de 1990, aparecieron los sistemas de objetos-relacional; el primero en venderse fue Informix, seguido relativamente rápido por Oracle y DB2.

No sólo evolucionó la tecnología relacional actual, también lo hicieron las personas involucradas. Michael Stonebraker abandonó UC Berkeley para fundar Illustra, que vende una base de datos de objeto-relacional, y se convirtió en el director científico de Informix cuando se fusionó con Illustra. Luego fundó Cohera, StreamBase Systems y Vertica, y en la actualidad es profesor en el MIT. Bob Epstein, quien trabajó en el proyecto Ingres con Stonebraker, se mudó a su empresa junto con el producto Ingres. Después pasó a Britton-Lee (que fue adquirida por NCR) para trabajar en las primeras *máquinas de bases de datos* (computadoras especializadas en ejecutar sólo bases de datos) y luego fundó Sybase, donde fue director científico durante varios años, y en la actualidad participa en cuestiones ambientales y equipos de cómputo reutilizables. Por cierto, las máquinas de bases de datos desaparecieron porque eran demasiado costosas comparadas con la combinación de un RDBMS que funcionaba en una computadora común. En esa época, el área de la bahía de San Francisco fue un lugar emocionante para los científicos de las bases de datos porque todos los grandes productos relacionales comenzaron ahí, más o menos en paralelo con el crecimiento exclusivo de Silicon Valley. Algunos se han trasladados de ahí, pero DB2, Oracle y Sybase todavía realizan gran parte de sus operaciones en el área de la bahía.

¿Por qué concentrarse en el modelo relacional?

El resto de este libro se concentra en el modelo relacional, y cubre un poco de los modelos OO y objetos-relacional. Aparte de que el modelo relacional es el más frecuente de todos ellos

en los sistemas de negocios modernos, otras razones importantes apoyan este método, sobre todo para quienes conocen sobre bases de datos por primera vez:

- Es fácil definir, mantener y manipular las estructuras de almacenamiento de datos.
- Los datos se recuperan mediante consultas *ad hoc* sencillas.
- Los datos están bien protegidos.
- Existen normas ANSI (American National Standards Institute) e ISO (International Organization for Standardization) bien establecidas.
- Muchos vendedores ofrecen gran cantidad de productos.
- Es relativamente fácil la conversión entre las implementaciones de los vendedores.
- Los RDBMS son productos maduros y estables.

✓ Autoexamen Capítulo 1

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. ¿Cuál de los siguientes conceptos es aportado por la capa lógica del modelo ANSI/SPARC?

A Independencia física de los datos.

B Relaciones elementos principales-secundarios.

C Independencia lógica de los datos.

D Encapsulado.

2. ¿Cuál de los siguientes conceptos es proporcionado por la capa externa del modelo ANSI/SPARC?

A Independencia física de los datos.

B Relaciones elementos principales-secundarios.

C Independencia lógica de los datos.

D Encapsulado.

3. En relación con las vistas de usuario, ¿cuál de las siguientes afirmaciones *no* es verdadera?

A Los programas de aplicación hacen referencia a ellas.

B Las personas que consultan la base de datos hacen referencia a ellas.

C Se ajustan a las necesidades del usuario de la base de datos.

D Las actualizaciones de los datos se muestran en forma desfasada.

4. El esquema de la base de datos está contenido en la capa _____ del modelo ANSI/SPARC.

5. Las vistas de usuario están en la capa _____ del modelo ANSI/SPARC.

6. Cuando los programas de aplicación emplean los sistemas de archivo simple, ¿dónde residen las definiciones del archivo?

7. En relación con el modelo jerárquico de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

A Fue desarrollado primero por Peter Chen.

B Los datos y los métodos se guardan juntos en la base de datos.

C Cada nodo puede tener muchos padres.

D Los registros se conectan mediante apuntadores a una dirección física.

8. En relación con el modelo de red de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

A Fue desarrollado primero por E. F. Codd.

B Los datos y los métodos se guardan juntos en la base de datos.

C Cada nodo puede tener muchos elementos principales.

D Los registros se conectan mediante apuntadores a una dirección física común.

9. En relación con el modelo relacional de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

A Fue desarrollado por Charles Bachman.

B Los datos y los métodos se guardan juntos en la base de datos.

C Los registros se conectan mediante apuntadores a una dirección física.

D Los registros se conectan mediante elementos de datos comunes en cada registro.

10. En relación con el modelo orientado a objetos de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

A Fue desarrollado por Charles Bachman.

B Los datos y los métodos se guardan juntos en la base de datos.

C Los datos se presentan en tablas bidimensionales.

D Los registros se conectan mediante elementos de datos comunes en cada registro.

11. En relación con el modelo de objetos-relacional de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

- **A** Sólo atiende a un mercado reducido y casi todos los expertos creen que así se mantendrá.
- **B** Los registros se conectan mediante apuntadores a una dirección física.
- **C** Fue desarrollado al incorporar propiedades similares a las de objetos al modelo relacional.
- **D** Fue creado al agregar propiedades similares a las relacionales al modelo orientado a objetos.

12. Según los defensores del modelo relacional, ¿cuál de las siguientes afirmaciones describe los problemas con el modelo de CODASYL?

- **A** Es demasiado matemático.
- **B** Es muy complicado.
- **C** Las consultas orientadas a conjuntos son muy difíciles.
- **D** No tiene sustento formal en la teoría matemática.

13. De acuerdo con los defensores del modelo de CODASYL, ¿cuál de las siguientes afirmaciones describe los problemas del modelo relacional?

- **A** Es demasiado matemático.
- **B** Las consultas orientadas a conjuntos son muy difíciles.
- **C** Los sistemas de aplicación necesitan procesar de registro en registro.
- **D** Es menos eficiente que las bases de datos del modelo CODASYL.

14. La capacidad para incorporar un objeto nuevo a una base de datos sin alterar los procesos existentes es un ejemplo de _____.

15. La propiedad que más distingue a una tabla de base de datos relacional de una hoja de cálculo es la capacidad para presentar a varios usuarios _____ propias.

Capítulo 2

Exploración de los componentes de una base de datos relacional

Habilidades y conceptos clave

- Componentes del diseño conceptual de una base de datos
- Componentes de los diseños lógico y físico de una base de datos

En este capítulo se exploran los componentes conceptual, lógico y físico que forman el modelo relacional. El *diseño conceptual de una base de datos* incluye el estudio y modelado de los datos de manera independiente de la tecnología. El modelo conceptual de los datos que se obtiene, en teoría, se puede implementar en cualquier base de datos o aun en un sistema de archivo simple. La persona que efectúa el diseño conceptual de una base de datos se denomina *modelador de datos*. El *diseño lógico de una base de datos* es el proceso de trasladar, o *ubicar*, el diseño conceptual en un diseño lógico que se ajuste al modelo de base de datos elegido (relacional, orientado a objetos, de objetos-relacional, etc.). A un especialista que desarrolla el diseño lógico de una base de datos se le conoce como *diseñador de base de datos*, pero el administrador de una base de datos (DBA, DataBase Administrator) realiza de manera total o parcial este paso del diseño. El paso final es el *diseño físico de una base de datos*, que requiere la ubicación del diseño lógico en uno o más diseños físicos, cada uno ajustado al DBMS específico que administrará la base de datos y el equipo de cómputo en particular en que funcionará la base de datos. La persona que efectúa el diseño físico de la base de datos suele ser el DBA. Los procesos relacionados con el diseño de una base de datos se cubren en el capítulo 5.

En las secciones siguientes se exploran los componentes del diseño conceptual de una base de datos, y después los componentes de los diseños lógico y físico.

Componentes del diseño conceptual de una base de datos

En la figura 2-1 se presenta el diseño conceptual de la base de datos de Northwind. Este diagrama es similar a la figura 1-7 del capítulo 1, pero se han agregado algunos elementos para ilustrar puntos importantes. Los elementos con nombres (Entidad, Atributo, Relación, Regla empresarial y Datos de intersección) son los componentes básicos que forman un diseño conceptual de una base de datos. Cada uno se presenta en las secciones siguientes, excepto los datos de intersección, que se exponen en “Relaciones varios a varios”.

Entidades

Una *entidad* (o *clase de entidad*) es una persona, lugar, cosa, suceso o concepto sobre el que se recopilan datos. En otras palabras, las entidades son los objetos reales que nos interesan

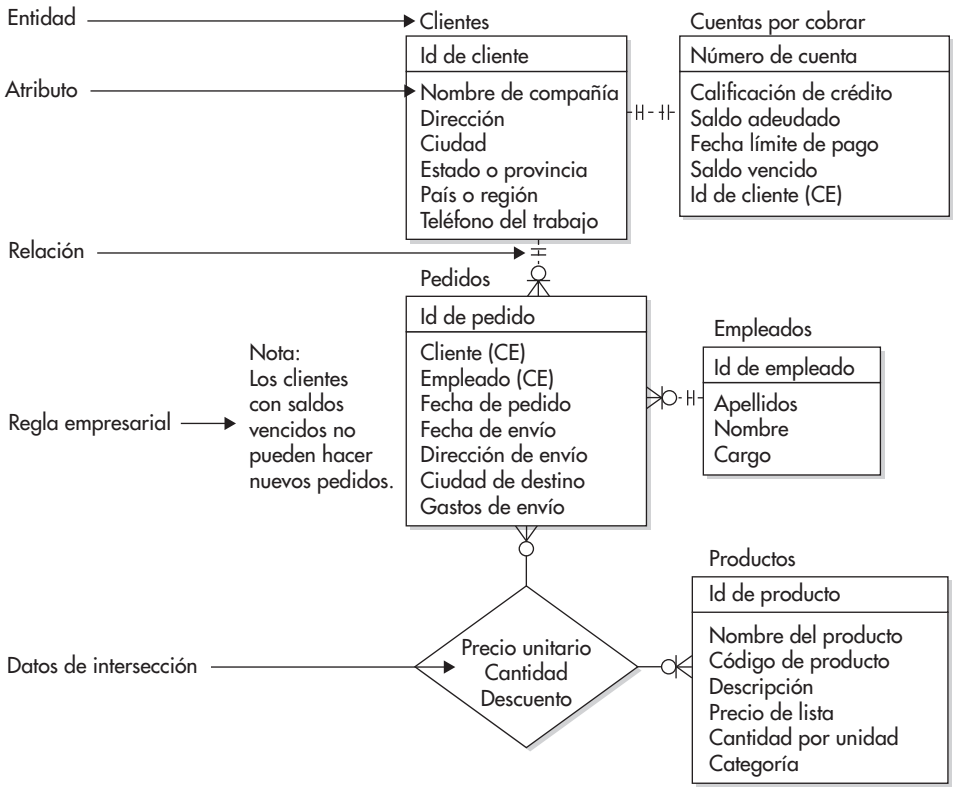


Figura 2-1 El diseño conceptual de la base de datos de Northwind.

lo suficiente como para capturar y guardar sus datos en una base de datos. Una entidad se representa como un rectángulo en el diagrama. Cualquier cosa que se designe con un nombre puede ser una entidad. Sin embargo, para no incluir en el diseño de nuestra base de datos todo lo que hay en el planeta, nos limitamos a las entidades que interesan a las personas que utilizarán nuestra base de datos. Cada entidad presentada en el modelo conceptual (figura 2-1) representa la clase completa de esa entidad. Por ejemplo, la entidad Clientes representa el conjunto de todos los clientes de Northwind. A los clientes individuales se les denomina *instancias* de la entidad.

Una *entidad externa* es la entidad con la que nuestra base de datos intercambia datos (le envía datos, los recibe de ella, o ambas cosas) pero sobre la que no se recopilan datos. Por ejemplo, casi todas las empresas que preparan cuentas de crédito para clientes adquieren informes de crédito de una o más oficinas de crédito. Envían la información de identificación del cliente a esa oficina y obtienen un informe de crédito, pero todos estos datos se refieren al *cliente*, no a la oficina. Al suponer que no existe una razón apremiante para que la base de

datos guarde información acerca de la oficina de crédito, como su dirección, la oficina de crédito no aparecerá en el diseño conceptual de la base de datos como una entidad. En realidad, sólo en raras ocasiones aparecen las unidades externas en los diseños de una base de datos, pero suelen aparecer en los diagramas de flujo como origen o destino de datos. Los diagramas de flujo se analizan en el capítulo 7.

Atributos

Un *atributo* es un hecho unitario que caracteriza o describe de alguna manera a una entidad. En el diagrama de diseño conceptual presentado en la figura 2-1, están incorporados como nombres dentro de un rectángulo que representa la entidad a la que pertenecen. Los atributos que aparecen en la parte superior del rectángulo (sobre la línea horizontal) forman el *identificador único* de la entidad. Como lo sugiere su nombre, un identificador único proporciona un valor único para cada instancia de la entidad. Por ejemplo, el atributo Id de cliente es el identificador único de la entidad Clientes, de modo que cada cliente debe tener un valor único en ese atributo. Recuerde que un identificador único puede tener varios atributos, pero cuando hay varios atributos, todavía se le considera *un* identificador único.

Se dice que los atributos son un hecho *aislado* porque deben ser *indivisibles*, lo que significa que no pueden dividirse en unidades más pequeñas que tengan algún significado. Por lo tanto, un atributo es la unidad de datos con nombre más pequeño que aparece en un sistema de base de datos. En este sentido, Dirección debe ser considerada un atributo sospechoso, porque fácilmente podría dividirse en Línea de dirección 1, Línea de dirección 2 y, tal vez, Línea de dirección 3, como suele ocurrir en los sistemas empresariales. Por ejemplo, este cambio tendría sentido porque facilita la impresión de etiquetas de direcciones. Por otra parte, el diseño de una base de datos no es una ciencia exacta y deben tomarse decisiones de juicio. Aunque es posible separar el atributo Teléfono del trabajo en sus atributos componentes, como Código de país, Código de área, Prefijo, Sufijo y Extensión, es necesario ponderar si ese cambio agrega sentido o valor. Aquí no existe una respuesta correcta o incorrecta, de modo que debemos confiar en la ayuda de las personas que emplearán la base de datos, o quienes financian el proyecto de base de datos, para tomar esas decisiones. Siempre recuerde que un atributo *debe* describir o caracterizar la entidad de alguna manera (por ejemplo, tamaño, forma, color, cantidad, ubicación).

Relaciones

Las *relaciones* son las asociaciones entre las entidades. Como las bases de datos se concentran en guardar datos relacionados, las relaciones se vuelven el pegamento que mantiene unida la base de datos. Las relaciones se muestran en el diagrama de diseño conceptual (figura 2-1) como líneas que conectan a una o más entidades. Cada extremo de una línea de relación muestra la *cardinalidad máxima* de la relación, que es la cantidad máxima de instancias de una entidad que se puede asociar con la entidad en el extremo opuesto de la línea. La cardi-

nalidad máxima puede ser *una* (la línea sin ningún símbolo especial en su extremo) o *varias* (la línea con una pata de gallo en el extremo). Justo antes del extremo de la línea está otro símbolo que indica la *cardinalidad mínima*, que es la cantidad mínima de instancias de una entidad que se pueden asociar con la entidad en el extremo opuesto de la línea. La cardinalidad mínima puede ser *cero*, que se representa con un círculo dibujado sobre la línea, o *uno*, que se señala con una breve línea perpendicular dibujada sobre la línea de la relación. Muchos modeladores de datos emplean dos líneas perpendiculares para indicar "una y *sólo* una", igual que en la figura 2-1.

Se requiere práctica para aprender a leer las relaciones, y definirlas y dibujarlas correctamente requiere *mucha más* práctica. El truco está en considerar la asociación entre las entidades en una dirección, y después invertir la vista para analizar en la dirección opuesta. Por ejemplo, para la relación entre Clientes y Pedidos, se deben formular dos preguntas: ¿cada cliente puede tener muchos pedidos? y ¿cada pedido puede tener muchos clientes? Por lo tanto, las relaciones se clasifican en tres tipos: *uno a uno*, *uno a varios*, y *varios a varios*, que se analizan en las secciones siguientes. Algunas personas afirman que varios a uno es un tipo de relación, pero en realidad es sólo una relación uno a varios desde una perspectiva inversa. Los tipos de relaciones se aprenden mejor mediante ejemplos. Aplicar las relaciones correctas es *esencial* para un diseño exitoso.

Pregunta al experto

P: Afirmó que las relaciones en el diseño conceptual son entre una o más entidades. Sin embargo, siempre me han dicho que las relaciones en un RDBMS son sólo entre dos tablas. ¿Cómo explica esto?

R: El diseño conceptual de una base de datos se suele crear en un nivel de abstracción superior a la base de datos física. Como verá más adelante en este capítulo, las restricciones referenciales aplicadas a la base de datos relacionados sólo permiten relaciones entre dos tablas, excepto para un caso especial llamado *relaciones recursivas* que incluyen una sola tabla. No obstante, nada evita que un diseñador sea más general en el diseño conceptual y presente una relación entre más de dos entidades. Por ejemplo, la relación entre Pedidos y Productos presentada en la figura 2-1 podría ser representada en un diseño conceptual como una entre Pedidos, Productos y Almacén de envío (el lugar que conserva el producto en el caso de un artículo de línea). Esta relación tendría que resolverse durante el diseño lógico, igual que deben ser aclarados los datos de intersección mostrados en la figura 2-1 (en algún momento se guardarán en una tabla). No se sienta amedrentado si esto parece confuso: se aclarará conforme aprenda sobre el diseño de bases de datos en los capítulos posteriores. En realidad, las relaciones que implican más de dos entidades son razonablemente raras, y también representan un tema avanzado, de modo que no se emplean en este libro.

Relaciones uno a uno

Una *relación uno a uno* es una asociación en que una instancia de una entidad se puede asociar *cuando mucho* con una instancia de la otra entidad, y viceversa. En la figura 2-1, la relación entre las entidades Clientes y Cuentas por cobrar es uno a uno. Esto significa que un cliente puede tener *cuando mucho* una cuenta por cobrar asociada, y una cuenta puede tener *cuando mucho* un cliente asociado. La relación también es *obligatoria* en ambas direcciones, lo que significa que un cliente debe tener *cuando menos* una cuenta por cobrar asociada con él, y una cuenta por cobrar debe tener *cuando menos* un cliente asociado con ella. Al integrar esto, se entiende que, en la relación entre las entidades Clientes y Cuentas por cobrar, "un cliente sólo puede tener una cuenta por cobrar asociada, y una cuenta por cobrar sólo puede tener un cliente asociado".

Otro concepto importante es la *transferibilidad*. Se dice que una relación es *transferible* si el elemento principal puede cambiar con el tiempo o, dicho de otro modo, si el elemento secundario puede ser reasignado a un elemento principal distinto. En este caso, es obvio que la relación entre Clientes y Cuentas por cobrar no es transferible porque nunca se toma la cuenta de un cliente y se transfiere a otro (hacerlo sería una práctica contable terrible). Por desgracia, no existe un símbolo universalmente aceptado para mostrar la transferibilidad en los modelos de datos, pero en algunos casos es una consideración importante, sobre todo con las relaciones uno a uno que son obligatorias en ambas direcciones.

Las relaciones uno a uno son sorprendentemente raras entre entidades. En la práctica, las relaciones uno a uno que son obligatorias en ambas direcciones y *no* son transferibles representan un defecto en el diseño que debe corregirse al combinar las dos entidades. Después de todo, ¿una cuenta por cobrar no es sólo más información sobre el cliente? No vamos a recopilar datos *sobre* una cuenta por cobrar; en cambio, la información en la entidad Cuentas por cobrar sólo son más datos que recopilamos *sobre* un cliente. Por otra parte, si compramos nuestro software de finanzas de un vendedor independiente (lo que es una práctica común), es casi seguro que vendrá con una base de datos predefinida que lo acepta, de modo que tal vez no tengamos otra opción que aceptar la situación. No podremos modificar el diseño de la base de datos del vendedor para agregar datos del cliente que nos interesa y, al mismo tiempo, no podremos hacer que el software del vendedor reconozca nada de lo que conservamos en nuestra propia base de datos.

En la figura 2-2 se expone un tipo diferente de una relación uno a uno que es *opcional* (algunos dicen *condicional*) en ambas direcciones. Suponga que diseñamos la base de datos para un distribuidor de automóviles. Éste entrega automóviles a algunos empleados, por lo general vendedores, para que los conduzcan cierto tiempo. Es obvio que no entrega *todos* los automóviles a los empleados (si lo hiciera, no tendría ninguno para venta). Podemos leer la relación entre las entidades Empleado y Automóvil de este modo: "En cualquier momento, cada empleado puede tener ningún o un automóvil que se le entrega, y cada automóvil puede

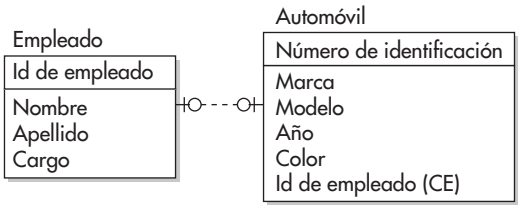


Figura 2-2 Relación entre empleados y automóviles.

estar asignado a ningún o un empleado". Observe la cláusula *en cualquier momento*. Si se le retira un automóvil a un empleado y se reasigna a otro, aún se tendría una relación uno a uno. Esto se debe a que cuando consideramos relaciones, siempre pensamos como si se tratara de una fotografía tomada en un momento arbitrario. Asimismo, por la descripción anterior, es evidente que la relación es transferible.

Relaciones uno a varios

Una *relación uno a varios* es una asociación entre dos entidades en que cualquier instancia de la primera entidad puede asociarse con una o más instancias de la segunda, y cualquier instancia de la segunda entidad puede asociarse con cuando mucho una instancia de la primera. En la figura 2-1 se presentan dos de estas relaciones: entre las entidades Clientes y Pedidos, y entre las entidades Empleados y Pedidos. La relación entre Clientes y Pedidos, que sólo es obligatoria en una dirección, se lee así: "En cualquier momento, cada cliente puede tener de cero a varios pedidos, y cada pedido debe tener uno y sólo un cliente propietario".

Las relaciones uno a varios son muy comunes. En realidad, son el bloque de construcción fundamental del modelo relacional de base de datos, porque todas las relaciones de una base de datos relacional se implementan como si fueran uno a varios. Es raro que sean opcionales en el lado "uno" e incluso más raro que sean obligatorias en el lado "varios", pero estas situaciones ocurren. Considere los ejemplos expuestos en la figura 2-3. Cuando se cierra la cuenta de un cliente, registramos el motivo de que se cerrara mediante un código de motivo de cierre de cuenta. Debido a que algunas cuentas están abiertas en cualquier momento, se trata de un código opcional. La relación se lee de esta manera: "En cualquier momento, cada valor de código de motivo de cierre de cuenta puede tener cero, uno o muchos clientes asignados a él, y cada cliente puede tener cero o un código de motivo de cierre de cuenta asignado a él". Luego supongamos que, como política de la compañía, no es posible abrir una cuenta de cliente sin obtener primero un informe de crédito, y que todos los informes de crédito se conserven en la base de datos, lo que significa que cualquier cliente puede tener más de un informe de crédito en la base de datos. Esto convierte en uno a varios la relación entre las entidades Clientes e Informes de crédito, además de obligatoria en ambas direcciones. Por lo tanto, la relación se

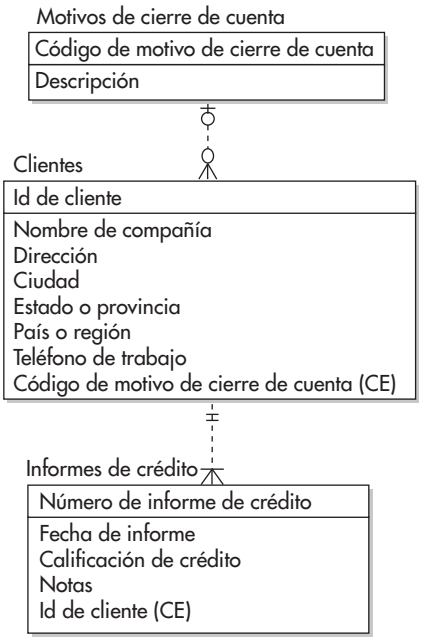


Figura 2-3 Relaciones uno a varios.

lee: "En cualquier momento, cada cliente puede tener uno o varios informes de crédito, y cada informe de crédito pertenece a un cliente y sólo a uno".

Relaciones varios a varios

Una *relación varios a varios* es una asociación entre dos entidades en que cualquier instancia de la primera entidad puede asociarse con cero, una o más instancias de la segunda, y viceversa. De nuevo en la figura 2-1, la relación entre Pedidos y Productos es varios a varios. La relación se lee así: "En cualquier momento, cada pedido contiene de cero a varios productos, y cada producto aparece en cero a varios pedidos".

Esta relación particular tiene datos asociados, como se observa en el rombo de la figura 2-1. A los datos que pertenecen a una relación varios a varios se les denomina *datos de intersección*. Los datos no tienen sentido a menos que los asocie con ambas entidades al mismo tiempo. Por ejemplo, Cantidad no tiene sentido a menos que sepa *quién* (cuál cliente) pidió *qué* (cuál producto). Si vuelve a observar la figura 1-7 del capítulo 1, reconocerá estos datos como la tabla Detalles de pedido del modelo relacional de Northwind. Entonces, ¿por qué no se muestra simplemente Detalles de pedido como una entidad? La respuesta es sencilla: no se ajusta a la definición de una entidad. No recopilamos datos sobre los artículos de

línea en el pedido; en cambio, los artículos de línea del pedido son sólo más datos acerca del pedido.

Las relaciones varios a varios son muy comunes, y casi todas ellas tienen datos de intersección. Las malas noticias son que el modelo relacional no permite directamente las relaciones varios a varios. No hay problema por tener relaciones varios a varios en un diseño conceptual, porque ese diseño es independiente de cualquier tecnología específica. Sin embargo, si la base de datos va a ser relacional, debe hacer algunos cambios cuando ubique el modelo conceptual sobre el modelo lógico correspondiente. La solución consiste en ubicar los datos de intersección en una tabla separada (una *tabla de intersección*) y la relación varios a varios en dos relaciones uno a varios, con la tabla de intersección en medio y en el lado "varios" de ambas relaciones. En la figura 1-7 se muestra este resultado, en que la tabla Detalles de pedido contiene los datos de intersección y participa en dos relaciones uno a varios que reemplazan la relación varios a varios original. El proceso para reconocer y manejar el problema de varios a varios se cubre de manera detallada en el capítulo 6.

Relaciones recursivas

Hasta este momento, ha conocido relaciones entre instancias de entidades diferentes. Sin embargo, pueden existir relaciones entre instancias de entidades del mismo tipo. Se denominan *relaciones recursivas.* Cualquiera de los tipos de relaciones ya presentados (uno a uno, uno a varios, o varios a varios) puede ser una relación recursiva. En la figura 2-4 y en la lista siguiente se presentan ejemplos de cada uno:

- **Uno a uno** Si fuéramos a rastrear cuáles empleados se casaron con otros empleados, esperaríamos que cada uno estuviera casado con cero o uno de otros empleados en cualquier momento.

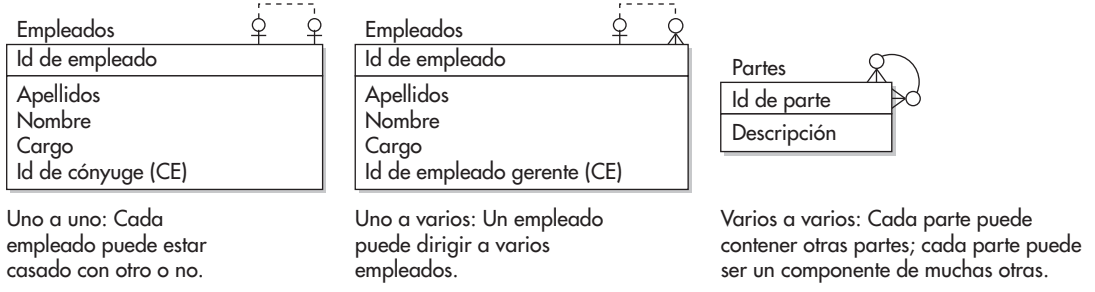


Figura 2-4 Ejemplos de relaciones recursivas.

- **Uno a varios** Es común rastrear la "cadena de alimentación" de empleados para saber quién está bajo las órdenes de quién. En casi todas las organizaciones, las personas sólo pueden tener un supervisor o gerente. Por lo tanto, normalmente esperamos ver que cada empleado está bajo las órdenes de ninguno o de otro empleado, y los empleados que son gerentes o supervisores tienen uno o más subordinados directos.
- **Varios a varios** En producción, una relación común tiene que ver con las partes que forman un producto terminado. Por ejemplo, si analiza una unidad de CD-ROM en un equipo de cómputo personal, puede imaginar que está formado por varias partes y, no obstante, el ensamble completo sólo se muestra como un elemento de la lista de partes de su equipo. De modo que cualquier parte puede estar formada por muchas otras y, al mismo tiempo, cualquier parte puede ser un componente de muchas otras.

Reglas de negocios

Una *regla de negocios* es una política, procedimiento o norma adoptada por una organización. Las reglas de negocios son *muy* importantes en el diseño de una base de datos porque determinan los controles que deben aplicarse a los datos. En la figura 2-1, se aprecia una regla de negocios que afirma que sólo se aceptarán pedidos de clientes que no tienen un saldo vencido. Casi todas las reglas de negocios se imponen mediante procedimientos manuales que se ordena seguir a los empleados o mediante la lógica de los programas de aplicación. Sin embargo, hay posibilidades de que se pase por alto a cada uno de éstos: los empleados olvidan un procedimiento manual o deciden no seguirlo, y personas autorizadas pueden actualizar directamente las bases de datos, superando los controles incluidos en los programas de aplicación. La base de datos puede funcionar muy bien como última línea de defensa. Las reglas de negocios se implementan en la base de datos como *restricciones*, que son las reglas formalmente definidas que limitan, de alguna manera, los valores de los datos de la base de datos. La sección "Restricciones", en páginas posteriores del capítulo, contiene más información sobre las restricciones. Observe que las reglas de negocios no suelen mostrarse en un diagrama de un modelo conceptual de datos; la que se exhibe en la figura 2-1 es sólo un ejemplo. Resulta mucho más común incluirlas en un documento de texto que acompaña al diagrama.

Pruebe esto 2-1 Exploración de la base de datos Northwind

Durante el resto de este capítulo y todo el capítulo 3, se emplean Microsoft Access 2007 y la base de datos de Northwind para ejemplificar conceptos. En este ejercicio se conectará a la base de datos de muestra de Northwind, ya sea en su propio equipo o mediante Microsoft Office Online, y se familiarizará con el desplazamiento por Microsoft Access para seguir los

ejemplos utilizados en este capítulo y el siguiente. No olvide que Access 2007 tiene un aspecto completamente diferente de las versiones anteriores, de modo que tal vez le resulte difícil seguir los procedimientos si emplea éstas. Sin embargo, la solución es sencilla porque sólo requiere Microsoft Office Online en un navegador Web y una conexión a Internet razonablemente rápida.

La elección de Microsoft Access para estos ejemplos conceptuales se hizo por conveniencia y no representa una recomendación de este producto sobre cualquier otro. En realidad, al cubrir SQL en el capítulo 4, se emplean otros productos RDBMS para demostración, entre ellos Oracle.

Paso a paso

1. Si cuenta con Microsoft Access 2007, descargue e instale la base de datos de ejemplo de Northwind mediante los pasos siguientes:
   a. Inicie Access 2007 desde el menú Inicio sin ninguna base de datos abierta.
   b. En el lado izquierdo de la pantalla de inicio, haga clic en Ejemplo, bajo el título Desde Microsoft Office Online.
   c. Haga clic en el ícono Northwind 2007.
   d. En la esquina inferior derecha del panel, haga clic en el botón Descargar, y responda a cualquier mensaje adicional.
   e. Una vez conectado a la base de datos, se mostrará una pantalla semejante a la de la figura 2-5.
2. Si no cuenta con Microsoft Access 2007, puede acceder a ella mediante Microsoft Office Online, usando sólo su navegador Web, si sigue estos pasos:
   a. Escriba en su navegador la URL **http://office.microsoft.com/en-us/products/** y luego oprima ENTER.
   b. En la parte central de la pantalla, busque y haga clic en el ícono Try Office 2007 Online.
   c. En la página siguiente, haga clic en Launch Test Drive y responda a los mensajes adicionales. El proceso de cargar el software y establecer una conexión con la base de datos puede requerir varios minutos.
   d. En la página Tutorial Menu, haga clic en Office Access 2007.

(*continúa*)

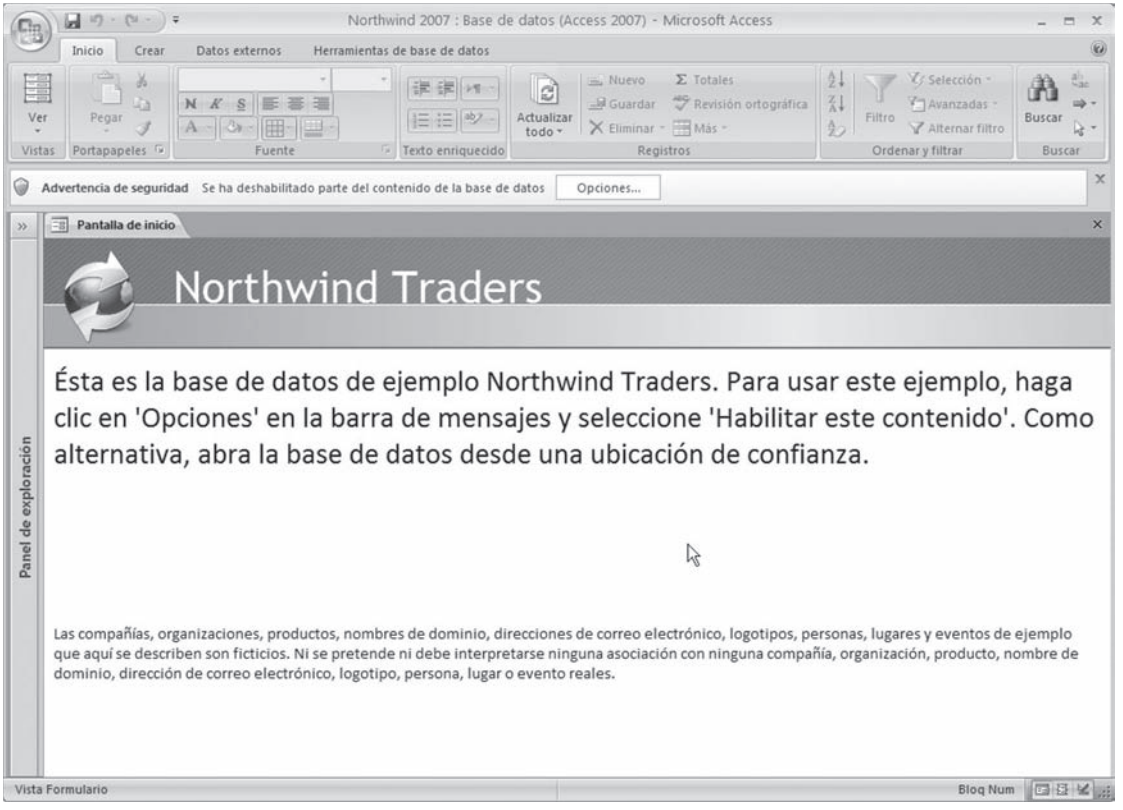


Figura 2-5 La pantalla de inicio de la base de datos de Northwind.

e. En el margen izquierdo del panel Getting Started, haga clic en Ejemplo, bajo el título Desde Microsoft Online.

f. En la esquina inferior derecha del panel (es posible que necesite cambiar el navegador a pantalla completa para verlo), haga clic en el botón Download y responda a los mensajes adicionales. En particular, observe lo siguiente:

- Desde el sitio Web puede recibir uno o más mensajes acerca de ejecutar complementos (add-ons). Éstos aparecerán cerca de la parte superior de la pantalla, bajo la línea con la estrella dorada, que suele tener un fondo amarillo claro (similar a la advertencia de seguridad de la figura 2-5).

- Tendrá que responder a la Advertencia de seguridad presentada en la figura 2-5. Haga clic en el botón Opciones, en la línea del mensaje, y seleccione la opción Habilitar este contenido.
- La primera vez que abra la base de datos, es probable que le pidan que inicie sesión. Si ocurre esto, haga clic en la opción Cancelar.

g. Una vez conectado a la base de datos, se mostrará una pantalla semejante a la de la figura 2-5.

3. En la cinta de opciones (el área a lo largo de la parte superior del panel que contiene opciones), haga clic en Herramientas de base de datos, y luego en la opción Relaciones. Se muestra el panel Relaciones, que presenta 18 tablas y las relaciones entre ellas. Verá un diagrama muy saturado, pero si sigue las líneas, puede apreciar fácilmente cada relación.

4. Cierre el panel Relaciones al hacer clic en la X que se encuentra a la derecha de la ficha Relaciones.

5. Amplíe el Panel de exploración (a lo largo del margen izquierdo del panel) al hacer clic en el ícono >>, cerca de la parte superior del panel. La base de datos contiene varias pantallas, informes y otros objetos utilizados para mostrar las opciones de programación dentro de Microsoft Access 2007. Sin embargo, sólo nos interesan los objetos de la base de datos (la programación de aplicaciones está más allá del alcance de este libro). Amplíe Objetos auxiliares para ver una lista de todas las tablas incluidas en la base de datos Northwind. Para cada tabla, puede hacer clic con el botón secundario del ratón en su nombre y seleccionar Abrir, para ver su contenido (las filas de datos), o Vista Diseño, para ver la definición de la tabla. No se preocupe si no comprende todo lo que observa; estos paneles se describen con mayor detalle las secciones siguientes.

6. Cierre Microsoft Access 2007 (u Office 2007 Online y la ventana de su navegador).

Resumen de Pruebe esto

Ha conseguido consultar la base de datos de ejemplo de Northwind que servirá para mostrarle conceptos por el resto de este capítulo y el siguiente. Se ha desplazado al panel Relaciones y a la lista Objetos auxiliares en el Panel de exploración.

Componentes de los diseños lógico y físico de una base de datos

El diseño lógico de una base de datos se implementa en una capa lógica del modelo ANSI/SPARC analizado en el capítulo 1. El diseño físico se implementa en la capa física de ANSI/SPARC. Sin embargo, al abrirnos paso por el DBMS para implementar la capa física, se vuelve difícil separar las dos capas. Por ejemplo, cuando creamos una tabla, incluimos una cláusula en el comando *create table* que indica al DBMS dónde queremos ponerla. Luego, éste asigna espacio automáticamente para la tabla en los archivos solicitados del sistema operativo. Debido a que gran parte de la implementación física está encerrada en las definiciones del DBMS de las estructuras lógicas, aquí se ha optado por no tratar de separarlas. Durante el diseño lógico de la base de datos, es posible asignar propiedades de almacenamiento físico (nombre de archivo o de espacio de tabla, ubicación del almacenamiento e información de tamaño) a cada objeto de la base de datos, a medida que se ubican desde el modelo conceptual, o se pueden omitir al principio y agregarse después en el paso del diseño físico que sigue al diseño lógico. Para usar el tiempo de manera eficiente, casi todos los DBA efectúan los dos pasos del diseño (lógico y físico) en paralelo.

Tablas

En el modelo relacional, la principal unidad de almacenamiento es la *tabla*, que es una estructura bidimensional formada por filas y columnas. Cada fila corresponde a una aparición de la entidad que representa la tabla, y cada columna corresponde a un atributo de esa entidad. Al proceso de ubicar las entidades del diseño conceptual en tablas en el diseño lógico se le denomina *normalización* y se cubre con detalle en el capítulo 6. A menudo, una entidad del modelo conceptual se ubica exactamente en una tabla en el modelo lógico, pero esto no ocurre siempre. Por razones que conocerá con el proceso de normalización, las entidades se suelen dividir en varias tablas y, en raros casos, varias entidades se pueden combinar en una tabla. En la figura 2-6 se muestra una lista de partes de la tabla Pedidos de Northwind.

Debe recordar que una tabla relacional es una estructura de almacenamiento *lógico* y, por lo general, no existe en forma tabular en la capa física. Cuando el DBA asigna una tabla a los archivos del sistema operativo en la capa física (llamados *espacios de tabla* en casi todos los RDBMS), es común que varias tablas se ubiquen en un solo espacio de tabla. Sin embargo, las tablas grandes pueden colocarse en su propio espacio de tabla o dividirse en varios espacios de tabla, y a esto se le denomina *partición*. Esta flexibilidad no existe en los RDBMS para equipos personales, como Microsoft Access.

El DBA que la crea debe asignar a cada tabla un nombre único. La longitud máxima de estos nombres varía mucho entre los productos de RDBMS, de un mínimo de 18 caracteres a un máximo de 255. Los nombres de las tablas deben ser descriptivos y reflejar el nombre de

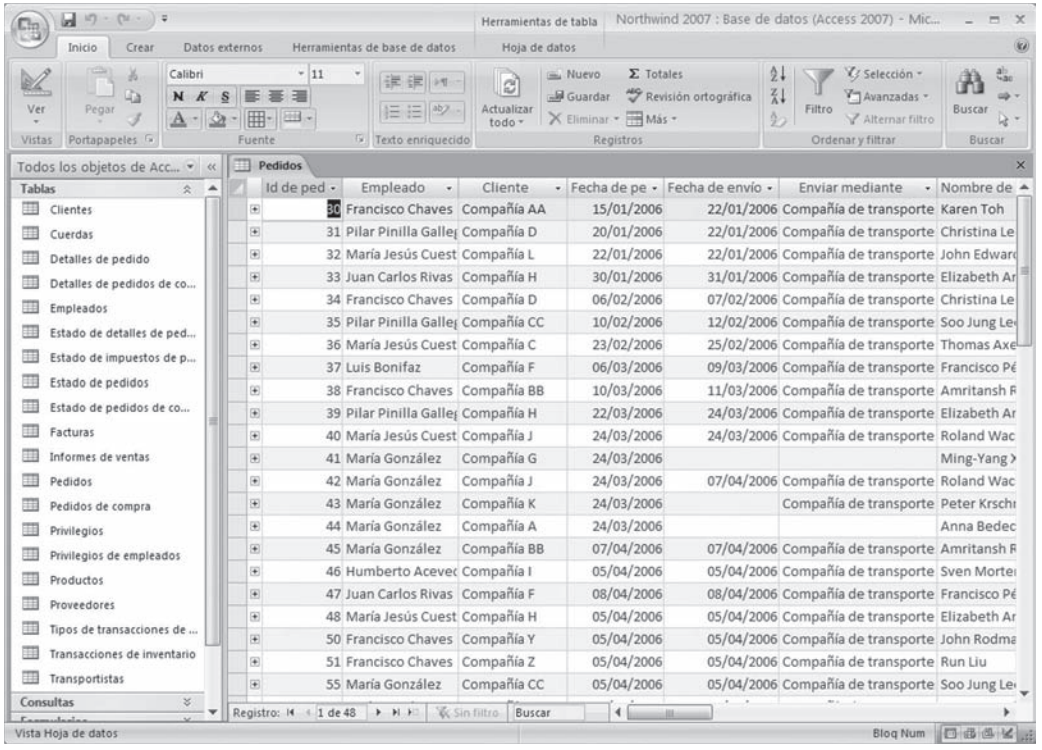


Figura 2-6 La tabla Pedidos de Northwind (lista parcial).

la entidad real que representan. Por convención, algunos DBA siempre asignan nombres en singular a las entidades y en plural a las tablas, y verá que esta convención se emplea en la base de datos Northwind. Lo importante aquí es que debe establecer normas de nomenclatura desde el principio para que los nombres no se asignen de manera casual, porque esto puede crear confusiones después. Cabe aclarar que Microsoft Access permite incrustar espacios en los nombres de tablas y columnas, lo que es contrario a las normas en la industria. Además, Microsoft Access, Sybase ASE y Microsoft SQL Server permiten nombres combinados, como DetallesDePedido, mientras que Oracle, DB2, MySQL en Windows y otros imponen que todos los nombres estén en mayúsculas, a menos que estén entre comillas. Como no es fácil leer nombres de tabla como DETALLESDEPEDIDO, una opción más conveniente, de acuerdo con las normas industriales, es utilizar guiones bajos para separar las palabras. Es posible que quiera establecer normas que prohíban el uso de nombres con espacios incrustados y mayúsculas y minúsculas intercaladas, porque esos nombres son contrarios a las normas y vuelven mucho más difícil una conversión entre distintos tipos de bases de datos.

Pregunta al experto

P: **Mencionó archivos y espacios de tabla: ¿son lo mismo?**

R: Puede considerar que un espacio de tabla es como un archivo lógico que forma una capa de abstracción entre las capas física y lógica, con lo que proporciona una mayor independencia lógica de los datos. Un espacio de tabla tiene uno o más archivos físicos asignados. Y en lugar de asignar tablas a archivos físicos, las asigna a espacios de tabla. Esto aporta mayor flexibilidad en el manejo de los archivos físicos que forman la base de datos. Por ejemplo, cuando los espacios de tabla comienzan a llenarse, una opción para el DBA es agregar otro archivo en un dispositivo diferente (como un disco duro).

Columnas y tipos de datos

Como se mencionó, cada columna de una tabla relacional representa un atributo del modelo conceptual. La *columna* es la unidad de datos con nombre más pequeña a la que se puede hacer referencia en una base de datos relacional. A cada columna debe asignarse un nombre único (dentro de la tabla) y un tipo de datos. Un *tipo de datos* es una categoría para el formato de una columna específica. Los tipos de datos proporcionan varios beneficios valiosos:

- Limitar los datos de la columna a caracteres que tienen sentido para el tipo de datos (por ejemplo, todos los dígitos numéricos o sólo fechas válidas).
- Ofrecer un conjunto de conductas útiles para el usuario de la base de datos. Por ejemplo, si resta un número de otro, obtiene un número como resultado; pero si resta una fecha de otra, el resultado es una cantidad que representa los días transcurridos entre las dos fechas.
- Ayudar al RDBMS a guardar de manera eficiente los datos de la columna. Por ejemplo, los números suelen guardarse en un formato numérico interno que ahorra espacio, comparado con guardar los dígitos numéricos como una cadena de caracteres.

En la figura 2-7 se muestra la definición de la tabla Pedidos, de Northwind, de Microsoft Access 2007 (la misma tabla presentada en la figura 2-6). El tipo de datos de cada columna aparece en la segunda columna. Los nombres de los tipos de datos suelen ser evidentes, pero si alguno le resulta confuso, consulte sus definiciones en las páginas de ayuda de Microsoft Access.

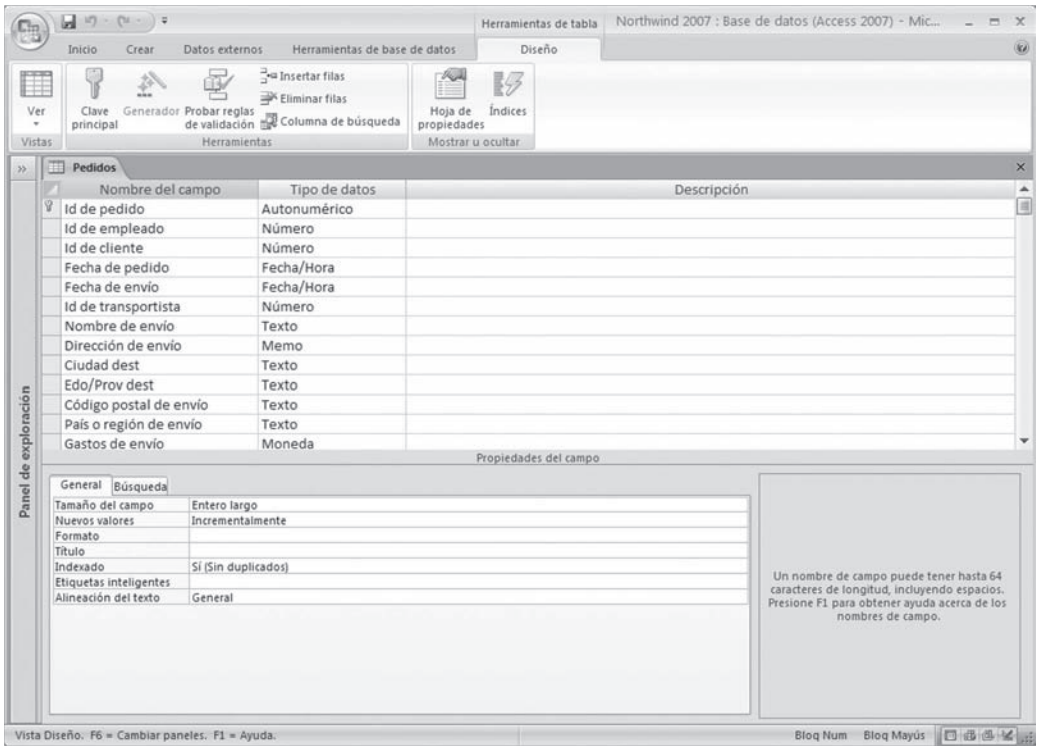


Figura 2-7 Definición de la tabla Pedidos, de Northwind (Microsoft Access 2007).

NOTA

Si compara las figuras 2-6 y 2-7, observará que en la figura 2-6 se muestran Empleado y Cliente en lugar de Id de empleado e Id de cliente, según se especifica en la definición de la figura 2-7. Esto no es un error, sino una función de Microsoft Access que se explica en la sección "Restricciones referenciales", más adelante en el capítulo.

Es muy desafortunado que las normas industriales estén a la zaga del desarrollo de los RDBMS. Casi todos los vendedores hicieron las cosas a su manera durante muchos años, antes de reunirse con otros para crear normas, y esto es evidente en la gran variedad de opciones de tipos de datos entre los principales productos de RDBMS. En la actualidad, las normas SQL de ANSI/ISO cubren los tipos de datos relacionales y los principales vendedores permiten todos o casi todos los tipos comunes. Sin embargo, cada vendedor tiene sus propias "extensiones" para las normas, en buena medida para permitir los tipos de datos desarrollados antes de que existieran las normas, pero también para agregar funciones que diferencien su

producto de los de los competidores. Es posible decir (en broma) que lo mejor de las normas para bases de datos es que hay muchas para elegir. En cuanto a las normas industriales para las bases de datos relacionales, es probable que Microsoft Access sea la que menos se apegue a los productos más populares. Debido a los numerosos niveles de cumplimiento de normas y todas las extensiones de vendedores, el DBA debe tener un conocimiento detallado de los tipos de datos disponibles en el DBMS específico que emplea para desplegar con éxito la base de datos. Y, por supuesto, se requiere mucho cuidado al convertir diseños lógicos del producto de un vendedor en otro.

En la tabla 2-1 se presentan los tipos de datos de diferentes vendedores de RDBMS que son, a grandes rasgos, equivalentes entre sí. Como siempre, la dificultad está en los detalles, lo que significa que éstos no son tipos de datos *idénticos*, sólo equivalentes. Por ejemplo, el tipo VARCHAR en Oracle puede tener hasta 4 000 caracteres de longitud (2 000 caracteres en versiones anteriores a Oracle8*i*), pero el tipo MEMO equivalente en Microsoft Access puede tener hasta 1 gigabyte de caracteres (casi mil millones de caracteres).

Restricciones

Una *restricción* es una regla aplicada a un objeto de la base de datos (por lo general una tabla o columna) que de algún modo limita los valores de datos permitidos para ese objeto de la base de datos. Son muy importantes en las bases de datos relacionales porque las restricciones son el modo en que se implementan las relaciones y las reglas de negocios especificadas en el diseño lógico. Se asigna un nombre único a cada restricción para permitir que se haga referencia a ella en los mensajes de error y en los comandos siguientes de la base de datos. Es una buena costumbre que los DBA proporcionen los nombres de restricciones, porque los generados automáticamente por los RDBMS nunca son muy descriptivos.

Tipos de datos	Microsoft Access	Microsoft SQL Server	Oracle
Carácter de longitud fija	TEXT	CHAR	CHAR
Carácter de longitud variable	MEMO	VARCHAR	VARCHAR
Texto extenso	MEMO	TEXT	CLOB o LONG (obsoleto)
Entero	INTEGER o LONG INTEGER	INTEGER o SMALLINT o TINYINT	NUMBER
Decimal	NUMBER	DECIMAL o NUMERIC	NUMBER
Moneda	CURRENCY	MONEY o SMALLMONEY	Ninguno, utilice NUMBER
Fecha/hora	DATE/TIME	DATETIME o SMALLDATETIME	DATE o TIMESTAMP

Tabla 2-1 Tipos de datos equivalentes en los principales productos RDBMS.

Restricciones de clave principal

Una *clave principal* es una columna o un conjunto de columnas que identifican de manera inequívoca a cada fila de una tabla. Por lo tanto, en el diseño conceptual se implementa un identificador único como clave principal en el diseño lógico. El ícono pequeño que parece una llave de puerta a la izquierda del nombre de campo Id de pedido, en la figura 2-7, indica que esta columna se ha definido como la clave principal de la tabla Pedidos. Cuando se define una clave principal, el RDBMS la implementa como *restricción de clave principal* para garantizar que dos filas de la tabla no tengan valores duplicados en las columnas de claves principales. Tome en cuenta que, en el caso de claves principales formadas por varias columnas, cada columna *puede* tener valores duplicados en la tabla, pero la *combinación* de los valores de todas las columnas de claves principales debe ser única entre todas las filas de la tabla.

Las restricciones de clave principal casi siempre son implementadas por el RDBMS mediante un *índice*, que es un tipo especial de objeto de base de datos que permite búsquedas rápidas de valores de columna. Conforme se insertan filas nuevas en la tabla, el RDBMS busca *automáticamente* el índice para comprobar que el valor de la clave principal en la fila nueva no se está usando en la tabla, y rechaza la solicitud de inserción, si lo está. Es posible buscar mucho más rápido en los índices que en las tablas; por lo tanto, el índice de la clave principal es esencial en tablas de cualquier tamaño para que no se genere un cuello de botella de rendimiento cuando se busquen claves duplicadas en cada inserción.

Restricciones referenciales

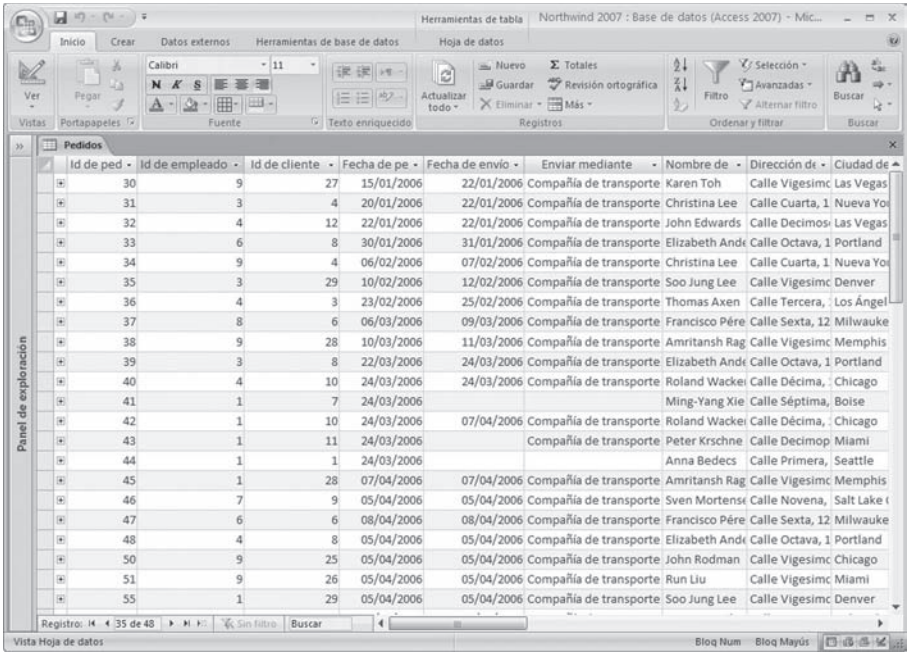
Para comprender la manera en que el RDBMS impone que las relaciones utilicen restricciones referenciales, primero debe comprender el concepto de claves externas. Cuando se implementan relaciones uno a varios en tablas, a la columna o el conjunto de columnas que se guardan en la tabla secundaria (la tabla en el lado "varios" de la relación), para asociarla con la tabla primaria (la tabla en el lado "uno"), se le denomina *clave externa*. Recibe su nombre de las columnas copiadas de otra tabla (externa). En la tabla Pedidos presentada en la figura 2-6, la columna Id de empleado es una clave externa para la tabla Empleado, y la columna Id de cliente es una clave externa para la tabla Clientes.

En casi todas las bases de datos relacionales, la clave externa debe ser la clave principal de la tabla principal o una columna o un conjunto de columnas del que se define un índice único. De nuevo, esto está en función de la eficiencia. Casi todas las personas prefieren que las columnas de clave externa tengan nombres idénticos a las columnas de clave principal correspondientes, pero otra vez hay opiniones encontradas, sobre todo porque las columnas con nombres similares son un poco más difíciles de emplear en los lenguajes de consulta. Es mejor establecer algunas normas antes y respetarlas durante su proyecto de bases de datos.

En el diseño conceptual, cada relación entre las entidades se convierte en una restricción referencial en el diseño lógico. Una *restricción referencial* (también denominada *restricción de integridad referencial*) es una restricción que impone una relación entre las tablas de una

base de datos relacional. *Impone* significa que el RDBMS comprueba automáticamente que cada valor de clave externa en una tabla secundaria siempre tenga un valor de clave principal correspondiente en la tabla principal.

Microsoft Access proporciona una función agradable para las columnas de clave externa, pero se requiere un poco de tiempo para acostumbrarse a ella. Cuando define una restricción referencial, puede definir una búsqueda automática de las filas de la tabla principal, como se hizo en toda la base de datos Northwind. En la figura 2-7, la tercera columna de la tabla aparece como Id de cliente. No obstante, en la figura 2-6, observará que la tercera columna de la tabla Pedidos expone el nombre del cliente y aparece como Cliente. Si hace clic en la columna Cliente de una de las filas, aparece un menú desplegable para permitirle elegir un cliente válido (de la tabla Clientes) para que sea el elemento primario (propietario) de la fila seleccionada de la tabla Pedidos. Asimismo, la columna Id de empleado de la tabla muestra el nombre del empleado. Esto es una función conveniente y fácil para el usuario de la base de datos, y evita que un cliente o un empleado que no existe se asocie con un pedido. Sin embargo, oculta la clave externa de manera que la figura 2-6 no es muy útil para ejemplificar cómo funcionan las restricciones referenciales en segundo plano. En la figura 2-8 se presenta la tabla Pedidos sin las búsquedas, para que vea los valores reales de la clave externa en las columnas Id de empleado e Id de cliente.



Id de ped	Id de empleado	Id de cliente	Fecha de pe	Fecha de envío	Enviar mediante	Nombre de	Dirección de	Ciudad de
30	9	27	15/01/2006	22/01/2006	Compañía de transporte	Karen Toh	Calle Vigesimc	Las Vegas
31	3	4	20/01/2006	22/01/2006	Compañía de transporte	Christina Lee	Calle Cuarta, 1	Nueva Yo
32	4	12	22/01/2006	22/01/2006	Compañía de transporte	John Edwards	Calle Decimos	Las Vegas
33	6	8	30/01/2006	31/01/2006	Compañía de transporte	Elizabeth Ande	Calle Octava, 1	Portland
34	9	4	06/02/2006	07/02/2006	Compañía de transporte	Christina Lee	Calle Cuarta, 1	Nueva Yo
35	3	29	10/02/2006	12/02/2006	Compañía de transporte	Soo Jung Lee	Calle Vigesimc	Denver
36	4	3	23/02/2006	25/02/2006	Compañía de transporte	Thomas Axen	Calle Tercera,	Los Ángel
37	8	6	06/03/2006	09/03/2006	Compañía de transporte	Francisco Pére	Calle Sexta, 12	Milwauke
38	9	28	10/03/2006	11/03/2006	Compañía de transporte	Amritansh Rag	Calle Vigesimc	Memphis
39	3	8	22/03/2006	24/03/2006	Compañía de transporte	Elizabeth Ande	Calle Octava, 1	Portland
40	4	10	24/03/2006	24/03/2006	Compañía de transporte	Roland Wacke	Calle Décima,	Chicago
41	1	7	24/03/2006			Ming-Yang Xie	Calle Séptima,	Boise
42	1	10	24/03/2006	07/04/2006	Compañía de transporte	Roland Wacke	Calle Décima,	Chicago
43	1	11	24/03/2006		Compañía de transporte	Peter Krschne	Calle Decimop	Miami
44	1	1	24/03/2006			Anna Bedecs	Calle Primera,	Seattle
45	1	28	07/04/2006	07/04/2006	Compañía de transporte	Amritansh Rag	Calle Vigesimc	Memphis
46	7	9	05/04/2006	05/04/2006	Compañía de transporte	Sven Mortense	Calle Novena,	Salt Lake
47	6	6	08/04/2006	08/04/2006	Compañía de transporte	Francisco Pére	Calle Sexta, 12	Milwauke
48	4	8	05/04/2006	05/04/2006	Compañía de transporte	Elizabeth Ande	Calle Octava, 1	Portland
50	9	25	05/04/2006	05/04/2006	Compañía de transporte	John Rodman	Calle Vigesimc	Chicago
51	9	26	05/04/2006	05/04/2006	Compañía de transporte	Run Liu	Calle Vigesimc	Miami
55	1	29	05/04/2006	05/04/2006	Compañía de transporte	Soo Jung Lee	Calle Vigesimc	Denver

Figura 2-8 La tabla Pedidos de Northwind (con los valores de clave externa exhibidos).

Cuando actualizamos la tabla Pedidos, tal como se presenta en la figura 2-8, el RDBMS debe imponer las restricciones referenciales definidas para la tabla. Lo bueno de las restricciones de una base de datos es que son *automáticas* y, por lo tanto, no pueden evitarse a menos que el DBA las elimine o las inhabilite.

He aquí los eventos específicos que el RDBMS debe manejar al imponer las restricciones referenciales:

- Cuando se intenta insertar una fila nueva en la tabla secundaria, la solicitud de inserción es rechazada si no existe la fila correspondiente en la tabla principal. Por ejemplo, si inserta una fila en la tabla Pedidos con una Id de empleado 12345, el RDBMS debe revisar la tabla Empleados para ver si ya existe una fila para la Id de empleado 12345. Si no existe, se rechaza la solicitud de inserción.
- Cuando pretende actualizar el valor de una clave externa en la tabla secundaria, la solicitud de actualización es rechazada si el valor nuevo de la clave externa no existe todavía en la tabla principal. Por ejemplo, si intenta modificar la Id de empleado para el pedido 48 de 4 a 12345, el RDBMS debe volver a comprobar la tabla Empleados para ver si ya existe una fila para la Id de empleado 12345. Si no existe, la solicitud de actualización es rechazada.
- Cuando se propone eliminar una fila de una tabla principal, y esa fila tiene filas relacionadas en una o más tablas secundarias, las filas de la tabla secundaria deben eliminarse junto con la fila primaria o la solicitud de eliminación será rechazada. Casi todos los RDBMS ofrecen la opción de eliminar automáticamente las filas secundarias, a lo que se le conoce como *eliminación en cascada*. Al principio, es probable que se pregunte por qué alguien necesitaría una eliminación automática de las filas secundarias. Considere las tablas Pedidos y Detalles de pedido. Si se va a eliminar un pedido, ¿por qué no borrar el pedido y los artículos de línea que incluye en un solo paso fácil? Sin embargo, con la tabla Empleados, es evidente que no necesita esa opción. Si intenta eliminar el empleado 4 de la tabla Empleados (tal vez porque la persona ya no es empleado), el RDBMS debe comprobar las filas asignadas a la Id de empleado 4 en la tabla Pedidos, y rechazar la solicitud de eliminación si encuentra alguna. En un negocio no tiene sentido eliminar automáticamente los pedidos cuando un empleado abandona la compañía.

En casi todas las bases de datos relacionales, se emplea una instrucción SQL para definir una restricción referencial. SQL se presenta en el capítulo 4. SQL es el lenguaje utilizado en los RDBMS para comunicarse con la base de datos. Muchos vendedores también ofrecen paneles de una interfaz gráfica de usuario (GUI. Graphical User Interface) para definir los objetos de una base de datos, como las restricciones referenciales. Por ejemplo, en SQL Server estos paneles GUI están dentro de la herramienta Management Studio, y en Oracle, una herramienta llamada SQL Developer tiene estas capacidades. En el caso de

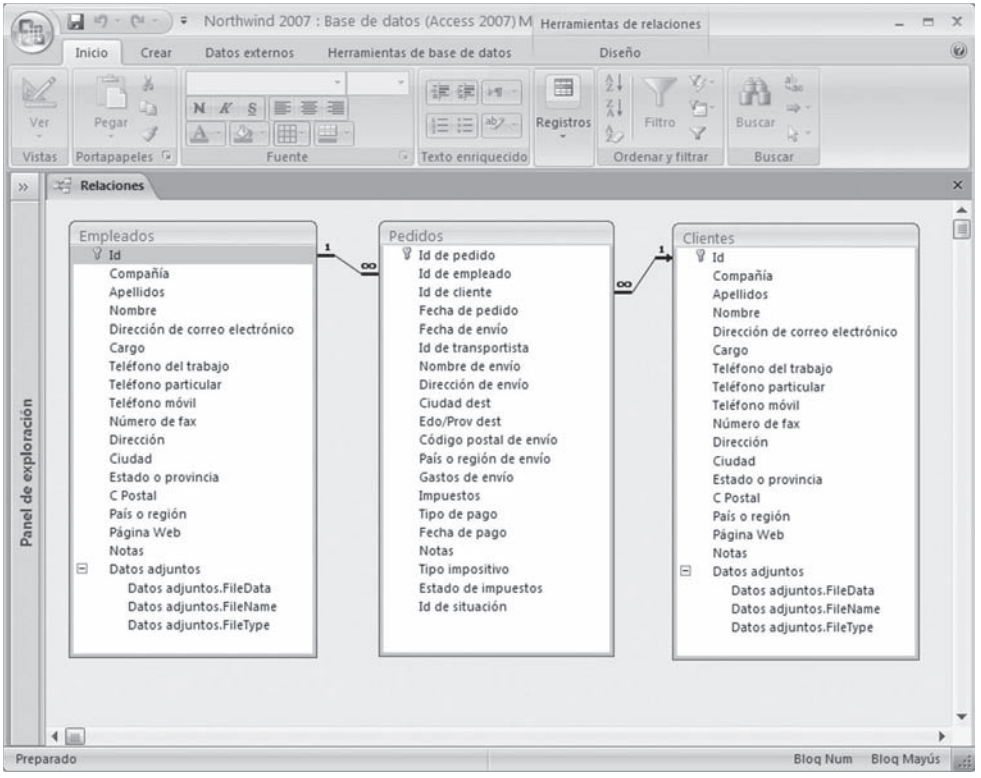


Figura 2-9 El panel Relaciones, de Microsoft Access 2007.

Microsoft Access, en la figura 2-9 muestra el panel Relaciones, que sirve para definir las restricciones referenciales.

Para mayor simplicidad, en la figura 2-9 sólo se muestra la tabla Pedidos y sus dos tablas primarias, Empleados y Clientes. Las restricciones referenciales se presentan como líneas gruesas con el símbolo numérico *1* cerca de la tabla principal (el lado "uno") y el símbolo matemático para infinito (una especie de 8 de costado) cerca de la tabla secundaria (el lado "varios"). Estas restricciones se definen con sólo arrastrar el nombre de la clave principal en la tabla principal al nombre de la clave externa en la tabla secundaria. Luego se exhibe automáticamente una ventana emergente con el fin de permitir la definición de las opciones para la restricción referencial, igual que la figura 2-10.

En la parte superior del panel Modificar relaciones aparecen los nombres de las dos tablas, con la tabla principal en el lado izquierdo y la tabla secundaria en el lado derecho. Si olvida cuál es cuál, el campo Tipo de relación que se encuentra en la parte inferior del panel

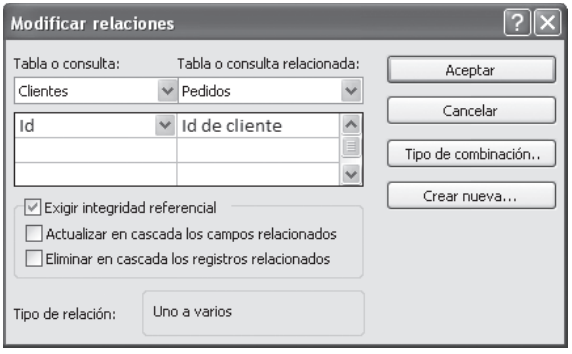


Figura 2-10 El panel Modificar relaciones, de Microsoft Access 2007.

se lo recuerda. Bajo cada nombre de tabla hay filas para elegir los nombres de columna que forman la clave principal y la clave externa. En la figura 2-10 se muestra la Id de columna de la clave principal de la tabla Clientes y la Id de cliente de la columna de clave externa en la tabla Pedidos. Las casillas de verificación ofrecen algunas opciones:

- **Exigir integridad referencial** Si selecciona esta casilla, se impone la restricción; al dejarla en blanco, se desactiva la exigencia de la restricción.
- **Actualizar en cascada los campos relacionados** Si pone una marca en esta casilla, cualquier actualización en el valor de la clave principal de la tabla principal generará actualizaciones automáticas correspondientes en los valores relacionados de clave externa. Una actualización de los valores de la clave principal es una situación rara.
- **Eliminar en cascada los registros relacionados** Si elige esta casilla, la eliminación de una fila en la tabla principal generará una eliminación en cascada automática de las filas relacionadas en la tabla secundaria. Pondere esta situación. En ocasiones debe aplicar esto, como en la restricción entre Pedidos y Detalles de pedido, y otras veces la opción puede provocar una desastrosa y no deseada pérdida de datos, como al eliminar un empleado (tal vez por accidente) y hacer que los pedidos que manejaba ese empleado se eliminen automáticamente de la base de datos.

Tablas de intersección

Durante el análisis de las relaciones varios a varios al inicio del capítulo se indicó que las bases de datos relacionales no pueden implementar estas relaciones directamente y que se forma una tabla de intersección para establecerlas. En la figura 2-11 se muestra la implementación de la tabla de intersección Detalles de pedido en Microsoft Access.

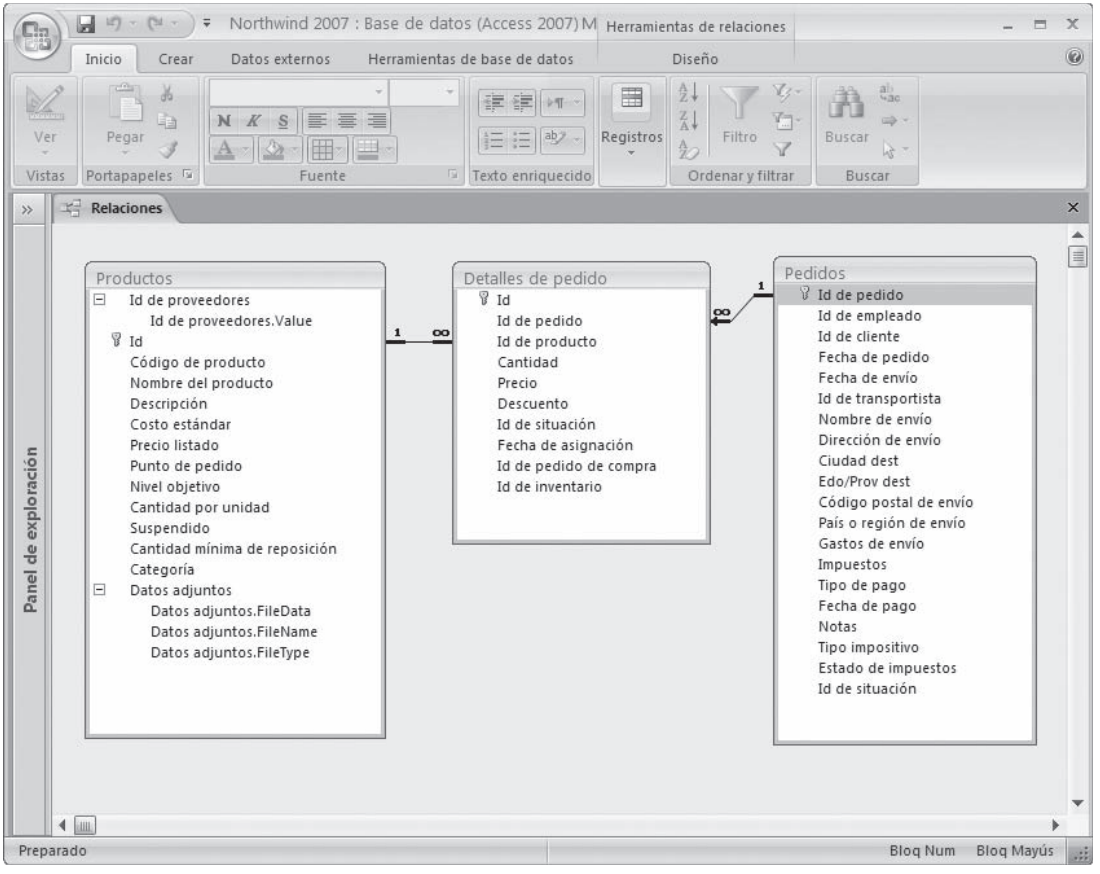


Figura 2-11 La tabla de intersección Detalles de pedido en Microsoft Access 2007.

La relación varios a varios entre pedidos y productos en el diseño conceptual se convierte en una tabla de intersección (Detalles de pedido) en el diseño lógico. Después, la relación se implementa como dos relaciones uno a varios con la tabla de intersección en el lado "varios" de cada una. La clave principal de la tabla Detalles de pedido pudo formarse mediante la combinación de Id de pedido e Id de producto, en donde Id de pedido es una clave externa para la tabla Pedidos e Id de producto es una clave externa para la tabla Productos. Sin embargo, en este caso el diseñador eligió agregar sólo un valor de clave única, Id, como la clave principal de la tabla Detalles de pedido. A esta distribución se le conoce como *clave sustituta*, porque la llamada *clave natural* ha sido reemplazada con otra. Dedique un momento a examinar el contenido de la tabla de intersección y las dos restricciones referenciales. Comprender esta

distribución es fundamental para entender cómo funcionan las bases de datos relacionales. He aquí algunos puntos que debe tomar en cuenta:

- Cada fila de la tabla de intersección Detalles de pedido pertenece a la intersección de un producto y un pedido. No tendría sentido incluir Nombre de producto en esta tabla, porque ese nombre es el mismo cada vez que el producto aparece en un pedido. Asimismo, no tendría sentido incluir Id de cliente y Detalles de pedido, porque todos los artículos de línea del mismo pedido pertenecen al mismo cliente.
- Cada fila de la tabla Productos puede tener muchas filas Detalles de pedido relacionadas (una para cada artículo de línea del pedido en que se solicitó el producto), pero cada fila Detalles de pedido pertenece a una y sólo a una fila de la tabla Productos.
- Cada fila de la tabla Pedidos puede tener muchas filas Detalles de pedido relacionadas (una para cada artículo de línea de ese pedido específico), pero cada fila Detalles de pedido pertenece a una y sólo una fila de la tabla Pedidos.

Restricciones de integridad

Como ya se mencionó, las reglas de negocios del diseño conceptual se vuelven restricciones en el diseño lógico. Una *restricción de integridad* promueve la exactitud de los datos de la base de datos. El beneficio principal es que el RDBMS invoca automáticamente estas restricciones y no pueden evitarse (a menos que usted sea el DBA), sin importar cómo se conecte a la base de datos. Los principales tipos de restricciones de integridad son NOT NULL (no nulo), CHECK (comprobar) e impuestas con encadenadores.

Restricciones NOT NULL

Mientras define las columnas de las tablas de la base de datos, tiene la opción de especificar si se permiten valores nulos para la columna. Un *valor nulo* de una base de datos relacional es un código especial que puede colocarse en una columna y que indica que se desconoce el valor de esa columna en esa fila. Un valor nulo no es lo mismo que un espacio en blanco, una cadena de caracteres vacía o un cero; en realidad se trata de un código especial que no tiene otro significado en la base de datos.

Un modo uniforme de tratar los valores nulos se especifica en la norma SQL de ANSI/ISO. No obstante, existe un debate sobre la utilidad de la opción, porque la base de datos no puede informarle *por qué* se desconoce el valor. Por ejemplo, si deja nulo el valor de Cargo, en la tabla Empleados de Northwind, no sabe si es nulo porque en verdad se desconoce (sabe que el empleado debe tener un cargo, pero no lo conoce), no se aplica (tal vez algunos empleados no tienen cargos) o no está asignado (en algún momento obtendrá un cargo, pero su gerente todavía no ha determinado cuál). El otro dilema es que los valores nulos no son igua-

les a algo más, lo que incluye otros valores nulos, y esto introduce una lógica de tres valores en las búsquedas en la base de datos. Cuando se utilizan valores nulos, una búsqueda puede devolver la condición *verdadero* (el valor en la columna coincide), *falso* (el valor en la columna no coincide) o *desconocido* (el valor en la columna es nulo). Los desarrolladores que escriben los programas de aplicación deben manejar los valores nulos como un caso especial. Aprenderá más sobre los valores nulos cuando conozca SQL en el capítulo 4.

En Microsoft Access, la restricción NOT NULL es controlada por la opción Requerido, en el panel de diseño de tabla. En la figura 2-12 se presenta la definición de la columna Descuento, de la tabla Detalles de pedido. Observe que la columna es obligatoria (es decir, no puede ser nula) porque la opción Requerido tiene asignada la opción Sí. En las definiciones de tablas de SQL, simplemente se incluye la palabra clave NULL o NOT NULL en la definición de columna. Tenga cuidado con los valores predeterminados. En Oracle, si pasa por alto la especificación, lo predeterminado es NULL, lo que significa que la columna puede

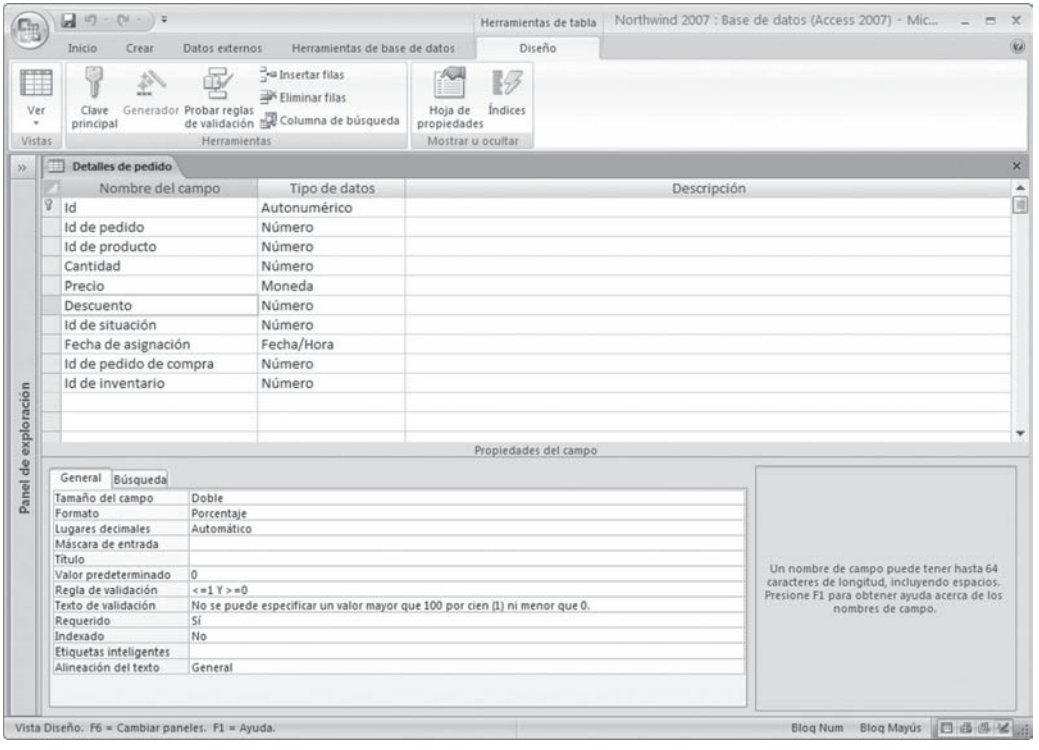


Figura 2-12 Panel de definición de la tabla Detalles de pedido, columna Descuento.

contener valores nulos. Pero en algunas implementaciones de DB2, Microsoft SQL Server y Sybase ASE, ocurre lo contrario: si pasa por alto la especificación, lo predeterminado es NOT NULL, lo que significa que *no puede* contener valores nulos.

Restricciones CHECK

Una restricción CHECK utiliza una instrucción lógica simple para validar un valor de columna. El resultado de la instrucción debe ser un *verdadero* o *falso* lógico, en donde *verdadero* permite que el valor de la columna se coloque en la tabla, y *falso* hace que el valor de la columna sea rechazado con un mensaje de error adecuado. En la figura 2-12, observe que *<=1 Y >=0* aparece en la opción Regla de validación de la columna Descuento. Esta regla evita que los descuentos sean mayores de 100 por ciento (introducidos como 1.00) o menores de 0 por ciento. Aunque la sintaxis de la opción puede variar en otras bases de datos, el concepto es el mismo. En Oracle SQL, se escribiría de esta manera:

```
CHECK (DISCOUNT <=1 AND DISCOUNT >=0)
```

Restricciones impuestas con desencadenadores

Algunas restricciones son demasiado complicadas para imponerse mediante declaraciones. Por ejemplo, la regla de negocios incluida en la figura 2-1 (los Clientes con saldo vencido no pueden hacer pedidos nuevos) cae en esta categoría, porque abarca más de una tabla. Necesitamos evitar que se agreguen filas nuevas a la tabla Pedidos, si la fila Cuentas por cobrar del cliente tiene un saldo vencido mayor que cero. Como ya se mencionó, tal vez sea mejor implementar reglas de negocios como ésta en la lógica de la aplicación. Sin embargo, si queremos agregar una restricción que será impuesta sin importar cómo se actualiza la base de datos, un desencadenador cumplirá esa misión. Un *desencadenador* es un módulo de lógica de programación que se "enciende" (ejecuta) cuando ocurre un evento específico en la base de datos. En este ejemplo, queremos que el desencadenador se encienda cuando se inserte una fila nueva en la tabla Pedidos. El desencadenador obtiene el saldo vencido del cliente de la tabla Cuentas por cobrar (o de donde se guarde físicamente la columna). Si esta cantidad es mayor que cero, el desencadenador accionará un error de base de datos que detiene la solicitud de inserción y provoca que se muestre un mensaje de error adecuado.

En Microsoft Access, los desencadenadores se escriben como macros mediante el lenguaje Microsoft Visual Basic para Aplicaciones (VBA). Algunos RDBMS ofrecen un lenguaje especial para escribir módulos de programa como desencadenadores: PL/SQL en Oracle y Transact SQL en Microsoft SQL Server y Sybase ASE. En otros RDBMS, como DB2, se puede emplear un lenguaje de programación genérico, como C.

Vistas

Una *vista* es una consulta guardada que se proporciona al usuario de una base de datos con un subconjunto personalizado de los datos de una o más tablas. Dicho de otro modo, una vista es una *tabla virtual*, porque parece una tabla y, en su mayor parte, se comporta como una tabla, pero no almacena datos (sólo se guarda la consulta que los define). Las vistas de usuario forman la capa externa del modelo ANSI/SPARC. Durante el diseño lógico, cada vista se crea utilizando un método adecuado para la base de datos específica. En muchos RDBMS, una vista se define mediante SQL. En Microsoft Access, las vistas no se permiten directamente. Sin embargo, Access permite un tipo de objeto equivalente, llamado *consulta*, que se crea mediante el panel Consulta. En la figura 2-13 se muestra la definición en Microsoft Access de una vista sencilla que presenta los pedidos realizados por los clientes que viven en el estado de Madrid.

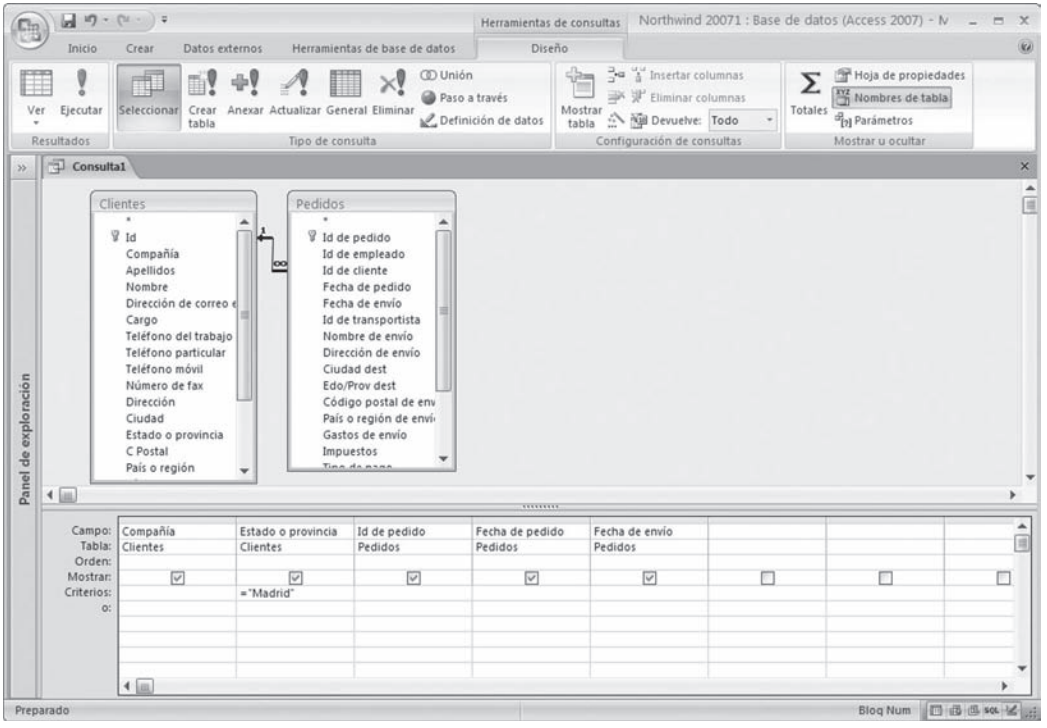


Figura 2-13 Definición de vista en Microsoft Access 2007: presentar todos los pedidos de clientes residentes en Madrid.

La vista de la figura 2-13 sólo muestra dos columnas de la tabla Clientes, junto con tres columnas de la tabla Pedidos. Además, la vista especifica la coincidencia (unión) de las tablas Clientes y Pedidos y *filtra* las filas para que sólo se incluyan los pedidos para los clientes en Madrid, en virtud del valor en la propiedad Criterios de la columna Estado o provincia (=*'Madrid'*). El panel Consulta de Microsoft Access se explora con detalle en el capítulo 3. En la figura 2-14 se presentan los resultados de la consulta cuando se ejecuta contra la base de datos. Aunque dos clientes viven en Madrid, sólo uno de ellos ha hecho pedidos, y sólo dos de esos pedidos aparecen en la tabla.

Las vistas tienen varias funciones útiles:

- Ocultar columnas que el usuario no necesita ver (o no está autorizado para ver).
- Ocultar filas de tablas que un usuario no necesita ver (o no está autorizado para ver).
- Ocultar operaciones complejas de la base de datos, como uniones de tablas.
- Mejorar el desempeño de una consulta (en algunos RDBMS, como Microsoft SQL Server).

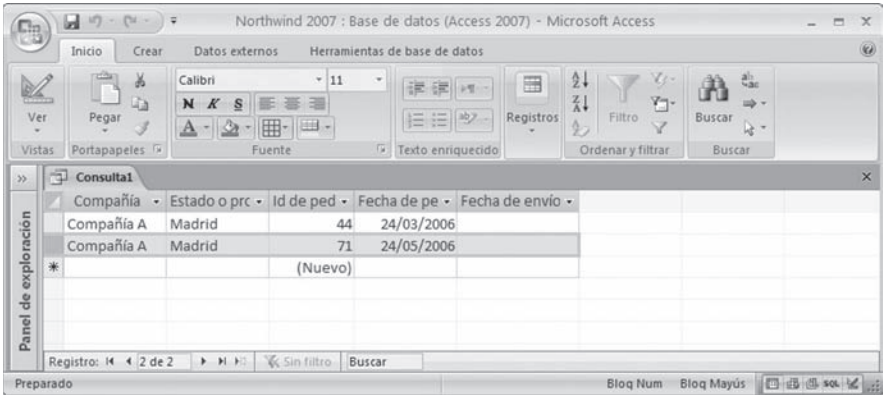


Figura 2-14 Resultados de la ejecución de la consulta presentada en la figura 2-13.

Autoexamen Capítulo 2

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. Ejemplos de una entidad son

- **A** Un cliente.
- **B** Un pedido de clientes.
- **C** El cheque de pago de un empleado.
- **D** El nombre de un cliente.

2. Ejemplos de un atributo son

- **A** Un empleado.
- **B** El nombre de un empleado.
- **C** El cheque de pago de un empleado.
- **D** La lista alfabética de empleados.

3. ¿Cuál de los siguientes elementos denota la cardinalidad de "cero, uno o más" en una línea de relación?

- **A** Una línea perpendicular cerca del extremo de la línea y una pata de gallo en el extremo de la línea.
- **B** Un círculo cerca del extremo de la línea y una pata de gallo en el extremo de la línea.
- **C** Dos líneas perpendiculares cerca del extremo de la línea.
- **D** Un círculo y una línea perpendicular cerca del extremo de la línea.

4. Los tipos válidos de relaciones en una base de datos relacional son

- **A** Uno a varios.
- **B** Ninguno a varios.
- **C** Varios a varios.
- **D** Uno a uno.

5. Si un producto se puede fabricar en varias plantas, y una planta puede fabricar varios productos, ¿éste es un ejemplo de qué tipo de relación?

A Uno a uno.

B Uno a varios.

C Varios a varios.

D Recursiva.

6. Entre los siguientes, ¿cuáles son ejemplos de relaciones recursivas?

A Una unidad organizativa formada por departamentos.

B Un empleado que dirige a otros empleados.

C Un empleado que dirige un departamento.

D Un empleado que tiene muchos dependientes.

7. Ejemplo de una regla de negocios es

A Una restricción referencial debe hacer referencia a la clave principal de la tabla principal.

B Un empleado debe tener cuando menos 18 años de edad.

C Una consulta de base de datos elimina las columnas que un empleado no debe ver.

D No se permite modificar pedidos a los empleados con una calificación de pago menor de 6.

8. Una tabla relacional

A Está formada por filas y columnas.

B Debe tener asignado un tipo de datos.

C Debe tener asignado un nombre único.

D Es la unidad de almacenamiento principal del modelo relacional.

9. Una columna de una tabla relacional

A Debe tener asignado un tipo de datos.

B Debe tener asignado un nombre único dentro de la tabla.

C Se deriva de una entidad en el diseño conceptual.

D Es la unidad de almacenamiento con nombre más pequeña de una base de datos relacional.

10. Un tipo de datos

- **A** Ayuda al DBMS a guardar los datos de manera eficiente.
- **B** Ofrece un conjunto de comportamientos para una columna que ayuda al usuario de la base de datos.
- **C** Puede ser seleccionada con base en las reglas de negocios de un atributo.
- **D** Limita los caracteres permitidos en la columna de una base de datos.

11. Una restricción de clave principal

- **A** Debe hacer referencia a una o más columnas de una sola tabla.
- **B** Debe definirse para cada tabla de una base de datos.
- **C** Suele implementarse mediante un índice.
- **D** Garantiza que dos filas de una tabla no tengan valores de clave principal duplicados.

12. Una restricción referencial

- **A** Debe tener columnas de clave principal y clave externa con nombres idénticos.
- **B** Asegura que una clave principal no tenga valores duplicados en una tabla.
- **C** Define una relación varios a varios entre dos tablas.
- **D** Asegura que un valor de clave externa siempre haga referencia a un valor existente de clave principal en la tabla principal.

13. Una restricción referencial se define

- **A** Mediante el panel Relaciones en Microsoft Access.
- **B** Mediante SQL en casi todas las bases de datos relacionales.
- **C** Mediante el tipo de datos referencial de las columnas de clave externa.
- **D** Mediante un desencadenador de base de datos.

14. Los tipos principales de restricciones de integridad son

- **A** Las restricciones de comprobación (CHECK).
- **B** Las relaciones uno a uno.
- **C** Las restricciones NOT NULL.
- **D** Las restricciones impuestas con desencadenadores.

15. Las tablas _____ se emplean para resolver las relaciones varios a varios.

16. Una entidad en el diseño conceptual se vuelve _____ en el diseño lógico.

17. Un atributo en el diseño conceptual se vuelve _____ en el diseño lógico.

18. Los elementos en el nivel externo del modelo ANSI/SPARC se vuelven _____ en el modelo lógico.

19. Una relación en el diseño conceptual se vuelve _____ en el modelo lógico.

20. Una restricción de clave principal se implementa mediante _____ en el diseño lógico.

Capítulo 3

Consultas a bases de datos con formularios

Habilidades y conceptos clave

- QBE: las raíces de las consultas mediante formularios
- Introducción a Microsoft Access
- El panel Relaciones de Microsoft Access
- Creación de consultas en Microsoft Access

Con base en la teoría de que no es posible diseñar un automóvil si nunca se ha conducido uno, este capítulo ofrece una breve descripción de las consultas a una base de datos, antes de abundar en los detalles del diseño de una base de datos. En este capítulo se proporciona un compendio sobre la formación y ejecución de consultas en una base de datos mediante la herramienta de consultas basadas en formularios de Microsoft Access. No se pretende en absoluto ofrecer una guía detallada de Microsoft Access; sólo se utiliza como un vehículo para presentar los conceptos de las consultas en una base de datos que serán los cimientos para la teoría del diseño de una base de datos que se cubre más adelante en el libro. Sin embargo, la intención es proporcionarle suficiente información básica sobre el uso de Microsoft Access para que pueda avanzar en su propio equipo de cómputo mientras explora las consultas basadas en formularios.

QBE: las raíces de las consultas mediante formularios

Un lenguaje de consultas *mediante formularios* emplea un panel GUI para la creación de una consulta. El usuario de una base de datos define las consultas al introducir valores de datos de ejemplo directamente en una plantilla de consultas para representar el resultado que va a obtener la base de datos. Un método alterno emplea un lenguaje de consultas *basado en comandos*, en que las consultas se escriben como comandos de texto. SQL es el lenguaje universal de las consultas basadas en comandos para las bases de datos relacionales y se analiza en el capítulo 4. En los lenguajes de consultas mediante formularios y comandos se hace hincapié en *qué* debe ser el resultado, y no en *cómo* se consigue éste. La diferencia entre los dos es el modo en que el usuario describe el resultado deseado; es similar a la diferencia entre usar el Explorador de Windows para copiar un archivo en comparación con usar el comando **copy** de MS-DOS (en la ventana de comandos de DOS) para hacer lo mismo.

Pregunta al experto

P: **Ha mencionado consultas mediante comandos y formularios. No me queda claro en el aprendizaje de cuál debo concentrarme.**

R: Para definir cuál debe aprender primero debe tomar en cuenta lo que quiere hacer con la base de datos, y en algún momento tendrá que conocer ambos. Las consultas mediante comandos son esenciales si quiere incrustarlas en otro lenguaje de programación (no puede incrustar una consulta basada en formularios en otro lenguaje). Sin embargo, al preparar consultas *ad hoc*, las personas suelen preferir una GUI interactiva que les permita apuntar y hacer clic sobre los comandos de texto que requieren más escritura. En la década de 1970, IBM efectuó un estudio controlado para determinar si QBE o SQL era el preferido de los usuarios de una base de datos en esa época. IBM identificó que casi todos los usuarios preferían emplear el método que aprendieron primero; parece que así es la naturaleza humana.

La primera herramienta de consultas mediante formularios muy aceptada fue la consulta mediante el ejemplo (Query By Example, QBE), desarrollada por IBM en la década de 1970. En esa época no se conocían los equipos personales, Microsoft Windows, el ratón y muchos otros elementos modernos de los equipos de cómputo; no obstante, la interfaz era gráfica. Se mostraba un formulario, y los usuarios de las bases de datos escribían datos de ejemplo y comandos sencillos en cuadros, mientras que en la actualidad hacen clic en el botón de una pantalla mediante un ratón. SQL, también desarrollado al principio por IBM, una novedad en la década de 1970.

La experiencia ha demostrado que es útil conocer ambos métodos. Las consultas mediante formularios son convenientes para las personas que están más acostumbradas a ambientes GUI que a comandos escritos. Sin embargo, los usuarios de una base de datos que conocen la sintaxis de los comandos y poseen razonables habilidades de escritura pueden introducir consultas mediante comandos más rápido que sus equivalentes en una GUI, y las consultas mediante comandos se emplean directamente dentro de un lenguaje de programación como Java o C.

Introducción a Microsoft Access

Todas las consultas utilizadas en este capítulo presentan la base de datos de ejemplo Northwind ofrecida por Microsoft para utilizarse con Access o SQL Server. Obtendrá la mejor experiencia de aprendizaje si prueba las consultas presentadas en este capítulo mientras avanza en su lectura. Es obvio que se requiere la base de datos de ejemplo, y debe emplear Microsoft Access 2007 porque existen diferencias sustanciales entre esta versión y las anteriores, entre ellas la base de datos de ejemplo. Por suerte, le resultará relativamente sencillo descargar e instalar la base de datos de Northwind (si ya tiene instalado Access 2007) o conectarse a Microsoft Access 2007 de manera remota mediante Microsoft Office Online. Sólo siga los pasos del ejercicio Pruebe esto, en el capítulo 2 (si todavía no lo ha hecho). Recuerde que es fácil

actualizar la base de datos en forma accidental al utilizar Microsoft Access, y no existe ninguna función "deshacer" sencilla. Sin embargo, si ocurre esto, puede simplemente descargar otra vez la base de datos y volver al punto donde comenzó.

Cuando inicie Microsoft Access 2007 (ya sea en forma local o mediante Microsoft Office Online) verá un panel de inicio similar al de la figura 3-1.

Si ya ha descargado y utilizado la base de datos Northwind, debe estar listada bajo el encabezado Abrir base de datos reciente en el lado derecho del panel. Para abrir la base de datos simplemente haga clic en el nombre de archivo listado. Si la base de datos no aparece en la lista, puede descargarla al hacer clic en Ejemplo, bajo el encabezado Desde Microsoft Office Online, en el lado izquierdo del panel. Se expondrá un panel similar al mostrado en la figura 3-2. Haga clic en el ícono de Northwind 2007 para seleccionarlo y luego haga clic en el botón Descargar, en la esquina inferior derecha del panel.

Sabrá que ha podido conectarse con éxito a la base de datos Northwind cuando vea el panel principal de Microsoft Access 2007 que muestra la ficha Pantalla de inicio para Northwind Traders, igual que en la figura 3-3. Antes de analizar las opciones de este panel, organicemos algunas cosas. La base de datos de ejemplo viene con código de aplicación (macros de Visual

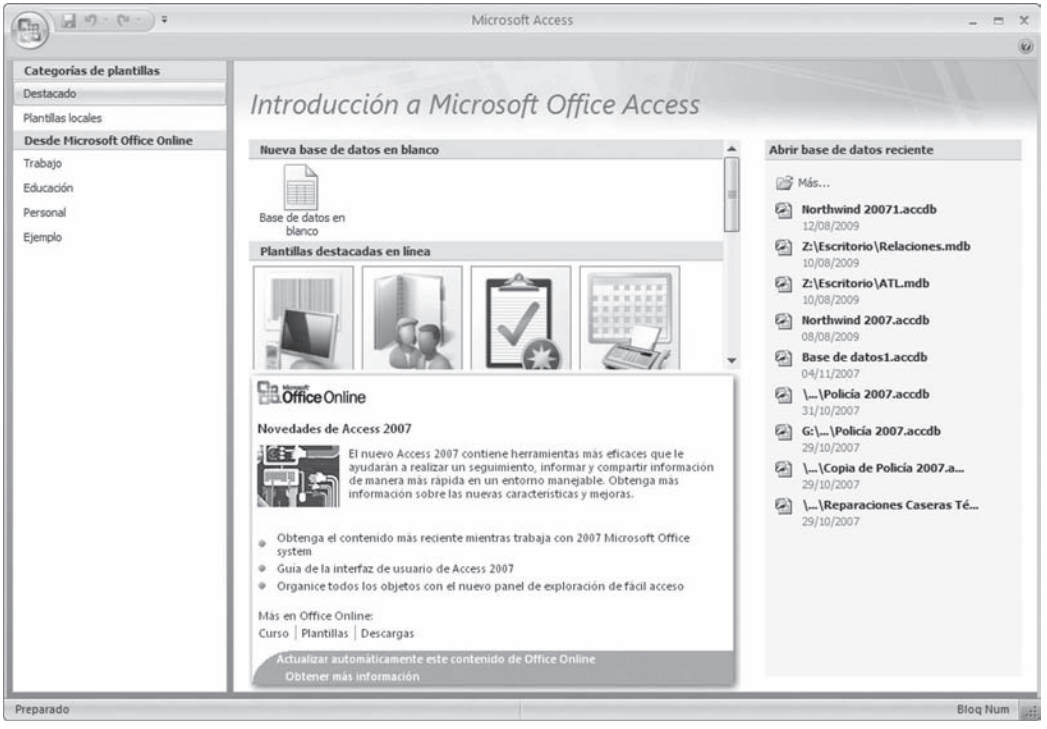


Figura 3-1 El panel de inicio de Microsoft Access 2007.

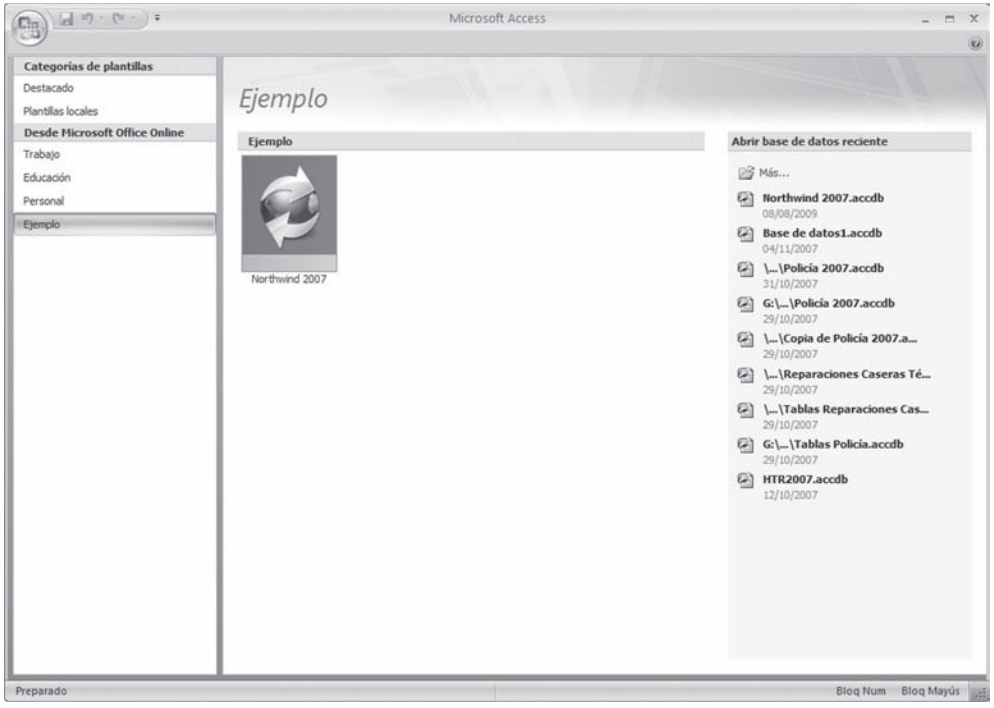


Figura 3-2 El panel de la base de datos de ejemplo de Microsoft Access 2007.

Basic) que no puede ejecutarse hasta que responda a la Advertencia de seguridad que se exhibe en el panel. Puede seguir las instrucciones en la pantalla para habilitar el contenido, si lo prefiere, pero no utilizará el contenido de la aplicación en este capítulo, de modo que puede simplemente cerrar el mensaje al hacer clic en el botón Cerrar (la X) en el extremo derecho del mensaje Advertencia de seguridad. (*No* haga clic en la X de la esquina superior derecha de su pantalla; eso cerraría Microsoft Access y tendría que volver a empezar.) También puede cerrar la pantalla de inicio de Northwind Traders. Para ello, haga clic en el botón Cerrar que se encuentra a la derecha de la ficha Pantalla de inicio, o haga clic con el botón secundario del ratón en la ficha y seleccione Cerrar. Una vez que organice el panel, se verán con mayor claridad las opciones que ofrece.

NOTA

Al igual que casi todas las herramientas de base de datos para equipos de cómputo personales, Access no sólo ofrece una base de datos, sino un entorno de programación completo que permite la creación de pantallas, informes y lógica de aplicación en forma de macros. El desarrollo de una aplicación mediante Access está más allá del alcance de este libro. Este capítulo se concentra en los componentes que se relacionan directamente con la definición de las estructuras de datos y la administración de los datos guardados en ellas.

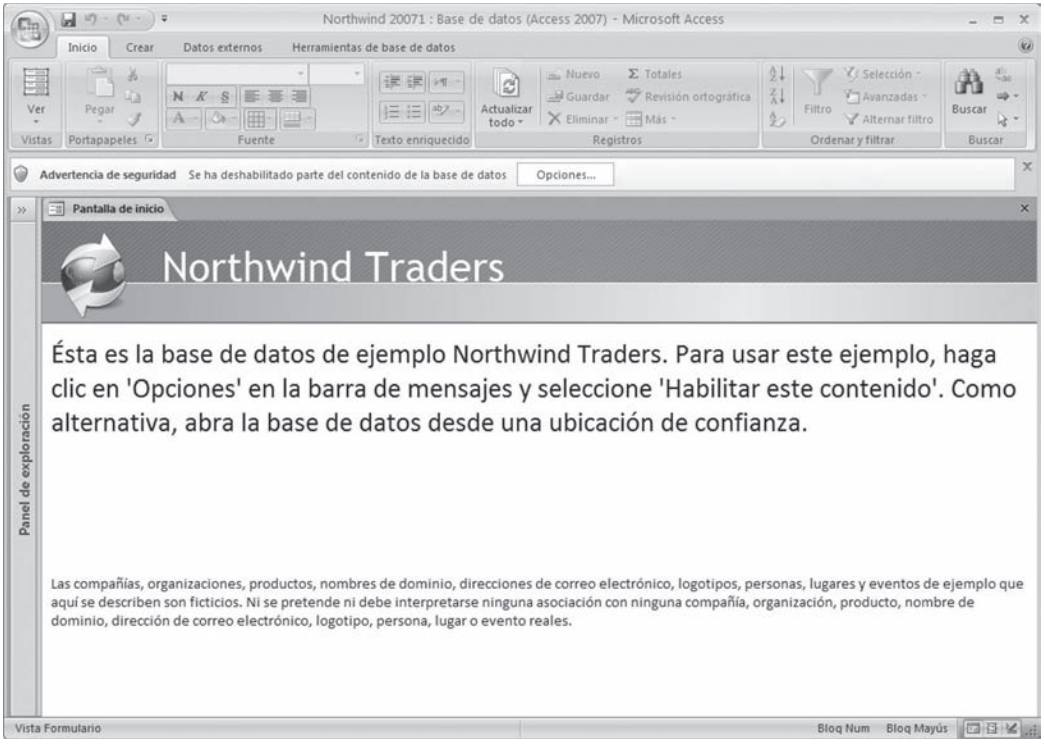


Figura 3-3 El panel principal de Microsoft Access 2007, cinta de opciones Inicio.

El área que corre a lo largo de la parte superior del panel que contiene todas las opciones que puede usar en Access es la *cinta de opciones*. Esta interfaz de usuario es nueva en Office 2007 (y Access es parte de la suite de aplicaciones Office) y supone un alejamiento radical de versiones anteriores, que empleaban una serie de menús desplegables. Si está acostumbrado a emplear la interfaz antigua, requerirá cierto tiempo para adaptarse a la nueva. El botón de Office, en la esquina superior izquierda, proporciona las opciones comunes para todas las aplicaciones de Microsoft Office, como abrir y guardar archivos. Haga clic en él para obtener un menú desplegable de opciones. En la parte superior de la cinta de opciones (a la derecha del botón de Office) está la barra de herramientas Acceso rápido, con opciones para Guardar, Deshacer, Repetir escritura, Imprimir, Vista previa de impresión y Abrir carpeta. Una opción final le permite personalizar la barra de herramientas. Los íconos son razonablemente intuitivos, pero puede colocar el puntero del cursor sobre cada uno de ellos un par de segundos para ver los nombres de las opciones. Éstas son comunes para todas las aplicaciones de Microsoft Office 2007 y, como lo sugiere su nombre, ofrecen una manera rápida de llegar a las opciones que se pueden consultar a través del botón de Office.

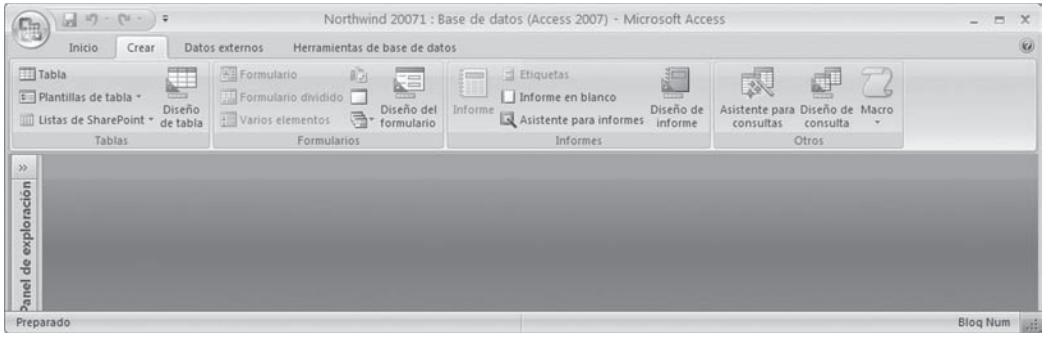


Figura 3-4 El panel principal de Access, cinta de opciones Crear.

Directamente bajo la barra de herramientas Acceso rápido hay fichas para los principales grupos de opciones de cintas disponibles dentro de Access. En las versiones anteriores, se utilizaban para abrir menús desplegables; en Office 2007, son fichas que cambian la cinta de opciones que aparece inmediatamente debajo. Por ejemplo, en la figura 3-3 se muestra la cinta de opciones Inicio. Muchas de las opciones de esta cinta se relacionan con la creación de los componentes de una aplicación dentro de Access (formularios, informes, etc.), que están más allá del ámbito de trabajo de una base de datos. No obstante, empleará a menudo la opción Vista, porque le permite cambiar entre la Vista Diseño, que muestra los metadatos que definen un objeto de la base de datos, y la Vista Hoja de datos, que muestra los datos guardados en el objeto de base de datos en filas y columnas.

La cinta de opciones Crear, presentada en la figura 3-4, aporta opciones para crear tablas, formularios, informes y otros tipos de objetos. No utilizaremos formularios o informes, porque son funciones de programación de aplicaciones más que de base de datos. Como puede ver, el grupo de opciones Tablas le permite crear tablas relacionales mediante diversas herramientas. El grupo Otros, en el lado derecho de la cinta, contiene opciones para las consultas. Estas opciones le permiten crear, ejecutar y guardar consultas relacionadas con la base de datos, que se parece mucho a lo que casi todos los otros DBMS y la norma ISO/ANSI de SQL llaman *vistas*.

En la figura 3-5 se muestra la cinta de opciones Datos externos, que contiene opciones para importar de fuentes externas y exportar a éstas, como casi todas las otras aplicaciones de Office. Aunque en la práctica estas opciones resultarán muy útiles, en este repaso de las funciones no las necesita porque emplea una base de datos de ejemplo que ya contiene información.

La cinta de opciones Herramientas de base de datos, presentada en la figura 3-6, contiene diversas herramientas que ayudan a administrar la base de datos. La más importante de ellas, en cuanto al diseño de una base de datos, es la opción Relaciones, que estudiará en la sección siguiente, aunque primero es necesario que cubra otra importante función de desplazamiento en Access.

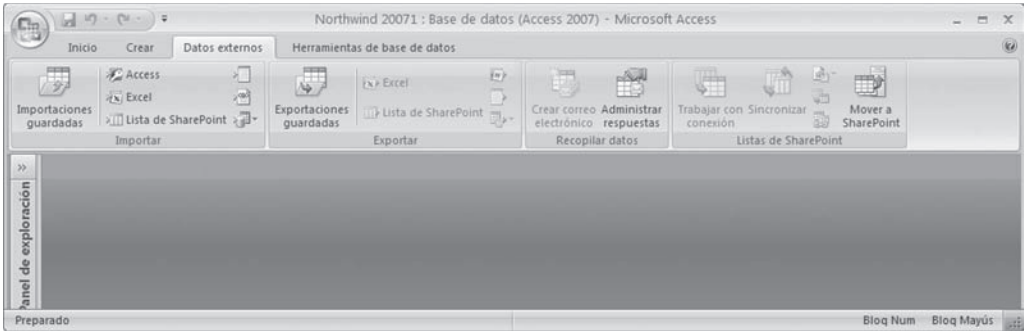


Figura 3-5 El panel principal de Access, cinta de opciones Datos externos.

Es probable que haya observado el Panel de exploración, a lo largo del lado izquierdo de los paneles que hemos examinado hasta este momento. Se trata de una función esencial de Access, porque proporciona un método común para organizar, mostrar y abrir (consultar) los objetos guardados en la base de datos. Cuando lo expande al hacer clic en las dos puntas de flecha (que apuntan a la derecha), verá un panel similar al mostrado en la figura 3-7.

La organización predeterminada del Panel de exploración clasifica los objetos por áreas dentro de la aplicación Northwind Traders, lo que no es nada útil para el trabajo con la base de datos. Si hace clic con el botón derecho del ratón en la parte superior del panel (donde aparece el nombre Northwind Traders) y luego hace clic en Categoría y después en Tipo de objeto, el Panel de exploración se organizará por tipo de objeto en la base de datos, como en la figura 3-8. Puede ampliar cualquier categoría que necesite para ver la lista de objetos en ella y, por supuesto, para minimizar las categorías que no le interesan.

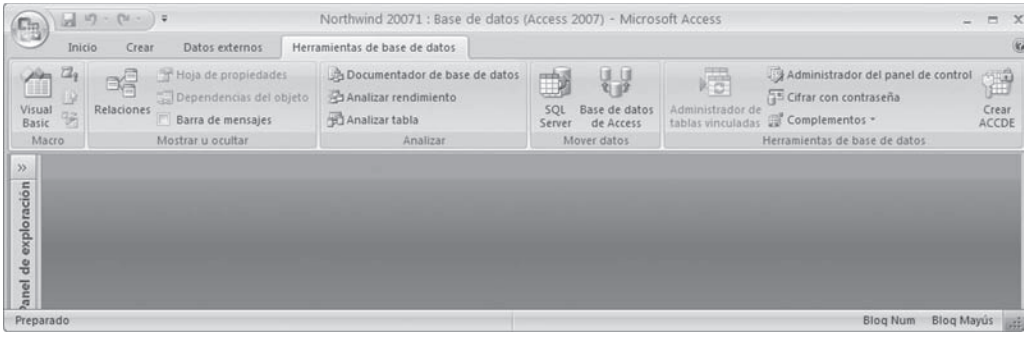


Figura 3-6 El panel principal de Access, cinta de opciones Herramientas de base de datos.

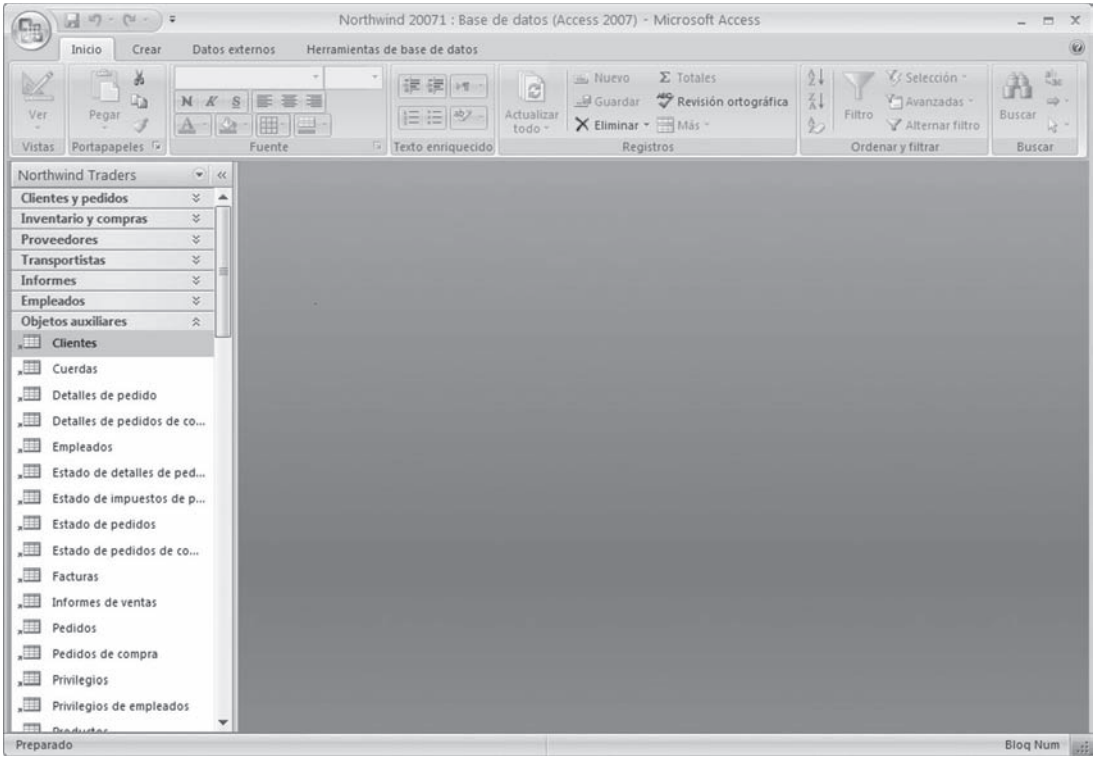


Figura 3-7 El panel principal de Access con el Panel de exploración expandido.

Si ha utilizado versiones anteriores de Access, la lista de tipos de objetos presentada en la figura 3-8 debe resultarle familiar, porque aparecía en el panel principal de las versiones anteriores. Éste es un resumen de los tipos que se pueden definir:

- **Tablas** Tablas relacionales. Contienen los datos reales de la base de datos en filas y columnas.
- **Consultas** Consultas guardadas en la base de datos. Se llaman *vistas* en casi todas las otras bases de datos relacionales.
- **Formularios** Formularios GUI para introducir y mostrar datos dentro de Microsoft Access.
- **Informes** Informes basados en consultas a la base de datos.

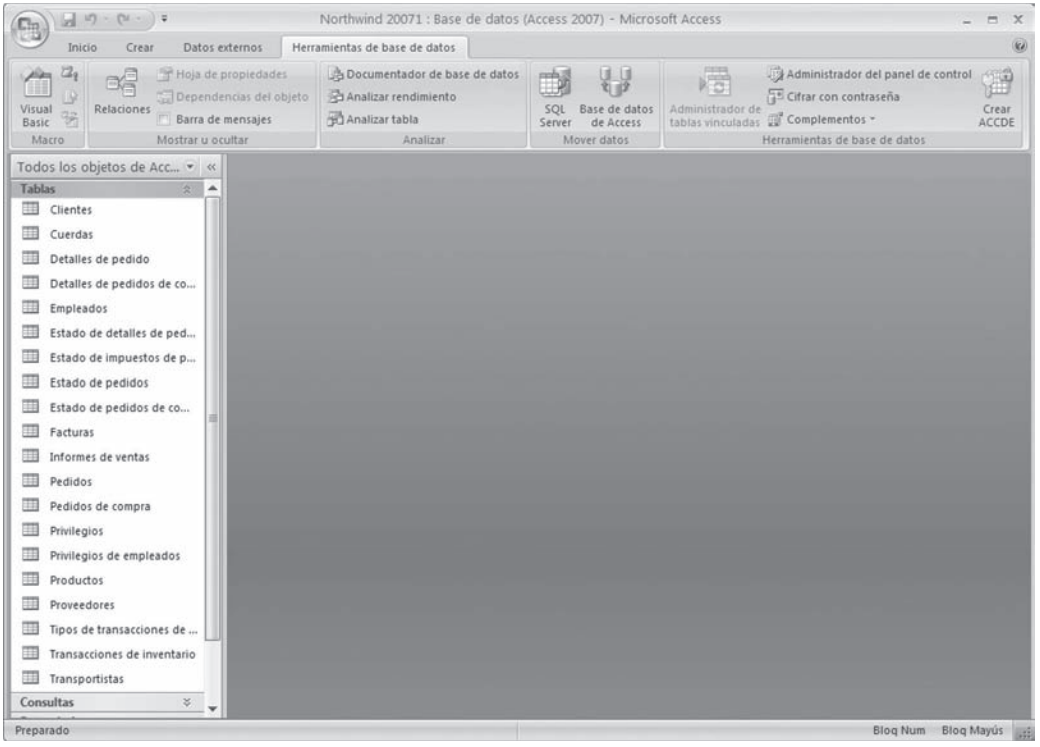


Figura 3-8 El Panel de exploración organizado por tipo de objetos.

- **Macros** Grupos de acciones en que cada una efectúa una operación específica, como abrir un formulario o imprimir un informe.
- **Módulos** Conjuntos de componentes del lenguaje de programación Visual Basic que se guardan como una unidad.

Como se señaló antes, Microsoft Access no es sólo una base de datos, sino un entorno de desarrollo completo para crear y ejecutar aplicaciones. Los productos de base de datos en el nivel empresarial que suelen funcionar en sistemas de cómputo más grandes y compartidos, llamados *servidores*, *no* suelen contener ambientes de desarrollo de aplicaciones. El hecho de aprender a crear programas de aplicaciones está mucho más allá del alcance de este libro, de modo que no manejaremos en absoluto los tipos Formularios, Informes, Macros y Módulos. Sólo nos concentraremos en los tipos Tablas y Consultas de Microsoft Access.

El mantenimiento de los objetos de la base de datos se efectúa desde este panel, que incluye las tareas siguientes:

- Para agregar un objeto nuevo, utilice la cinta de opciones Crear, y haga clic en el ícono adecuado. Por ejemplo, puede crear una tabla nueva al hacer clic en el ícono Tabla o Diseño de tabla, en la cinta de opciones Crear.
- Para eliminar un objeto existente, haga clic con el botón derecho del ratón en su nombre en el Panel de exploración, y elija la opción Eliminar.
- Para abrir un objeto, haga doble clic en su nombre en el Panel de exploración.
- Para mostrar la definición (el diseño) de un objeto, haga clic en su nombre con el botón derecho del ratón en el Panel de exploración y seleccione la opción Vista Diseño.

El panel Relaciones de Microsoft Access

Microsoft Access ofrece un panel Relaciones, expuesto en la figura 3-9, para la definición y el mantenimiento de las restricciones referenciales entre las tablas relacionales. Para mostrar este panel, haga clic en la opción Relaciones, en la cinta de opciones Herramientas de base de datos.

NOTA

Si sigue los pasos con su propia copia de la base de datos Northwind, el panel mostrará muchas más tablas y relaciones. Aquí se simplificó la imagen para que el lector comprendiera mejor la figura 3-9. Es posible que también observe la columna Id de gerente, de la tabla Empleados de la figura, que agregué para ilustrar una relación recursiva, lo que se cubre más adelante en el capítulo.

El panel Relaciones muestra gráficamente las tablas, presentadas como rectángulos, y las relaciones uno a varios, mostradas como líneas entre los rectángulos. Técnicamente, éstas son restricciones referenciales (las *relaciones* son sólo un término conceptual), pero debido a que Microsoft las llama relaciones en este panel, también se emplea aquí el término por uniformidad. El símbolo *1* indica el lado "uno" de cada relación, mientras que el símbolo de infinito (similar a un número 8 de costado) señala el lado "varios" de cada relación. También puede observar una punta de flecha en el extremo de algunas líneas, lo que denota las relaciones que tienen definida una búsqueda (tal como se analizó en el capítulo 2).

Las relaciones se modifican del modo siguiente:

- Para agregar tablas que no están mostradas, haga clic en el ícono Mostrar tabla (la tabla y un signo de más en amarillo) en la cinta de opciones, y seleccione las tablas del menú emergente.

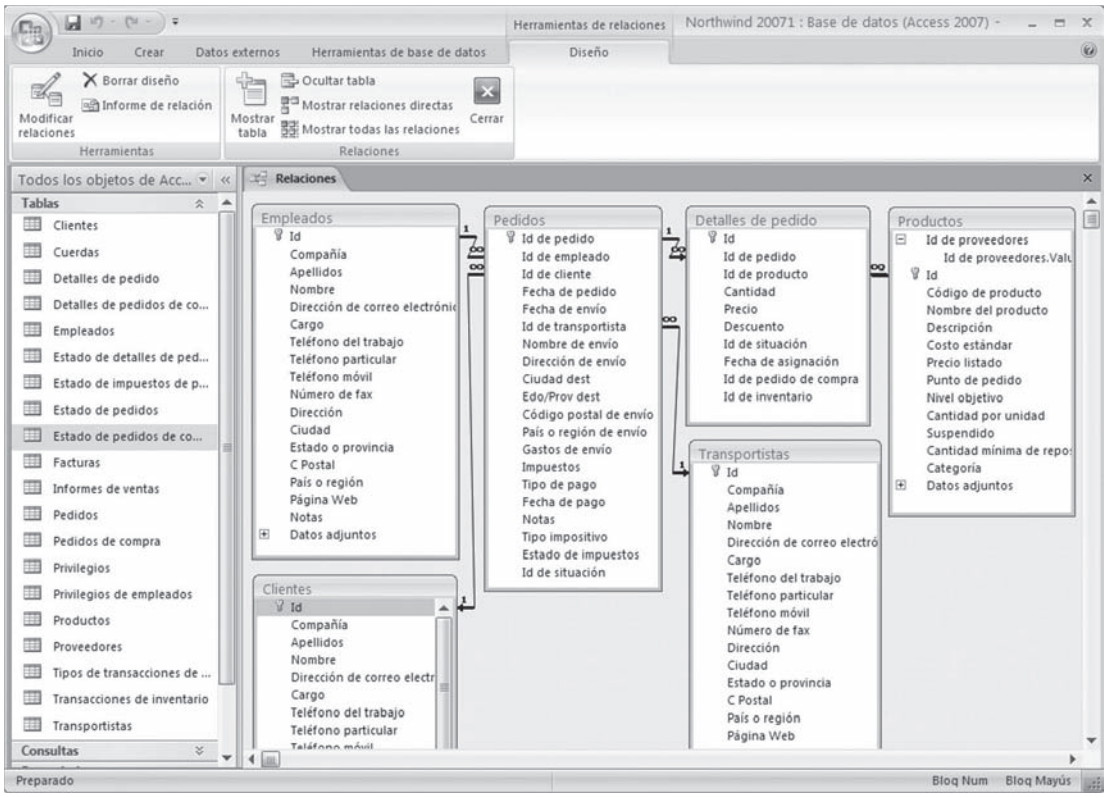


Figura 3-9 El panel Relaciones de Microsoft Access.

- Para evitar que se muestre una tabla, haga clic en ella para seleccionarla y luego oprima SUPR. Observe que esto *no* elimina la tabla o ninguna relación en que participa la misma; simplemente quita la tabla del panel.
- Para agregar una relación, arrastre la clave principal de una tabla hacia una clave externa coincidente en otra. En el caso de relaciones recursivas, la tabla debe agregarse para que se despliegue por segunda vez, y debe crearse la relación entre la copia mostrada de la tabla y la original. Al principio, esto parece extraño, pero sirve para facilitar el método de arrastrar y colocar para crear la relación. Una tabla presentada varias veces en el panel sólo existe una vez en la base de datos.
- Para eliminar una relación, haga clic en la parte estrecha (la sección media) de su línea y oprima SUPR. La selección de las relaciones es engañosa en Microsoft Access, porque sólo funciona cuando se hace clic en la parte estrecha de la línea y tal vez tenga que estirar las

líneas cortas desplazando una tabla en el panel con el fin de exponer la parte estrecha de la línea.

- Para modificar una relación, haga doble clic en la parte estrecha de su línea. Se utiliza una ventana emergente para modificar las diferentes opciones sobre la relación, entre ellas activar y desactivar la imposición de la relación como una restricción referencial (es decir, habilitar y deshabilitar la restricción). Cuando se deshabilita una restricción, el DBMS no evitará inserciones, actualizaciones ni eliminaciones a partir de la creación de valores de claves externas "huérfanas" (valores de claves externas que no tienen valores de clave principal coincidente en la tabla padre). Sin embargo, el DBMS no permitirá que se habilite una restricción si existen valores de claves externas huérfanas en la tabla hija.

Para cerrar el panel Relaciones, puede hacer clic en el botón Cerrar (X), que se encuentra en la esquina superior derecha del panel, o hacer clic con el botón derecho del ratón en la ficha Relaciones y elegir Cerrar.

La vista Diseño de tabla de Microsoft Access

Una tabla se selecciona al hacer doble clic en su nombre, en el Panel de exploración. La pantalla predeterminada, llamada vista Hoja de datos, se presenta en la figura 3-10. Los datos de la tabla se muestran en la conocida forma tabular, y se pueden actualizar, si es necesario, incluida la inserción y la eliminación de filas. Tenga cuidado, porque no hay una función Deshacer; una vez que hace avanzar el curso de una fila a la siguiente, no se podrá revertir con facilidad cualesquier cambio que haya hecho.

Puede llegar a la vista Diseño, que presenta la definición de la tabla, de dos maneras. Puede hacer clic con el botón derecho en la ficha con el nombre de la tabla y seleccionar vista Diseño, o puede elegir la cinta de opciones Inicio (si todavía no está seleccionada), hacer clic en el ícono Vista y seleccionar la opción Diseño. En la figura 3-11 se presenta la vista Diseño para la tabla Empleados.

La vista Diseño de una tabla presenta información como la siguiente:

- **Nombre del campo** El nombre de la columna.
- **Tipo de datos** El tipo de datos para la columna.
- **Descripción** Una descripción de la columna, por lo general proporcionada por un DBA.
- **Tamaño del campo** Un tipo secundario dentro del tipo de datos. Por ejemplo, se aplican Long Integer (entero largo) y Short Integer (entero corto) al tipo de datos más general, Number (número).

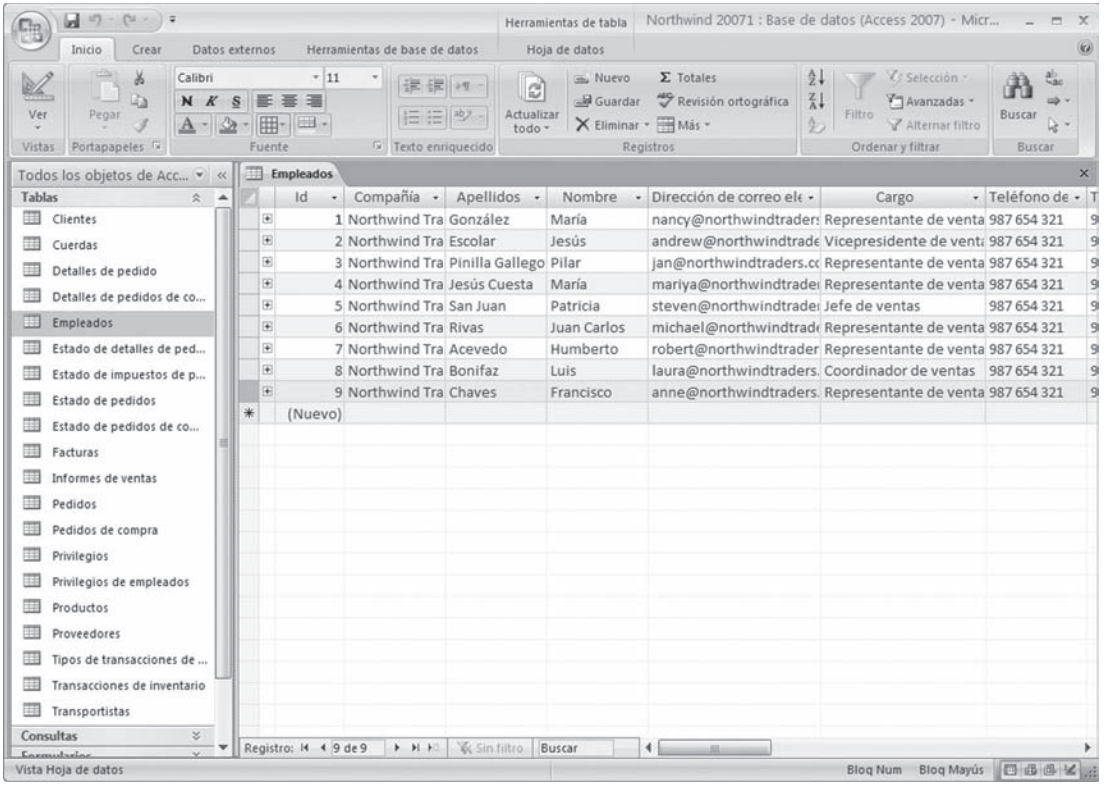


Figura 3-10 Vista Hoja de datos (tabla Empleados).

- **Requerido** Indica si la columna es opcional (es decir, si puede tener valores nulos).
- **Índice** Indica si la columna posee un índice.
- **Clave principal** Señalada con un ícono pequeño de llave junto al nombre del campo o los campos que integran la clave principal.

Confiamos en que haya reconocido que todo el contenido de este panel son *metadatos*. Están disponibles muchas opciones adicionales, pero no se señalan aquí, y Microsoft Access oculta y expone con gran inteligencia las opciones para que sólo se muestren las aplicables. Observe que se presenta automáticamente texto de ayuda en el área azul de la esquina inferior derecha del panel, mientras mueve el curso de una opción a otra.

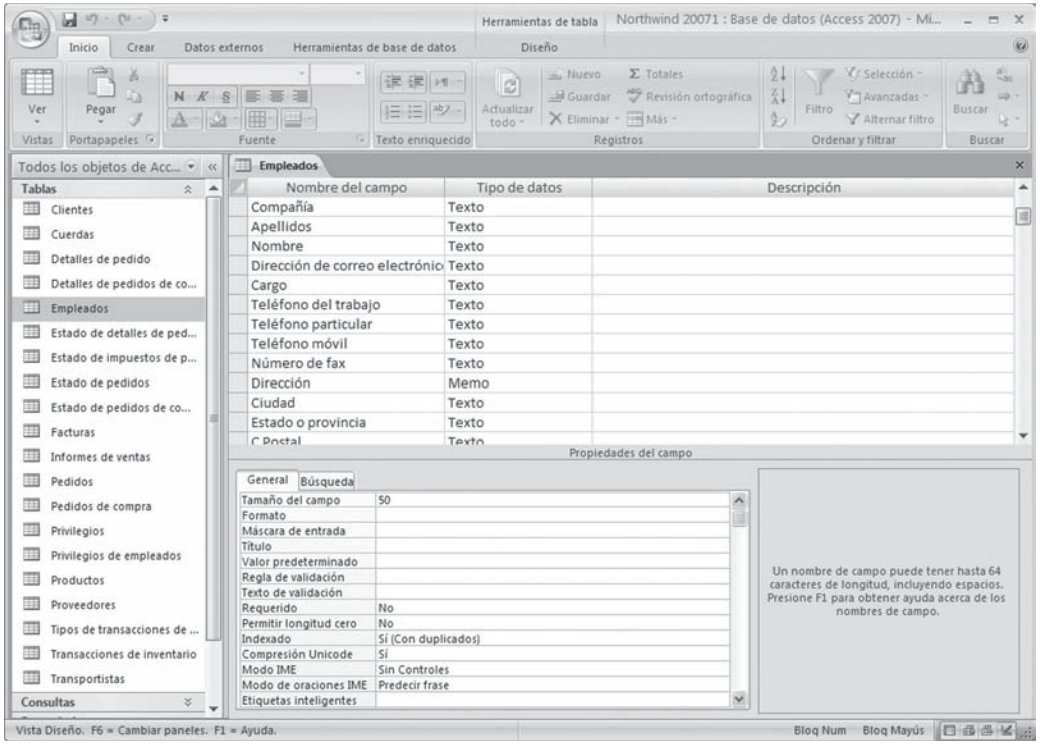


Figura 3-11 Vista Diseño (tabla Empleados).

Creación de consultas en Microsoft Access

Como ya se mencionó, las consultas de Microsoft Access se parecen mucho a lo que casi todos los DBMS llaman *vistas*, porque una vista se define en la norma SQL como una consulta guardada de la base de datos. Una semejanza importante es que las consultas de Access, igual que las vistas, no guardan datos; en cambio, los datos en guardan en las tablas. Sin embargo, las consultas de Access tienen ciertas capacidades que no se encuentran en las vistas, como la posibilidad de ajustar una consulta para efectuar inserciones o actualizaciones de las filas de datos en la base de datos. En el Panel de exploración, la ampliación de la categoría Consultas presenta una lista de todas las consultas guardadas en esta base de datos, como se observa en la figura 3-12.

Aunque Microsoft Access ofrece varias maneras de crear una consulta nueva, los principiantes comprenden con más facilidad la opción Diseño de consulta. Cuando se hace clic en el ícono Diseño de consulta (en el área Otros, de la cinta de opciones Crear), Access despliega el cuadro de diálogo Mostrar tabla, que se muestra en la figura 3-13.

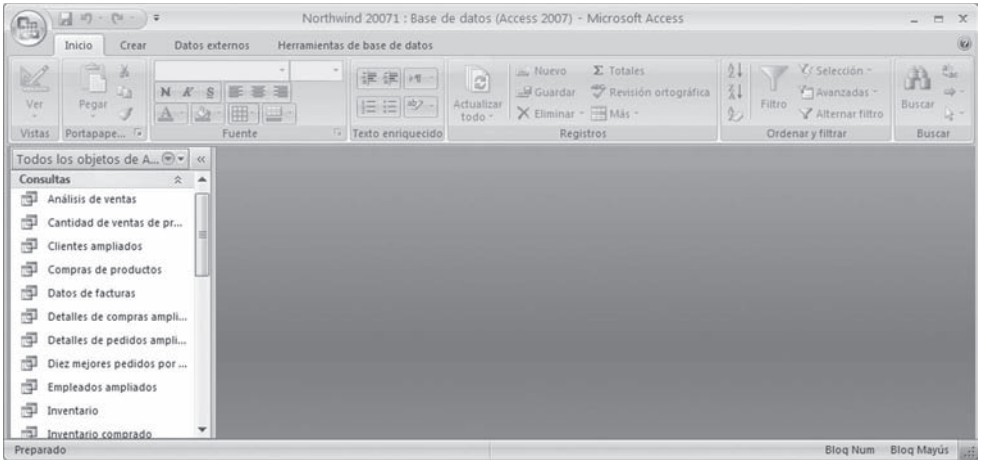


Figura 3-12 Lista de las consultas de la base de datos de Northwind.

Para cada consulta nueva, Access abre el cuadro de diálogo Mostrar tabla con el fin de permitirle que seleccione las tablas, las consultas, o ambas, en que se basará la nueva consulta (es decir, las tablas o consultas que van a ser el origen de los datos que se mostrarán). Conforme se agregan tablas y consultas, aparecen en el panel Diseño de consulta, que permite introducir la especificación para la consulta deseada. En la figura 3-14 se presenta el panel Diseño de consulta, con la tabla Clientes agregada.

El panel Diseño de consulta tiene los componentes siguientes:

- En el área libre que se encuentra en la parte superior del panel (con el fondo azul claro) aparece una representación gráfica de las tablas y las consultas que son el origen de la consulta, además de sus relaciones. Las relaciones definidas para las tablas se heredan automáticamente aquí.

Figura 3-13 Cuadro de diálogo Mostrar tabla.

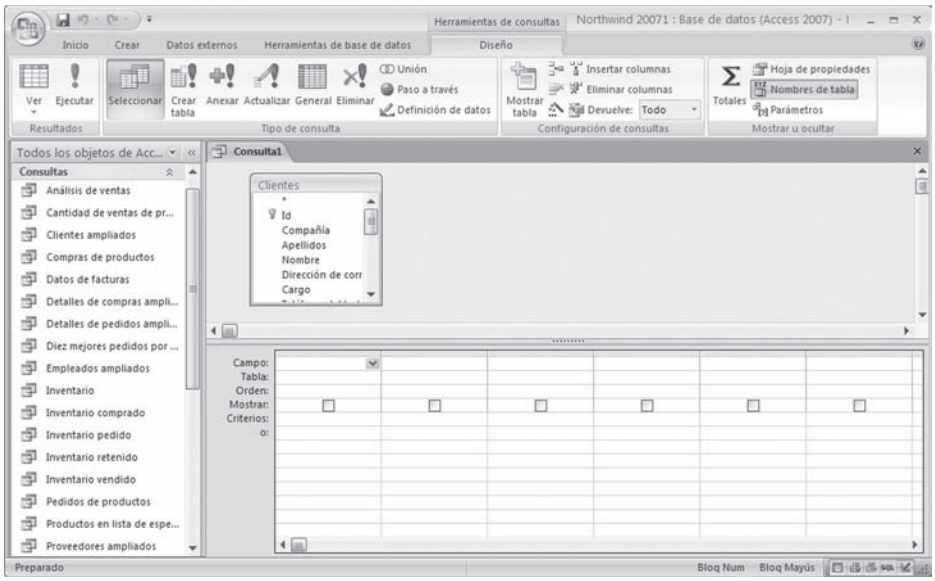


Figura 3-14 Panel Diseño de consulta (con la tabla Clientes agregada).

- En el área de cuadrícula que se encuentra en la parte inferior del panel, cada columna representa una columna de datos que se va a devolver en el conjunto de resultados cuando se ejecute la consulta. Las filas del área de la cuadrícula definen las diversas opciones que se van a aplicar a las columnas correspondientes. Las secciones siguientes proporcionan ejemplos de su utilización:

  - **Campo** La especificación del origen de la columna. Suele ser el nombre de una tabla o una columna de consulta, pero también puede ser una constante o una expresión similar a los cálculos utilizados en las hojas de cálculo.
  - **Tabla** El nombre de la tabla o la consulta de origen para la columna.
  - **Orden** La especificación de cualquier secuencia de orden para la columna (ascendente, descendente o ninguno).
  - **Mostrar** Una casilla de verificación que controla el despliegue de la columna. Si no se selecciona la casilla, la columna se emplea para formar la consulta, pero no aparece en sus resultados.
  - **Criterios** La especificación que determina cuáles filas de datos van a aparecer en los resultados de la consulta. Deben cumplirse todas las condiciones aplicadas a la misma línea para que se muestre una fila de datos en los resultados de la consulta. Las condiciones aplicadas a las líneas siguientes (denominadas “o” en el panel) son conjuntos

de condiciones alternas que harán que también se despliegue en los resultados una fila de datos coincidentes. No es probable que la utilización de los criterios cobre sentido hasta que vean los ejemplos siguientes pero, en resumen, las condiciones aplicadas a la misma línea se conectan con un operador **Y** lógico, y cada nueva línea de criterios se conecta con un operador **O** lógico con todas las otras líneas. Dicho de otro modo, cualquier fila que cumpla las especificaciones que aparecen en cualquiera de las líneas de criterios se incluirá en los resultados de la consulta.

La opción Criterios es la más compleja y por eso requiere una explicación más detallada. Las condiciones se suelen escribir mediante un operador de comparación y uno o más valores de datos. No obstante, puede omitirse el operador igual a (=). Por ejemplo, si sólo quiere seleccionar las filas en que el valor de una columna es igual a 0, puede escribir **=0** o simplemente **0**. Los valores de los caracteres se encierran entre apóstrofos o comillas, pero si omite eso, Access supone que están basados en el tipo de datos de la columna. Por ejemplo, si sólo quiere seleccionar las filas que contienen un valor de columna **M**, puede introducir la condición en cualquiera de los modos siguientes: **M**, **'M'**, **"M"**, **=M, ='M'** o **="M"**. Cuando escriba fechas, verá que Access delimita estos valores mediante el signo de número (#), por lo que no necesita preocuparse por hacerlo usted mismo. Como es de suponer, puede emplear otras operaciones de comparación además del signo de igual (=). La tabla siguiente presenta todos los operadores de comparación permitidos:

Operador	Descripción
=	Igual a
<	Menor que
<=	Menor que o igual a
>	Mayor que
>=	Mayor que o igual a
<>	No es igual a

Una vez completa la especificación, al hacer clic en el ícono Ejecutar (el signo de admiración) se ejecuta la consulta y se muestran los resultados mediante la vista Hoja de datos, igual que la presentada en la figura 3-10. Para regresar al panel Diseño de consulta, simplemente haga clic en el ícono vista Diseño (el ícono de regla, lápiz y escuadra en el grupo Vistas, de la cinta de opciones Inicio). En casi todas las consultas, las actualizaciones de datos se introducen directamente en la vista Hoja de datos de la tabla, y se aplican directamente a las tablas de origen para la consulta. Si una columna de los resultados de la consulta no puede ubicarse sobre una columna de tabla única (tal vez porque fue calculada de alguna manera), no puede actualizarse en los resultados de la consulta.

Si todo esto parece confuso, es porque el mejor modo de aprender a crear consultas en Microsoft Access es mediante la prueba. Por lo tanto, en el resto de este capítulo se utiliza una

serie de ejercicios Pruebe esto, para mostrar las poderosas funciones de la herramienta Consultas de Microsoft Access. Para reducir la cantidad de trabajo requerido para concluir cada uno, los ejercicios se suceden uno tras otro. Cada ejercicio ofrece una descripción del resultado deseado y los pasos requeridos con el fin de crear la especificación para la consulta en el panel Diseño de consulta. Después aparece una figura que contiene dos imágenes. La primera presenta el panel Diseño de consulta completado, y la otra muestra los resultados después de ejecutar la consulta.

Pruebe esto 3-1 Lista de todos los clientes

En este ejercicio simplemente presentará la tabla Clientes completa (todas las filas y todas las columnas).

Paso a paso

1. En la cinta de opciones Crear, haga clic en Diseño de consulta.
2. Efectúe las acciones siguientes en el cuadro de diálogo Mostrar tabla:
   a. Haga clic en Clientes para seleccionar la tabla Clientes.
   b. Haga clic en el botón Agregar.
   c. Haga clic en el botón Cerrar.
3. En el panel Diseño de consulta, haga doble clic en el asterisco de la plantilla de la tabla Clientes (cerca de la parte superior del panel).
4. Haga clic en el ícono Ejecutar de la cinta de opciones (el signo de admiración) para aplicar la consulta. El panel completado se presenta en la parte superior de la figura 3-15 y los resultados se muestran en la parte inferior.
5. Con el fin de prepararse para el siguiente ejercicio, haga lo siguiente en el panel de resultados de la consulta (parte inferior de la figura 3-15):
   a. Regrese al panel Diseño de consulta, al hacer clic en el ícono Ver (la escuadra, la regla y el lápiz) justo debajo del botón de Office.
   b. En el panel Diseño de consulta (parte superior de la figura 3-15), borre la especificación existente de la consulta al hacer clic en la franja gris muy delgada, justo arriba del nombre de campo Clientes* (lo que cambia la columna completa a un fondo negro). Luego oprima Supr para eliminar la columna.

(*continúa*)

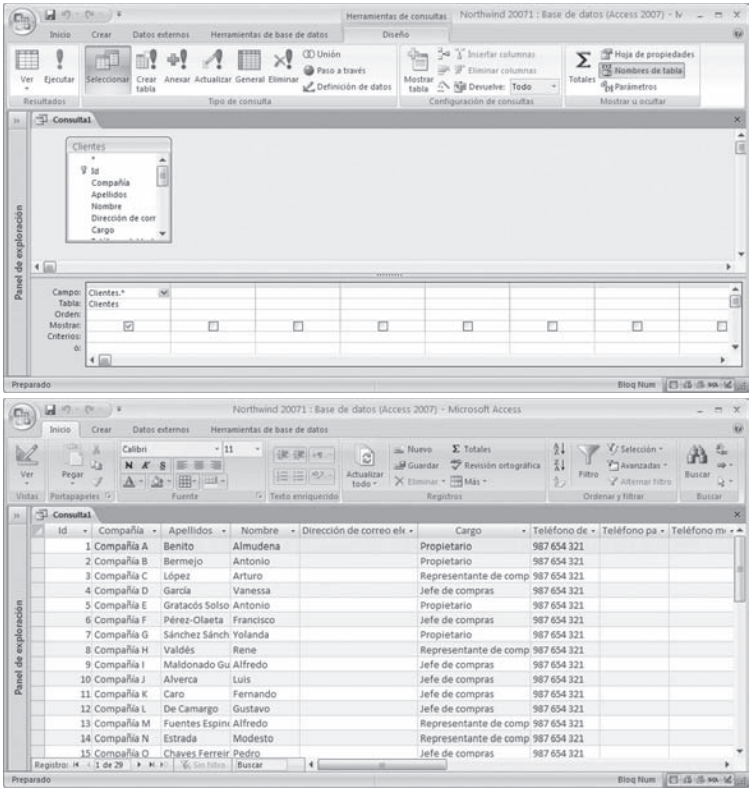


Figura 3-15 Pruebe esto 3-1 (Lista de todos los clientes), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-2 Elección de las columnas que se muestran

En lugar de mostrar todas las columnas, aquí especificará sólo las que quiere ver. Presentará las columnas Id, Compañía, Ciudad, Estado o provincia, y País o región para todos los clientes (todas las filas de la tabla Clientes).

Paso a paso

1. Ya debe tener abierto el panel Diseño de consulta con la tabla Clientes agregada a la consulta.

2. Para cada columna que quiera ver (Id, Compañía, Ciudad, Estado o provincia y País o región), haga doble clic en el nombre de la columna, en la tabla de la parte superior del formulario. Un método alterno consiste en arrastrar y colocar el nombre de la columna de la tabla mostrada en la parte superior del formulario a la cuadrícula de la parte inferior del formulario.
3. Haga clic en el ícono Ejecutar de la cinta de opciones para ejecutar su consulta. El panel completado se presenta en la parte superior de la figura 3-16, y los resultados de la consulta en la parte inferior.
4. Con el fin de prepararse para el siguiente ejercicio, regrese al panel Diseño de consulta, al hacer clic en el ícono Ver (la escuadra, la regla y el lápiz), bajo el botón de Office.

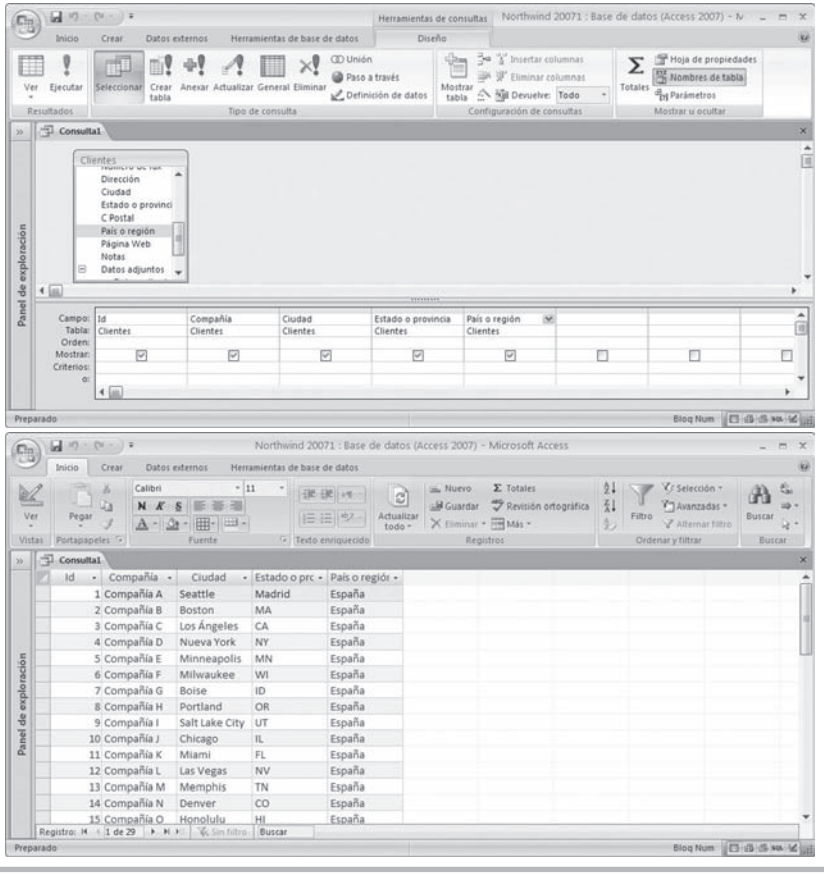


Figura 3-16 Pruebe esto 3-2 (Elección de las columnas que se muestran), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-3 Ordenamiento de los resultados

En cualquier RDBMS, las filas son devueltas sin ningún orden específico, a menos que solicite lo contrario. Microsoft Access emplea la especificación Orden para determinar el orden en que se devuelven en los resultados de la consulta. Modificará lo obtenido en Pruebe esto 3-2 para que las filas se clasifiquen en un orden ascendente por Ciudad, Estado o provincia, y País o región.

Paso a paso

1. Ya debe tener abierto el panel Diseño de consulta, con la consulta que generó en Pruebe esto 3-2 desplegada.
2. En la fila Orden de la columna Ciudad, haga clic en el espacio en blanco y seleccione Ascendente, de la lista desplegable (revise la figura 3-17).
3. Haga lo mismo para la columna Estado o provincia. Un método alterno sencillo es escribir **A** (de ascendente) en la especificación de Orden y oprimir ENTER.
4. Haga lo mismo para la columna País o región.
5. Haga clic en el ícono Ejecutar, de la cinta de opciones, para activar la consulta. El panel completado se presenta en la parte superior de la figura 3-17 y los resultados de la consulta se muestran en la parte inferior.
6. Con el fin de prepararse para el ejercicio siguiente, regrese al panel Diseño de consulta, al hacer clic en el ícono Ver (la escuadra, la regla y el lápiz) justo debajo del botón de Office.

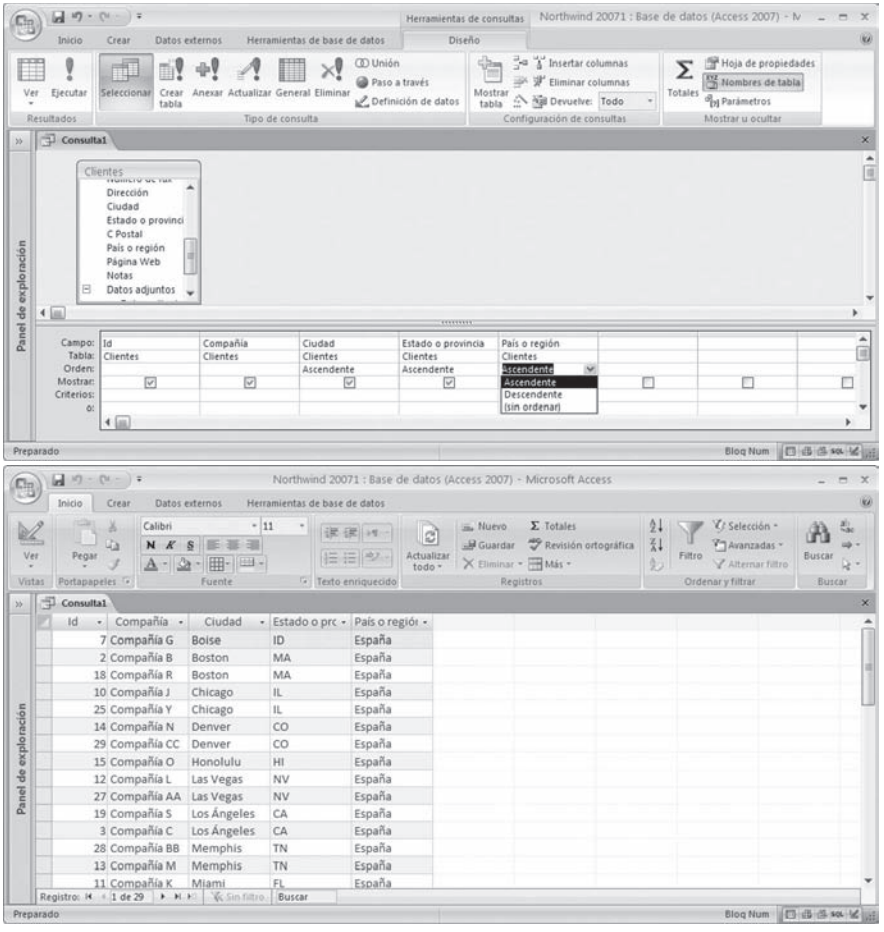


Figura 3-17 Pruebe esto 3-3 (Ordenamiento de los resultados), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-4 Ordenamiento avanzado

Al observar los resultados de Pruebe esto 3-3, verá que todas las ciudades se presentan en una secuencia ascendente y que la clasificación por Estado o provincia y después por País o región tuvo poco efecto y sólo importaría si existieran dos ciudades con el mismo nombre en diferentes estados o provincias y países o regiones. Como el lenguaje hablado no siempre tiene

(*continúa*)

una lógica precisa, no es probable que esto sea lo que buscaba al indicar que quería los datos clasificados por Ciudad, Estado o provincia y País o región. En cambio, tal vez quisiera que todas las filas de un País o región estuvieran juntas, y que para cada Estado o provincia, todas las ciudades se presentaran en una secuencia ascendente por nombre. Si hubiera señalado ordenar por Ciudad *dentro* de Estado o provincia *dentro* de País o región, la intención hubiera sido más clara. Ahora necesita un modo de ordenar primero por País o región, después por Estado o provincia y al final por Ciudad, pero Ciudad se muestra antes que Estado o provincia, y esta última antes que País o región. La clasificación de Microsoft Access funciona en las columnas de la consulta de izquierda a derecha. ¿Cómo se puede conseguir esto? Es posible poner las columnas Estado o provincia y Ciudad una segunda vez en la consulta, emplear las segundas copias para la clasificación, pero omitirlas en los resultados de la consulta mediante la casilla de verificación Mostrar.

En este ejercicio Pruebe esto, modificará lo obtenido en la sección Pruebe esto 3-3, para que las filas se ordenen como se analizó.

Paso a paso

1. Ya debe tener abierto el panel Diseño de consulta, con la consulta que obtuvo en Pruebe esto 3-3 expuesta.
2. Elimine las especificaciones de Orden, en la columna Ciudad existente:
   a. Haga clic en la fila Orden, de la especificación de consulta para la columna.
   b. Haga clic en la flecha que apunta hacia abajo para exhibir el menú desplegable.
   c. Seleccione la opción (sin ordenar) de la lista.
3. Haga lo mismo con la columna Estado o provincia.
4. Agregue la columna Estado o provincia a la especificación de la consulta una segunda vez al hacer doble clic en su nombre en la tabla Clientes.
5. Haga lo mismo para la columna Ciudad.
6. Incorpore la especificación de orden ascendente en las columnas Estado o provincia y Ciudad que acaba de agregar (las que están a la *derecha* de la columna País o región).
7. Quite la marca de la casilla de verificación Mostrar para las columnas Estado o provincia y Ciudad que acaba de incluir. Esto evitará que los datos en ellas se muestren por segunda vez en los resultados de su consulta.
8. Como este ejercicio es un poco complicado, compare su panel Diseño de consulta con el mostrar en la figura 3-18 para comprobar que hizo todo correctamente.

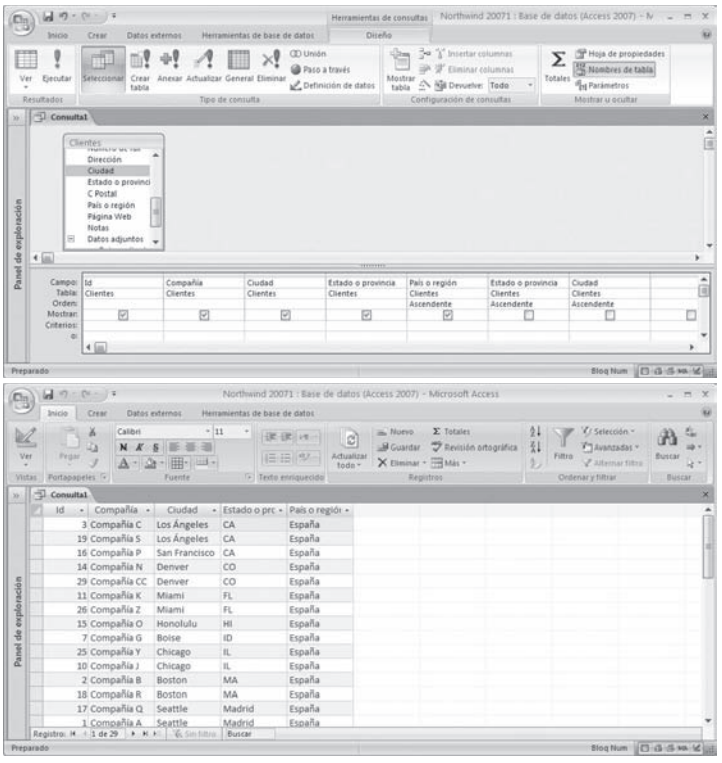


Figura 3-18 Pruebe esto 3-4 (Ordenamiento avanzado), diseño de consulta (arriba) y resultados de la consulta (abajo).

9. Haga clic en el ícono Ejecutar de la cinta de opciones para ejecutar su consulta. El panel completado aparece en la parte superior de la figura 3-18 y los resultados de la consulta se muestran debajo. Tome en cuenta que en casi todos los idiomas se lee de izquierda a derecha; por eso es natural esperar que las listas tabulares se ordenen al avanzar de izquierda a derecha, comenzando con la columna del extremo izquierdo. No es frecuente, y tal vez es parte de una ingeniería humana deficiente, clasificar las columnas de otro modo. Pero si alguna vez necesita hacerlo, ya sabe cómo.

10. Con el fin de prepararse para el ejercicio siguiente, haga esto:

- **a.** Regrese al panel Diseño de consulta, al hacer clic en el ícono Ver (la escuadra, la regla y el lápiz) justo bajo el botón de Office.
- **b.** Para simplificar los siguientes ejercicios Pruebe esto, regrese la especificación de la consulta al estado que guardaba al final de la sección Pruebe esto 3-3.

(*continúa*)

c. Elimine las columnas Estado o provincia y País o región que agregó a la especificación Orden, al hacer clic en la delgada franja gris sobre el nombre de campo (que cambia la columna completa a un fondo negro) y luego oprimir SUPR, para quitar cada una.

d. Incorpore la especificación Orden ascendente a las columnas Ciudad y Estado o provincia, al hacer clic en la fila Orden de cada una, escribir la letra **A**, y oprimir ENTER. Con esto indicará *Ascendente* para cada columna.

Pruebe esto 3-5 Elección de las filas que se muestran

Hasta este momento, ha mostrado las 26 filas de la tabla Clientes en cada consulta. Si no quiere ver todas las filas, resulta innecesario mostrarlas, y desperdicia los recursos del sistema, sobre todo si las ordena. Suponga que sólo quiere ver las filas de los clientes en San Francisco, CA. Puede incluir condiciones mediante la línea Criterios, en el panel Diseño de consulta, para filtrar las filas de modo que sólo se incluyan las que prefiera. Recuerde que para que una fila se muestre en los resultados, necesitan evaluarse como verdaderas todas las condiciones de cuando menos una de las líneas de Criterios. En este caso, Northwind tenía clientes en San Francisco y Los Ángeles, de modo que es importante incluir condiciones no sólo para el estado, sino también para la ciudad. (Se puede argumentar que la condición en la columna Estado o provincia es innecesaria porque ningún otro estado tiene una ciudad llamada San Francisco, pero es muy bueno cuando se crean consultas de bases de datos para incluir condiciones adicionales, porque suelen ayudar a que el DBMS procese la consulta con más eficiencia; además, evitan sorpresas innecesarias, en caso de que la consulta se reutilice después para otro propósito, como elegir una ciudad que no tenga un nombre único.)

En este ejercicio, modificará la especificación de la consulta de la sección Pruebe esto 3-3, para filtrar los resultados con el fin de incluir sólo los clientes de San Francisco.

Paso a paso

1. Debe comenzar con una especificación de consulta semejante a la de la figura 3-17.

2. En la fila Criterios, de la columna Ciudad, escriba **San Francisco**. Observe que Microsoft Access no pone atención a las mayúsculas cuando selecciona datos en las consultas, de modo que también puede escribir **SAN FRANCISCO** o **san francisco** y obtener el mismo resultado. Recuerde las constantes de caracteres que se utilizan en un RDBMS suelen encerrarse entre comillas. Sin embargo, Microsoft Access sabe que la columna Ciudad tiene un tipo de datos de caracteres, de modo que agregará las comillas automáticamente, en caso de que lo olvide.

3. En la misma fila, escriba **CA**, en la columna Estado o provincia. Es importante introducir los criterios Ciudad y Estado o provincia en la misma línea, porque quiere que sólo se devuelvan las filas donde Ciudad es San Francisco y Estado o provincia es CA.
4. Haga clic en el ícono Ejecutar, de la cinta de opciones, para activar su consulta. El panel completado está en la parte superior de la figura 3-19, y los resultados de la consulta aparecen debajo.
5. Con el propósito de prepararse para el ejercicio siguiente, regrese al panel Diseño de consulta, al hacer clic en el ícono Ver debajo del botón de Office.

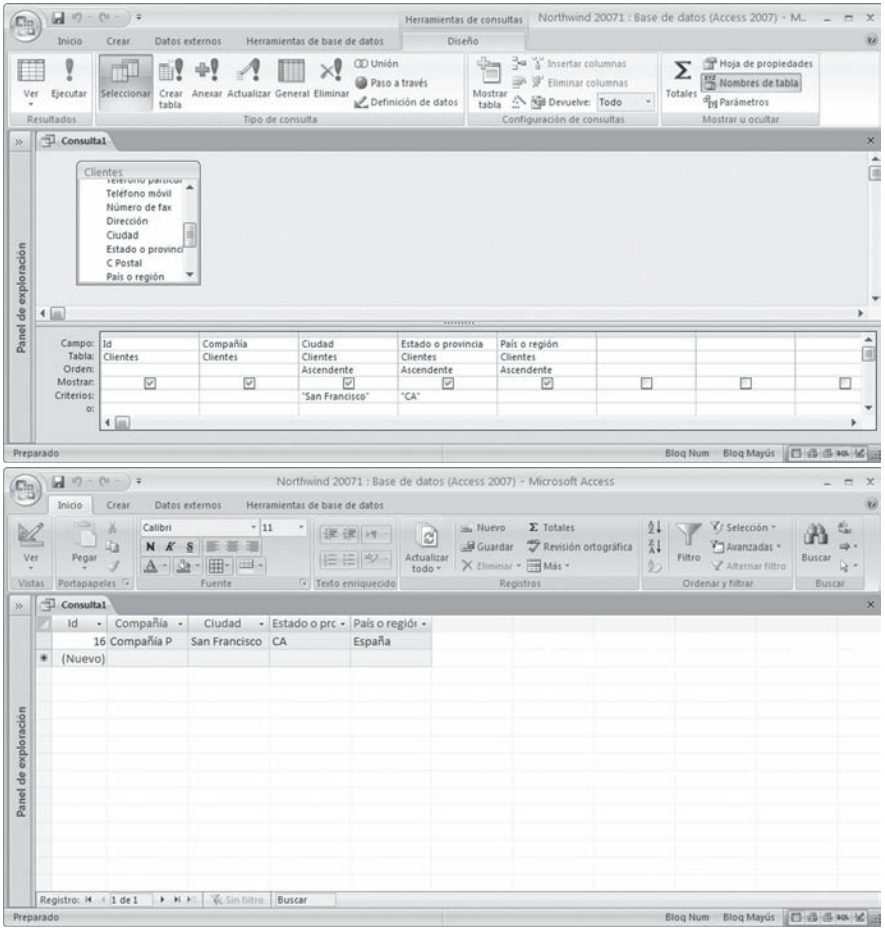


Figura 3-19 Pruebe esto 3-5 (Elección de las filas que se muestran), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-6 Selección de una fila combinada

Suponga que ahora quiere seleccionar a todos los clientes de Madrid, además de los de San Francisco. Debe agregar los criterios nuevos en una línea *diferente* del panel Diseño de consulta.

En este ejercicio Pruebe esto, modificará lo obtenido en la sección Pruebe esto 3-5 para incluir clientes adicionales.

Paso a paso

1. Debe comenzar con la especificación de consulta de Pruebe esto 3-5, igual que en la figura 3-19.
2. En la fila O escriba **Madrid** en la columna Estado o provincia. Observe que para que una fila aparezca en los resultados de la consulta, debe tener un valor de CA o Madrid en la columna Estado o provincia, y si el estado es CA, también debe tener un valor de San Francisco en la columna Ciudad. Los criterios en la misma línea se conectan con un **Y** lógico, mientras que las líneas de criterios se conectan con un **O** lógico.
3. Haga clic en el ícono Ejecutar de la cinta de opciones para aplicar su consulta. El panel completado se expone en la parte superior de la figura 3-20 y los resultados de la consulta en la parte inferior.
4. Como preparación para el ejercicio siguiente, regrese al panel Diseño de consulta al hacer clic en el ícono Ver bajo el botón de Office.

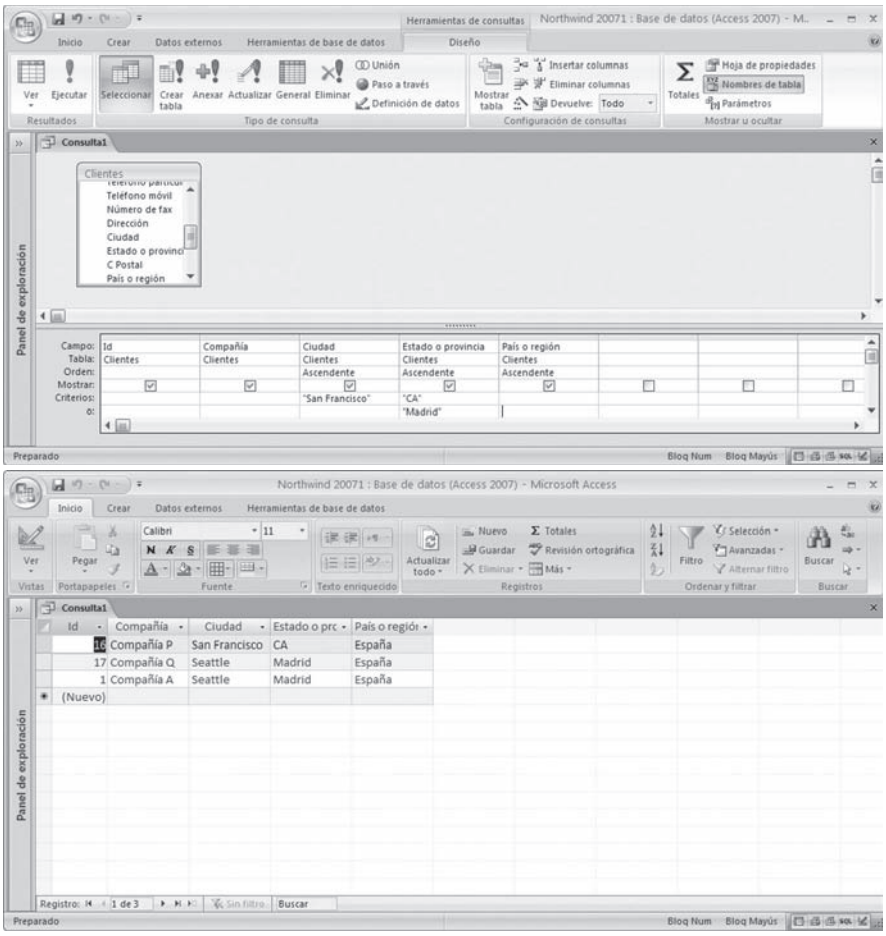


Figura 3-20 Pruebe esto 3-6 (Selección de una fila combinada), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-7 Uso del operador no es igual a

Hasta aquí se han analizado criterios de búsqueda que suponen el operador de comparación igual a (=). No obstante, es posible utilizar otros operadores de comparación, como se mencionó al principio de este capítulo. Por ejemplo, suponga que quiere presentar todos los clientes que no están en California (CA) ni en Madrid. El modo más fácil de hacer esto es emplear el operador no es igual a (<>).

(*continúa*)

Conforme las consultas se vuelven más complejas, encontrará que puede escribir la misma especificación de consulta de varias maneras, y éste es un ejemplo. Un modo consiste en escribir **<>CA Y <>Madrid** en una sola columna Estado o provincia. Otro es incluir la columna Estado o provincia en la consulta por segunda vez, quitar la marca de la casilla Mostrar, como lo hizo en la sección Pruebe esto 3-4, y escribir **<>CA** en una de las columnas Estado o provincia y **<>Madrid** en la *misma* fila Criterios de la otra columna Estado o provincia.

En este ejercicio, modificará la consulta de Pruebe esto 3-6 para hallar todos los clientes que no están en California (CA) ni en Madrid.

Paso a paso

1. Debe comenzar con una especificación de consulta semejante a la de la figura 3-20.
2. Borre todas las condiciones existentes en las líneas Criterios al seleccionar cada una (arrastre su cursor sobre ellas mientras mantiene oprimido el botón izquierdo de su ratón o dispositivo para apuntar) y luego oprima SUPR.
3. En una de las filas Criterios de la columna Estado o provincia, escriba esta condición: **<>CA Y <>Madrid**. Observe que Access puede cambiar de algún modo el formato, si elige algo más en el panel Diseño de consulta, pero el resultado será lógicamente igual.
4. Haga clic en el ícono Ejecutar, de la cinta de opciones para ejecutar su consulta. El panel completado se muestra en la parte superior de la figura 3-21 y los resultados de la consulta en la parte inferior.
5. Con el fin de prepararse para el siguiente ejercicio, haga esto:
   a. Regrese al panel Diseño de consulta, al hacer clic en el ícono Ver justo bajo el botón de Office.
   b. Haga clic en la tabla Clientes, en la parte superior del panel Diseño de consulta (el rectángulo que muestra el nombre de la tabla junto con una lista de algunos de los nombres de columnas) y luego presione SUPR. Esto borrará el formulario para que no contenga tablas, columnas ni criterios.

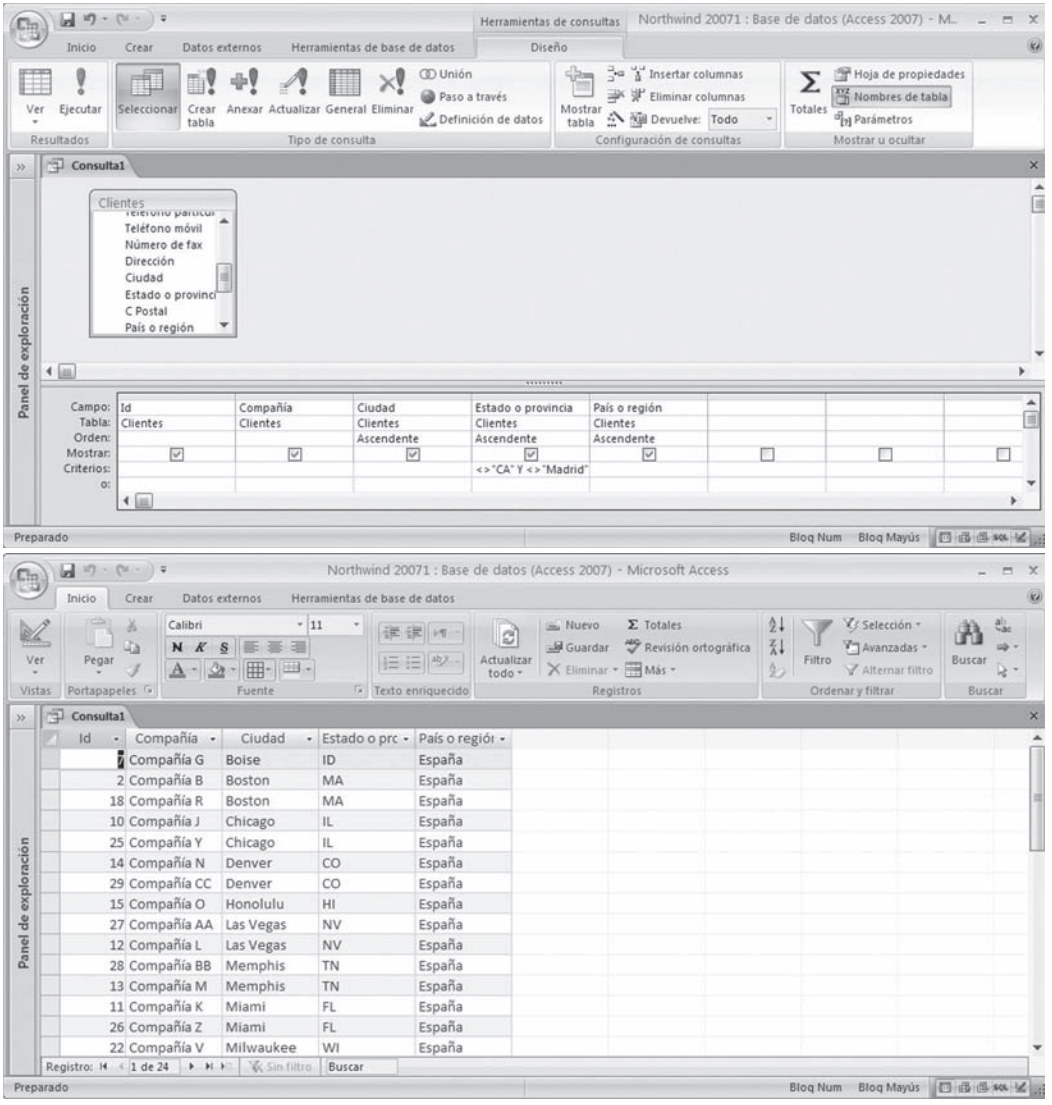


Figura 3-21 Pruebe esto 3-6 (Uso del operador no es igual a), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pregunta al experto

P: **En la sección Pruebe esto 3-7, se escribió** <>CA Y <>**Madrid para seleccionar todos los clientes que no eran de California ni de Washington. ¿No es** O **el operador lógico correcto aquí?**

R: El uso de O en este caso es absolutamente incorrecto. Las primeras veces que se preparan consultas en una base de datos, puede parecer extraño emplear el operador lógico Y aquí, pero si en lugar de eso emplea O, termina eligiendo todas las filas de la tabla Clientes (excepto las que tienen un valor NULL en la columna Estado o provincia). Ésta es la explicación. Si los criterios fueran **<>CA O <>Madrid**, se seleccionarían todas las filas de Madrid porque no es igual a CA (la condición en el lado izquierdo de **O** se evaluaría como Verdadera), se elegirían todas las filas de California porque CA no es igual a Madrid (la condición en el lado derecho de **O** se evaluaría como Verdadera) y se incluirían todas las otras filas con un valor NOT NULL en Estado o provincia porque las condiciones en ambos lados de **O** se evaluarían como Verdaderas.

Pruebe esto 3-8 Combinación de tablas

En este ejercicio, quiere mostrar tres columnas de la tabla Clientes junto con tres columnas de la tabla Pedidos para cada pedido que el cliente ha hecho con Northwind. En las bases de datos relacionales, a la unión de los datos de más de una tabla se le denomina *combinación*. Debido a que la relación entre los pedidos y los clientes es uno a varios, cuando un cliente tiene varios pedidos, se incluirá la misma información acerca del cliente en los resultados de la consulta para cada fila devuelta.

Es esencial comprender las combinaciones para familiarizarse con las bases de datos relacionales. Así como las relaciones uno a varios (implementadas en la base de datos como restricciones referenciales) son los bloques de construcción fundamentales de las bases de datos relacionales, así también lo son las combinaciones de las consultas en una base de datos relacional.

Paso a paso

1. Debe comenzar con un panel Diseño de consulta vacío (sin que se muestren tablas, columnas, criterios, etc.). Si éste no es el caso, seleccione (haga clic en) cada tabla presentada y oprima SUPR para quitarla de la consulta.

2. Haga clic en el ícono Mostrar tabla (con el signo de más en amarillo), para mostrar un cuadro de diálogo Mostrar tabla, igual que el expuesto en la figura 3-13.
3. Seleccione el nombre de la tabla Clientes, y luego haga clic en Agregar, para incluirla en la consulta.
4. Haga lo mismo con la tabla Pedidos, y luego cierre el cuadro de diálogo Agregar tabla. Observe la línea que conecta las dos tablas en el panel Diseño de consulta. Esto le indica que Access ya sabe cómo hacer coincidir las filas en estas dos tablas (Id de cliente como clave externa en la tabla Pedidos enlazada con Id como clave principal en la tabla Clientes) con base en los metadatos proporcionados por el diseñador de la base de datos en el panel Relaciones. En otras palabras, esta consulta *heredó* la relación entre las dos tablas especificada mucho tiempo antes en el panel Relaciones. Si no se incluyera la condición de combinación, como resultado obtendría un *producto cartesiano* (cada fila de una tabla combinada con cada fila de la otra; el *producto* de multiplicar las dos tablas entre sí) a menos que agregara la condición al arrastrar su apuntador de la columna de clave externa a la columna de clave principal (el método en Access para incluir en forma manual una condición de combinación). Es evidente que no quiere que en los resultados de su consulta parezca que cada cliente hizo cada pedido, de modo que Microsoft Access le ayuda a hacer lo correcto al heredar automáticamente la condición de combinación.
5. En la tabla Clientes, haga doble clic en las columnas Id, Compañía, Ciudad y Estado o provincia, para incorporarlas a la especificación de la consulta.
6. En la tabla Pedidos, haga doble clic en Fecha de pedido, Fecha de envío y Gastos de envío, para incluir estas columnas en la especificación de la consulta. Observe que no tiene que seleccionar la columna Id de cliente, aunque el criterio de combinación la utilizará para encontrar la fila correspondiente en la tabla Clientes.
7. Haga clic en el ícono Ejecutar, de la cinta de opciones, para ejecutar su consulta. El panel completado aparece en la parte superior de la figura 3-22 y los resultados de la consulta en la parte inferior. Observe la cuenta de registros en la parte inferior de los resultados de la consulta. Aunque sólo existen 29 clientes, el resultado contiene 48 filas. Esto se debe a que se han efectuado 48 pedidos. Cuando un cliente hace varios pedidos, en cada uno se repiten Id de compañía, nombre, ciudad y estado o provincia. Los clientes que no han hecho pedidos no se incluyen porque, como opción predeterminada, esta consulta emplea una *combinación interna*, en que sólo se muestran las filas coincidentes. En la sección Pruebe esto 3-10 probará una *combinación externa*, donde se incluyen las filas que no coinciden.
8. Con el fin de prepararse para el ejercicio siguiente, regrese al panel Diseño de consulta, al hacer clic en el ícono Ver, que se encuentra debajo el botón de Office.

(*continúa*)

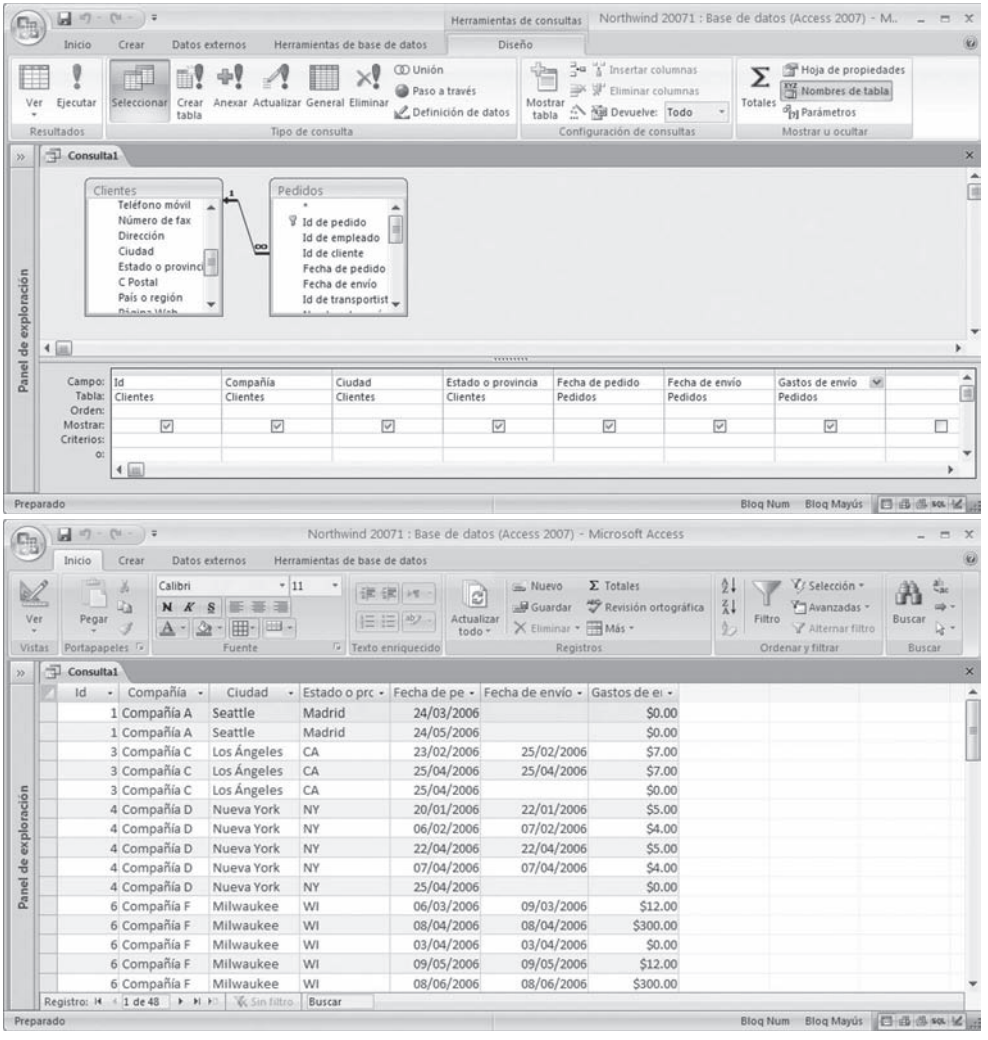


Figura 3-22 Pruebe esto 3-8 (Combinación de tablas), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-9 Limitación de los resultados de una combinación

En la sección Pruebe esto 3-8, combinó las tablas Clientes y Pedidos, pero los resultados contienen todos los pedidos y todos los clientes que tienen pedidos. Sin embargo, si no quiere ver todos los pedidos, puede emplear condiciones para limitar las filas de los resultados de la consulta, igual que lo hizo en los ejercicios anteriores. En este ejercicio Pruebe esto, limitará las filas para incluir sólo los clientes de California (CA) y sólo los pedidos del 1 de abril de 2006 (1/4/2006) o posteriores. Igual que en Pruebe esto 3-8, utilizará una combinación interna, lo que significa que los clientes de California que no tienen pedidos a partir del 1 de abril de 2006 no aparecerán en los resultados.

Paso a paso

1. Debe comenzar con la especificación de consulta de la sección Pruebe esto 3-8, igual que en la figura 3-22.
2. En la fila Criterios, escriba **CA** en la columna Estado o provincia.
3. En la fila Criterios, introduzca **>=1/4/2006** en la columna Fecha de pedido. Puede observar que Access cambia la condición al encerrar el valor de la fecha entre signos de número y añadiendo ceros (**>=#01/04/2006#**). Esto es simplemente el modo en que Access *delimita* un valor de fecha; casi todos los RDBMS emplean comillas sencillas a ambos lados de las cadenas de caracteres y los valores de fechas, de modo que ésta es una conducta atípica.
4. Haga clic en el ícono Ejecutar de la cinta de opciones para aplicar su consulta. El panel completado se expone en la parte superior de la figura 3-23 y los resultados de la consulta en la parte inferior.
5. Con el fin de prepararse para el ejercicio siguiente, regrese al panel Diseño de consulta, al hacer clic en el ícono Ver, que se encuentra debajo del botón de Office.

(*continúa*)

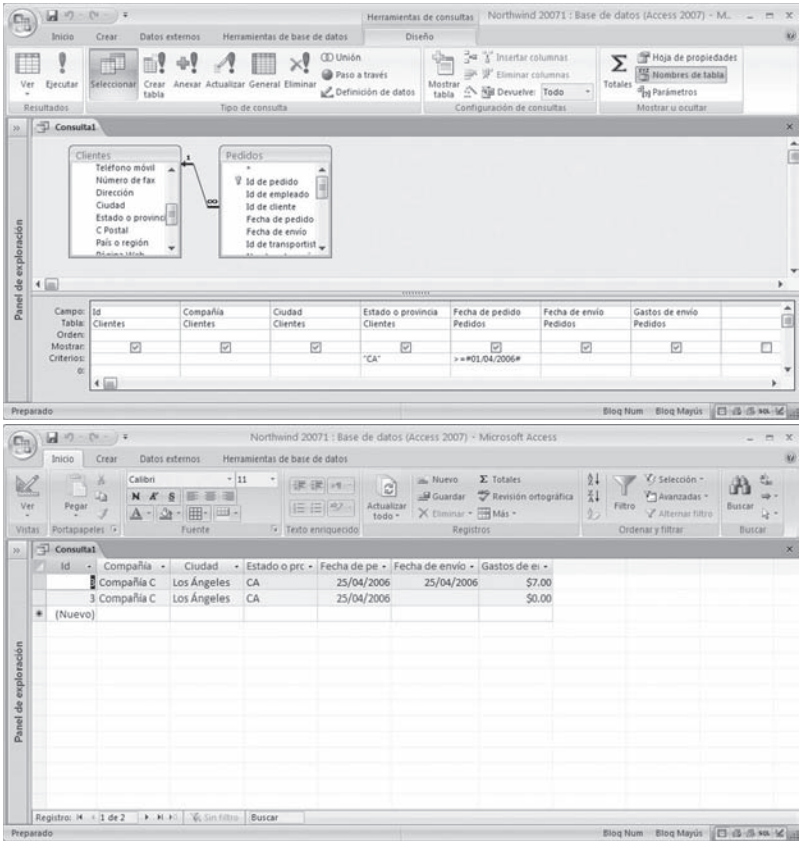


Figura 3-23 Pruebe esto 3-9 (Limitación de los resultados de una combinación), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-10 Combinaciones externas

Al igual que se describió en la sección Pruebe esto 3-9, la técnica que ha empleado hasta este momento es la *combinación interna*. Observe que algunos clientes de California no han hecho pedidos, de modo que los datos de esos clientes no aparecieron en los resultados de Pruebe esto 3-9. Si quiere incluir en los resultados a todos los clientes de California, sin tomar en cuenta que hayan hecho pedidos o no, debe usar una *combinación externa* (también llamada *combinación inclusiva*). Una combinación externa devuelve todas las filas de una (o ambas) tablas, sin tomar en cuenta si se encontraron filas coincidentes en las tablas combinadas. Cualquier dato que se va mostrar de la tabla donde no se encontró una fila coincidente se esta-

blece en NULL en los resultados de la consulta. (Para Microsoft Access, las columnas NULL aparecen en blanco.) Por ejemplo, para el cliente que no tiene pedidos, todas las columnas de la tabla Pedidos estarían en blanco en los resultados. Recuerde que las filas de datos devueltas todavía son filtradas mediante otros criterios de búsqueda (por ejemplo, sólo los clientes de California; sólo pedidos con fechas mayores o iguales a 1/4/2006), pero no es importante si el filtrado ocurre antes, durante o después de la operación de combinación, siempre y cuando se eliminen de la consulta las filas no deseadas. Recuerde que usted sólo describe el resultado que busca, no cómo lo obtiene.

Se emplean tres tipos de combinaciones externas y, por desgracia, la industria ha acordado que se les asignen nombres que pueden resultar confusos:

- **Combinación externa izquierda** Una combinación externa en que se devuelven todas las filas de la tabla que está en el lado izquierdo de la combinación, y también son devueltos los datos de cualquier fila que coincida con la tabla del lado derecho.
- **Combinación externa derecha** Una combinación externa en que se devuelven todas las filas de la tabla que está en el lado derecho de la combinación, y también son devueltos los datos de cualquier fila que coincidan con la tabla del lado izquierdo.
- **Combinación externa completa** Una combinación externa en que se devuelven todas las filas de ambas tablas, sin tomar en cuenta si se encuentran datos que coincidan entre ellas. En la actualidad Microsoft Access no permite este tipo de combinación.

La confusión mencionada proviene del uso de los términos *izquierda* y *derecha* en los nombres de los tipos de combinación. Si tan sólo invierte el orden de las tablas en cualquier consulta existente, en esencia convierte una combinación externa izquierda en una combinación externa derecha, o viceversa. Sin embargo, Microsoft Access no reconoce esta diferencia, y llama a todas las combinaciones simplemente *combinaciones externas*. Para ello, emplea un cuadro de diálogo llamado Propiedades de la combinación, presentado en la figura 3-24, para especificar el tipo de combinación que quiere usar, en que la opción predeterminada es una combinación interna.

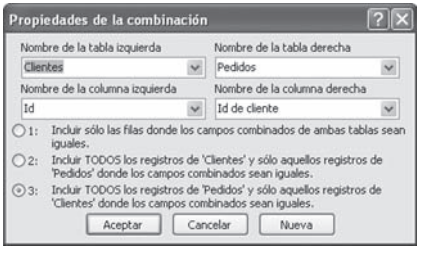


Figura 3-24 Cuadro de diálogo Propiedades de la combinación.

(*continúa*)

En este ejercicio Pruebe esto, transformará la consulta de la sección Pruebe esto 3-9 en una combinación externa para que se muestren todos los clientes de California, sin tomar en cuenta si han hecho pedidos después del 1/4/2006.

Paso a paso

1. Debe comenzar con la especificación de consulta de Pruebe esto 3-9, tal como se presenta en la figura 3-23.
2. Para acceder al cuadro de diálogo Propiedades de la combinación (mostrado en la figura 3-24), haga doble clic en algún punto de la línea entre las dos tablas mostradas en el panel Diseño de consulta, o haga clic con el botón derecho del ratón sobre la línea. Igual que con el panel Relaciones, puede resultar complicado colocar el puntero del cursor exactamente en el lugar correcto de la línea, pero un poco de práctica y paciencia siempre funcionan.
3. En el cuadro de diálogo Propiedades de la combinación, seleccione la opción Incluir TODOS los registros de 'Clientes' y sólo aquellos registros de 'Pedidos' donde los campos combinados sean iguales. Es más probable que sea la opción 2, pero si agregó las tablas a la consulta en el orden inverso, pudo haber terminado como la opción 3. Haga clic en Aceptar para cerrar el cuadro de diálogo.
4. Debido a que tiene una condición en Fecha del pedido de la tabla Pedidos, necesita modificarla para permitir valores nulos. En el caso de los clientes que no tienen pedidos, el valor de la columna Fecha de pedido será NULL. Añada la condición **O Es Nulo** (que también puede escribir como Or Is Null) en la columna Fecha del pedido.
5. Haga clic en el ícono Ejecutar, de la cinta de opciones para aplicar su consulta. El panel completado se expone en la parte superior de la figura 3-25 y los resultados de la consulta en la parte inferior. Observe la flecha en la línea entre las dos tablas que apunta hacia las tablas Pedidos. Éste es el modo en que Access le indica que la combinación es externa.
6. Con el propósito de prepararse para el ejercicio siguiente, regrese al panel Diseño de consulta, al hacer clic en el ícono Ver, que se encuentra justo debajo del botón de Office.

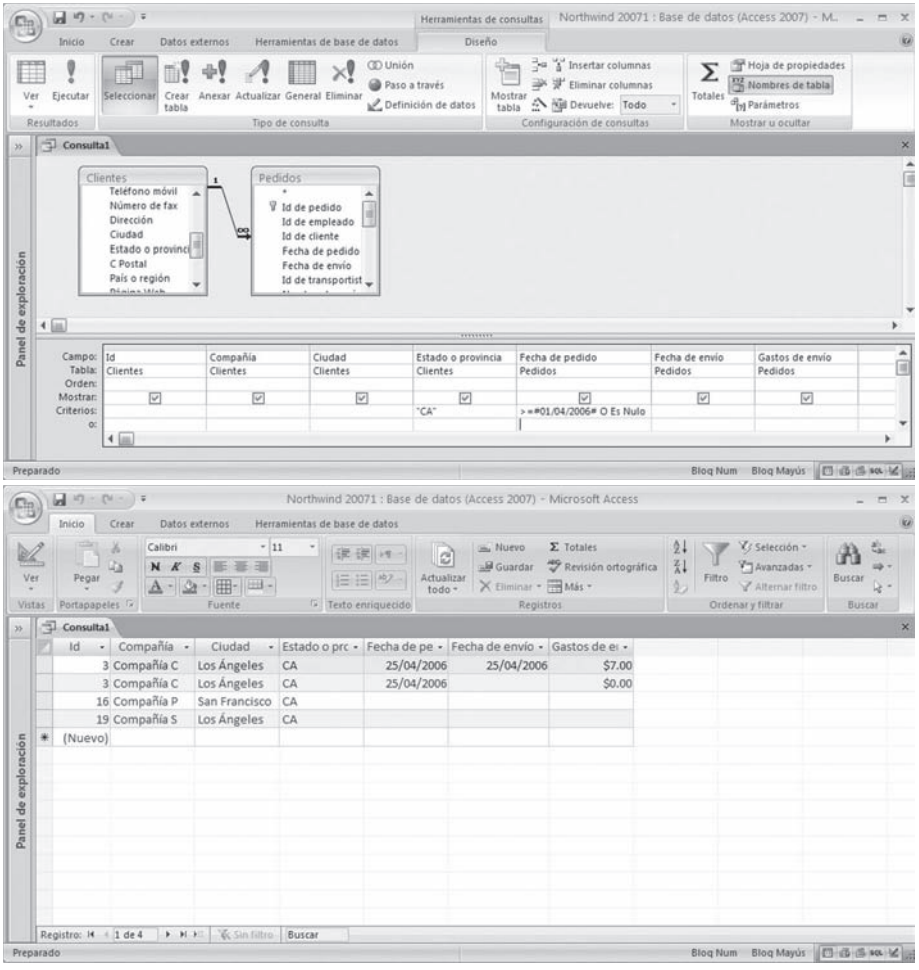


Figura 3-25 Pruebe esto 3-10 (Combinaciones externas), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-11 SQL en Microsoft Access

SQL se examina en el capítulo 4; sin embargo, como Microsoft Access genera automáticamente SQL para las consultas definidas en el panel Diseño de consulta, viene al caso un breve contacto con SQL. Para este ejercicio Pruebe esto, se mostrará el SQL para la consulta creada en Pruebe esto 3-10.

(*continúa*)

Paso a paso

1. Debe comenzar con la especificación de la consulta de la sección Pruebe esto 3-10 que se muestra en la figura 3-25.
2. En el panel Diseño de consulta, haga clic en la flecha que se encuentra bajo el ícono Ver (debajo del botón de Office) para ampliar las opciones. Seleccione la opción Ver SQL, como se aprecia en la parte superior de la figura 3-26. O puede hacer clic en el ícono de SQL que se encuentra en la barra de estado, en la esquina inferior derecha del panel. Una de las nuevas características de Office 2007 es la conmutación entre las funciones de zoom y vista/ventana en la barra de estado, que aparece en la parte inferior de diversos paneles de aplicación.
3. El SQL para la consulta actual se mostrará igual que en la parte inferior de la figura 3-26. Después de la palabra clave **SELECT** se ve una lista de las columnas que aparecerán en los resultados de la consulta. Luego de la palabra clave **FROM** están dos tablas y su condición de combinación externa. Al final está la palabra clave **WHERE**, seguida por las condiciones que limitan las filas a los clientes de California y a las fechas del pedido que son NULL o 1/4/2006 o después. Ésta es una magnífica función del producto, porque puede usarla no sólo para aprender SQL sino, una vez que ya lo conoce, puede pasar con rapidez de la Vista Diseño, de la consulta, a la Vista SQL, y viceversa, para crear consultas con rapidez. (Por cierto, Access SQL es el que menos se apega a las normas entre todos los RDBMS modernos, porque los nombres de objetos pueden contener espacios.)
4. Con el fin de prepararse para el ejercicio siguiente, haga esto:
   a. Regrese al panel Diseño de consulta, al hacer clic en el ícono Ver, que se encuentra debajo del botón de Office.
   b. Borre todas las columnas de criterios seleccionados al arrastrar el puntero de su ratón sobre las delgadas franjas grises sobre cada columna (justo arriba de la etiqueta Campo:). Las columnas se desplegarán en negro (video invertido) cuando las seleccione. Luego oprima SUPR para eliminarlas de la consulta.
   c. Devuelva la combinación entre las tablas Clientes y Pedidos a una combinación interna. Para esto, haga doble clic en la parte delgada de la línea entre las dos tablas expuestas en el panel Diseño de consulta, para desplegar el cuadro de diálogo Propiedades de combinación. Luego seleccione la opción 1 y haga clic en Aceptar.

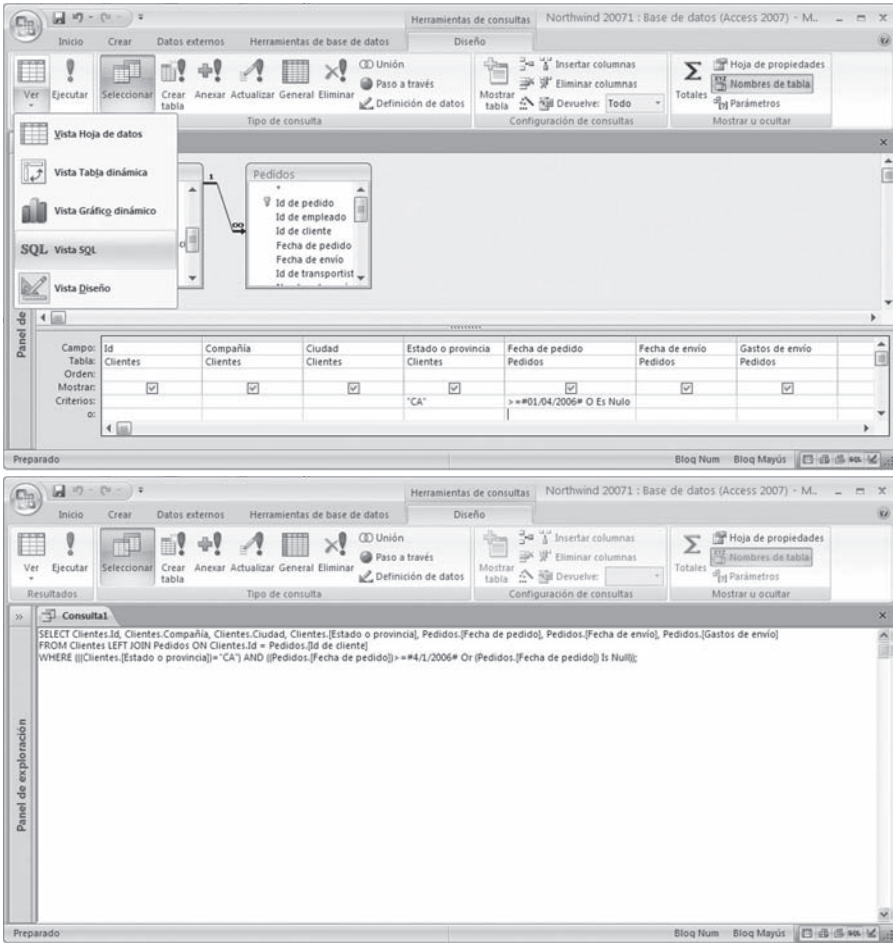


Figura 3-26 Pruebe esto 3-11 (SQL en Microsoft Access), diseño de consulta (arriba) y consulta SQL generada (abajo).

Pruebe esto 3-12 Combinaciones múltiples y columnas calculadas

Cuando necesite información de más de dos tablas en el mismo resultado de consulta, basta con agregar más tablas a la consulta y, por lo tanto, más operaciones de combinación. La be-

(*continúa*)

lleza de las bases de datos relacionales es que no necesita preocuparse por cuál combinación se procesa primero y otros detalles de la implementación. Puede confiar en el RDBMS para que tome esas decisiones.

Para este ejercicio Pruebe esto, considere otra situación: es posible que quiera saber el valor total en dólares de los artículos pedidos por los clientes de Florida. Al analizar las tablas que tiene disponibles, comprende que necesita la tabla Clientes, de modo que pueda filtrarla por la columna Estado o provincia, y la tabla Detalles de pedido, porque contiene los datos que necesita para calcular el valor total de cada artículo pedido; por ejemplo, la cantidad solicitada y el precio unitario de cada artículo. No obstante, no hay manera de combinar estas dos tablas directamente de una manera significativa. Si se fija en el panel Relaciones (consulte la figura 3-9), la solución se vuelve obvia: también necesita la tabla Pedidos. Después puede emplear la tabla Clientes para hallar los clientes en Florida, combinar las filas de la tabla Pedidos cuya Id (clave principal) coincida con la columna Id de cliente de la tabla Pedidos (la clave externa) para hallar los pedidos de los clientes de Florida y, por último, combinar esas filas con la tabla Detalles de pedido, para hallar los artículos de línea de esos pedidos. (Por supuesto, no está garantizado que el RDBMS procese realmente las combinaciones en esta secuencia, pero a pesar de todo el resultado final será el mismo). A partir de este ejemplo, debe ser evidente que un diagrama general para todas sus tablas y relaciones es un documento *esencial*, porque le proporciona la guía que necesita para formar consultas.

Este ejemplo también requiere una *columna calculada* (también llamada columna *derivada*), que se forma al multiplicar los valores de las columnas Precio unitario y Cantidad en cada fila. En una consulta a una base de datos relacional puede usar casi cualquier fórmula que pueda aplicar en una hoja de cálculo.

Paso a paso

1. Debe comenzar con una especificación de consulta que una las tablas Clientes y Pedidos con una especificación de combinación (una línea entre ellas) y ninguna otra condición, como aparece en la sección Pruebe esto 3-8 (figura 3-22). Compruebe que la combinación entre Clientes y Pedidos sea interna y que en este momento no haya columnas incluidas en la especificación de la consulta.
2. Agregue la tabla Detalles de pedido a la consulta, al hacer clic en el ícono Mostrar tabla y seleccionar la tabla de la lista del cuadro de diálogo Mostrar tabla.
3. En la tabla Clientes, agregue a la consulta las columnas Compañía y Estado o provincia, al hacer doble clic en sus nombres. O bien, puede arrastrar y colocar el nombre de la columna en la sección correspondiente de la especificación de la consulta.

4. En la tabla Detalles de pedido, agregue a la consulta las columnas Precio y Cantidad.
5. Para agregar la columna calculada, escriba lo siguiente en la fila Campo de la columna vacía que se encuentra a la derecha de la columna Cantidad: **Precio extendido: Precio * Cantidad**. La primera parte de la entrada es el *encabezado* de la columna nueva. Cada columna de sus resultados debe tener un nombre único, y si no lo asigna, Microsoft lo hará. Los nombres de columna predeterminados no suelen tener significado alguno y, en ocasiones, son simplemente deficientes, de modo que *siempre* es mejor proponer un encabezado (nombre) de columna para las columnas calculadas. Observe que no importan los espacios a cada lado del operador de multiplicación en las especificaciones del campo, de modo que puede descartarlos. Sin embargo, si alguna columna incluyera espacios, debe respetarlos, pero incluir el nombre completo entre corchetes. Es probable que Microsoft Access vuelva a escribir la especificación de su columna al eliminar los espacios y colocar corchetes alrededor del otro nombre de columna, de modo que no se sorprenda si ve que lo que ha escrito cambia en el panel cuando mueve el cursor a otro lugar.
6. Para limitar la consulta sólo a los clientes en Florida (FL), escriba **FL** en la fila Criterios, de la columna Estado o provincia.
7. Agregue un orden ascendente a la columnas Compañía ya sea el escribir la letra **A** en la fila Ordenar para la columna y luego oprimir Intro, o al hacer clic en ese lugar y seleccionar Ascendente de la lista.
8. Haga clic en el ícono Ejecutar, de la cinta de opciones, para aplicar la consulta. El panel completado se muestra en la parte superior de la figura 3-27 y los resultados de la consulta aparecen en la parte inferior.
9. Con el fin de prepararse para el siguiente ejercicio, regrese al panel Diseño de consulta, al hacer clic en el ícono Ver, que se encuentra justo debajo del botón de Office.

(*continúa*)

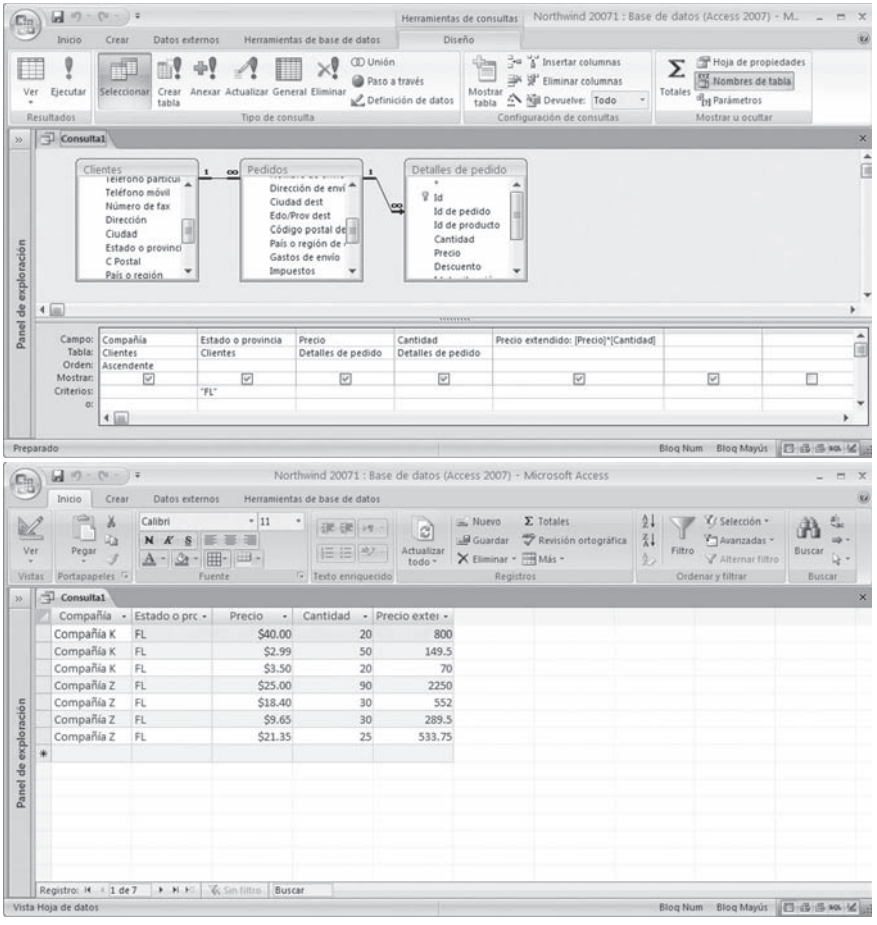


Figura 3-27 Pruebe esto 3-12 (Combinaciones múltiples y columnas calculadas), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-13 Funciones de obtención de totales

Al revisar los resultados de la sección Pruebe esto 3-12, es probable que haya observado que se devolvieron siete filas que cubrían pedidos de dos clientes diferentes en Florida. Todos los detalles están aquí, pero de un vistazo es difícil captar la cantidad *total* que cada cliente ha pedido a Northwind. Lo que en verdad necesita hacer es obtener el total de la columna Precio extendido para cada cliente. En las bases de datos relacionales, esto se hace con la función **SUMA**.

Una *función* es un tipo especial de programa que devuelve un valor único cada vez que se invoca, y cuyo nombre responde al concepto matemático de una función. Debido a que empleará la función para accionar sobre una columna, será invocada para cada fila y, por lo tanto, devolverá un valor único para cada fila que maneje la consulta. A veces se emplea el término *función de columna* para recordarle que la función se aplica a una columna de una tabla o vista. Un ejemplo de una función de columna sencilla es **ROUND**, la cual se usa para redondear números de diversas maneras. A las clases especiales de funciones que combinan varias filas en una sola se les denomina funciones de *obtención de totales*. En la tabla siguiente presentan las funciones de obtención de totales que se emplean a menudo en las bases de datos relacionales:

Nombre de función	Descripción
PROMEDIO	Calcula el valor promedio de una columna.
CUENTA	Cuenta la cantidad de valores encontrados en una columna.
MÁX	Determina el valor máximo en una columna.
MÍN	Halla el valor mínimo en una columna.
SUMA	Suma (totaliza) los valores en una columna.

Si utiliza una sola función de obtención de totales en una consulta, obtiene una fila para toda la consulta. Esto es correcto, porque no hay modo de que el RDBMS se entere de cuál otro resultado busca. De modo que, si quiere que el resultado del total sea de *grupos* de filas en la consulta, necesita incluir una especificación **AGRUPAR POR** para indicar al RDBMS que agrupe las filas por los valores de una o más columnas, y que aplique la función de obtención de totales a cada grupo. Esto es muy similar a pedir subtotales en lugar de un gran total para una lista de números.

Para este ejercicio, quiere que el RDBMS proporcione un total de la columna calculada Precio extendido para cada cliente. En otras palabras, quiere agrupar las filas por cliente y, para cada grupo, mostrar una sola fila que contenga el nombre de la compañía, el estado o provincia, y el importe total del pedido.

La columna Estado o provincia en realidad no es necesaria, porque en esta consulta sólo se incluyen los clientes de Florida. Sin embargo, se deja aquí para ejemplificar un concepto importante y difícil de comprender para casi todas las personas que no conocen las bases de datos relacionales: si selecciona las columnas Compañía, Estado o provincia y Precio total calculado, indica al RDBMS la fórmula para calcular el precio total y le pide que agrupe las filas en el resultado por Compañía, hay un problema de lógica oculto que hará que el RDBMS devuelva un error. En esencia, le ha pedido al RDBMS que devuelva el valor de Estado o provincia de cada fila en los resultados de la consulta pero, al mismo tiempo, que totalice las filas

(*continúa*)

por Compañía y proporcione el total calculado de cada total. Es ilógico pedir que se obtengan totales de algunas filas y de otras no. Para empeorar las cosas, el mensaje de error resultante resulta muy confuso. No es de extrañar que a las funciones de obtención de totales se les denomine funciones de complicación. Recuerde esta regla: cuando una consulta incluye una de estas funciones, *cada* columna de los resultados de la consulta debe formarse mediante una función de obtención de totales o incluyendo su nombre en la lista de la columna AGRUPAR POR. En Microsoft Access, el ícono Totales (la letra griega sigma) en la cinta de opciones conmuta (oculta y expone) una línea llamada Total en el panel Vista de consulta. Es la línea del total que le permite especificar las funciones de obtención de totales y los agrupamientos para su consulta.

Paso a paso

1. Debe iniciar con la especificación de consulta de la sección Pruebe esto 3-12, igual que en la figura 3-27.
2. Elimine las columnas Precio unitario y Cantidad al hacer clic en la franja gris, que se encuentra sobre el nombre del campo, y oprimir SUPR.
3. Cambie el encabezado de la columna Precio extendido a Precio total. Este nombre de columna describe mejor los resultados.
4. Haga clic en el ícono Totales, en la cinta de opciones, para mostrar la línea Total en la especificación de la consulta. Como opción predeterminada, cada columna tendrá al principio Agrupar por especificada en esa línea.
5. En la columna Precio total, haga clic en la línea Total y emplee la lista y el menú desplegable para seleccionar la función Suma.
6. Haga clic en el ícono Ejecutar de la cinta de opciones para aplicar su consulta. El panel completado se muestra en la parte superior de la figura 3-28 y los resultados de la consulta en la parte inferior.
7. Para completar este ejercicio, cierre el panel Diseño de consulta, al hacer clic en el botón Cerrar en la esquina superior derecha del panel (tenga cuidado de no hacer clic en el botón de la esquina superior derecha de la pantalla de Microsoft Access, porque esto cierra por completo la base de datos de Access), o al hacer clic con el botón derecho del ratón en la ficha que muestra el nombre de la consulta (nombre probable: Consulta1) y seleccionar Cerrar. Cuando aparezca un mensaje sobre guardar la consulta, haga clic en No.

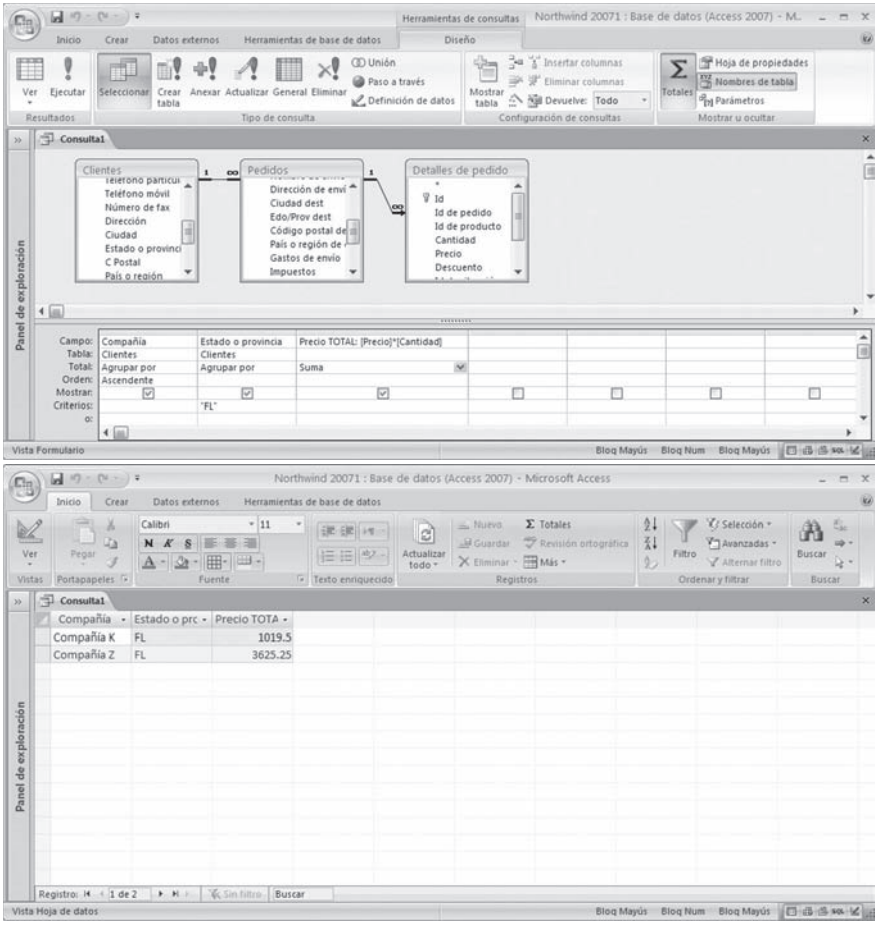


Figura 3-28 Pruebe esto 3-13 (Funciones de obtención de totales), diseño de consulta (arriba) y resultados de la consulta (abajo).

Pruebe esto 3-14 Autocombinaciones

Cuando las tablas tienen incorporada una relación recursiva, debe utilizar una *autocombinación* (unir una tabla consigo misma) para resolver la relación. Por desgracia, la versión 2007 de la base de datos Northwind no tiene incorporada una relación recursiva, de modo que tendrá que agregar una para facilitar una demostración de este importante concepto.

(*continúa*)

En este ejercicio Pruebe esto, primero agregará una columna Id de gerente a la tabla Empleados. Luego incluirá algunos datos en la columna para que todos los empleados, excepto el que está en la parte superior de la jerarquía de administración, tengan asignada la Id de gerente en la consulta correspondiente. Por último, creará una consulta que muestre Id, apellido y cargo de cada empleado, junto con el nombre del gerente. Para obtener el nombre del gerente, primero tendrá que combinar la tabla Empleados consigo misma para que Access pueda hacer que coincida Id de gerente (clave externa) con la fila en la tabla Empleados que contenga el nombre del gerente.

Paso a paso

1. Para agregar Id de gerente a la tabla Empleados, haga lo siguiente:

a. Abra el panel Vista Diseño de la tabla Empleados, igual que en la figura 3-11. Para ello, amplíe el Panel de exploración que se encuentra en el borde izquierdo del panel principal de Access, encuentre la tabla Empleados en la lista de objetos, haga clic con el botón derecho del ratón en su nombre y, luego, en Vista Diseño, en el menú desplegable.

b. Desplácese por las definiciones de campos (las filas en la parte superior del panel Vista Diseño) hasta que encuentre la primera donde Nombre del campo está en blanco. Escriba **Id de gerente** en la columna Nombre del campo, y seleccione Número, de la lista desplegable en la columna Tipo de datos. La entrada completada debe parecerse a la mostrada en la figura 3-29.

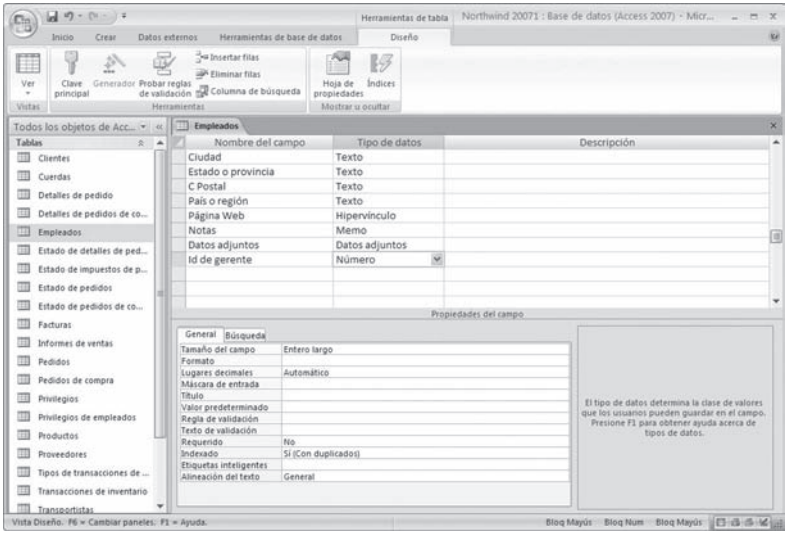


Figura 3-29 Tabla Empleados (Vista Diseño), con la columna Id de gerente agregada.

2. Para llenar con datos la columna Id de gerente recién agregada, haga lo siguiente:

a. Haga clic en el ícono Vista, en la cinta de opciones, para mostrar las filas y columnas de datos de la tabla Empleados, igual a lo mostrado en la figura 3-10.

b. Desplácese a la derecha con la barra de desplazamiento que se encuentra en la parte inferior del panel, para que se vea la columna Id de gerente. Debe ser la penúltima. (Observe que la columna mostrada en el extremo derecho es para agregar un nuevo campo [columna] a la tabla. Pudo haber agregado la nueva columna Id de gerente con esta opción, pero es preferible emplear la Vista Diseño para hacer cambios en la definición de la tabla, porque están disponibles muchas opciones más.)

c. Escriba los valores de datos en la columna Id de gerente, como se aprecia en la figura 3-30. Observe que no se incluyen valores en la segunda y última filas del panel. La segunda fila es para el vicepresidente de ventas, Jesús Escolar, quien es el gerente con más antigüedad incluido en la tabla (en este momento, el gerente que le corresponde no está en la tabla, de modo que deje en blanco su Id de gerente, lo que en realidad es un valor nulo). La última fila es para agregar empleados nuevos a la tabla, y como no piensa hacer eso, sino que sólo va a actualizar los existentes, debe dejar en blanco todos los valores de esta última fila.

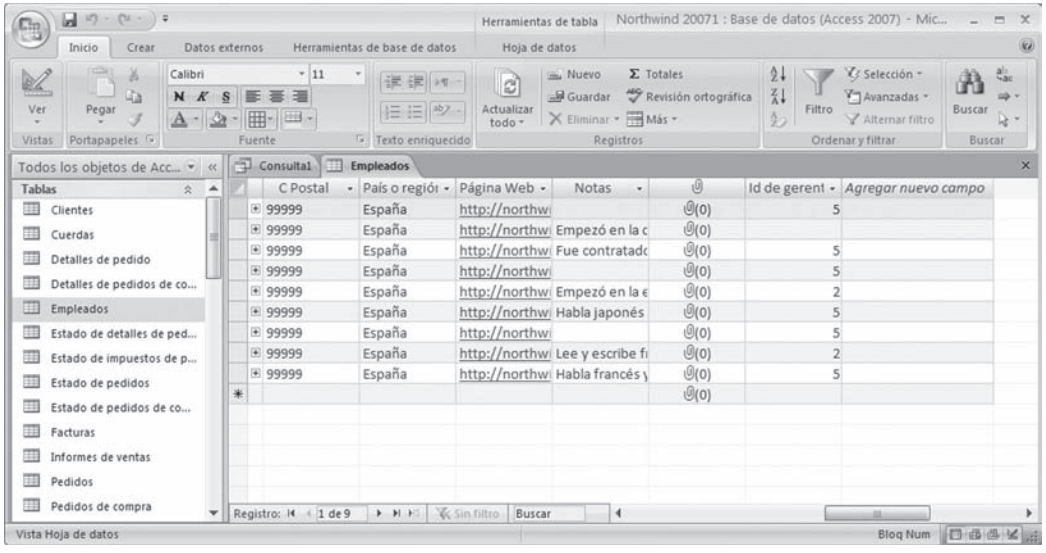


Figura 3-30 Tabla Empleados (Vista Hoja de datos), con los valores de Id de gerente agregados.

(*continúa*)

3. Podría (y es probable que deba) emplear el panel Relaciones para agregar la relación de la columna Id de gerente a la de clave principal, para definir a la primera como clave externa, pero como la columna era simplemente para demostración, puede saltar ese paso.
4. Cree una consulta nueva al abrir la cinta de opciones Crear y luego hacer clic en el ícono Diseño de consulta.
5. Cuando se abra el cuadro de diálogo Mostrar tabla, agregue *dos veces* la tabla Empleados a la consulta. Al principio, esto puede parecer extraño, pero es el único modo de indicar a Microsoft Access que se propone hacer que coincida cada fila de la tabla Empleados con una fila diferente (la del gerente) en la misma tabla. Observe que las tablas tienen los nombres Empleados y Empleados_1 en el panel, aunque en realidad las dos son representaciones de la misma tabla exacta.
6. Si quiere, puede minimizar el Panel de exhibición y cerrar la tabla Empleados (Vista Diseño) para reducir la cantidad de elementos que se ven en la pantalla.
7. En la tabla Empleados (en el lado izquierdo), desplácese hasta que vea la columna Id de gerente. Haga clic en su nombre y (mientras mantiene oprimido el botón del ratón) arrastre y suelte el nombre sobre la columna Id, en la tabla Empleados_1. Esto indica a Access cómo combinar la tabla empleados consigo misma. La tabla del lado izquierdo representa a los empleados y la del lado derecho es donde encontrará al gerente de cada empleado. No se preocupe si esto todavía parece confuso: repasará las relaciones recursivas en otros capítulos del libro.
8. Usted quiere que se muestre la fila de Juan Escolar, pero como no hay un gerente en la tabla, necesita cambiar la combinación a externa, para ver su fila. Haga doble clic en la línea entre las dos tablas, seleccione la Opción 2 del cuadro de diálogo Propiedades de la combinación, y haga clic en Aceptar.
9. De la tabla Empleados, seleccione las columnas Id, Apellidos, Nombre y Cargo al hacer doble clic en cada una de ellas.
10. De la tabla Empleados_1, seleccione la columna Apellidos al hacer doble clic en su nombre.
11. En este momento, en la consulta tiene dos columnas llamadas Apellidos. Necesita cambiar una de ellas para evitar confusiones y cumplir con el principio del RDBMS de que cada columna tenga un nombre único. En la columna Apellidos de la tabla Empleados_1 (la del extremo derecho en la especificación de la consulta), haga clic a la izquierda del nombre de la columna y escriba **Gerente:**, lo que asigna un nombre de alias a la columna en la consulta.
12. Haga clic en el ícono Ejecutar, en la cinta de opciones, para realizar su consulta. El panel completado se presenta en la parte superior de la figura 3-31 y los resultados de la consulta en la parte inferior. Sus resultados deben ser similares, pero si no especificó un orden, el orden de las filas en sus resultados puede ser diferente.

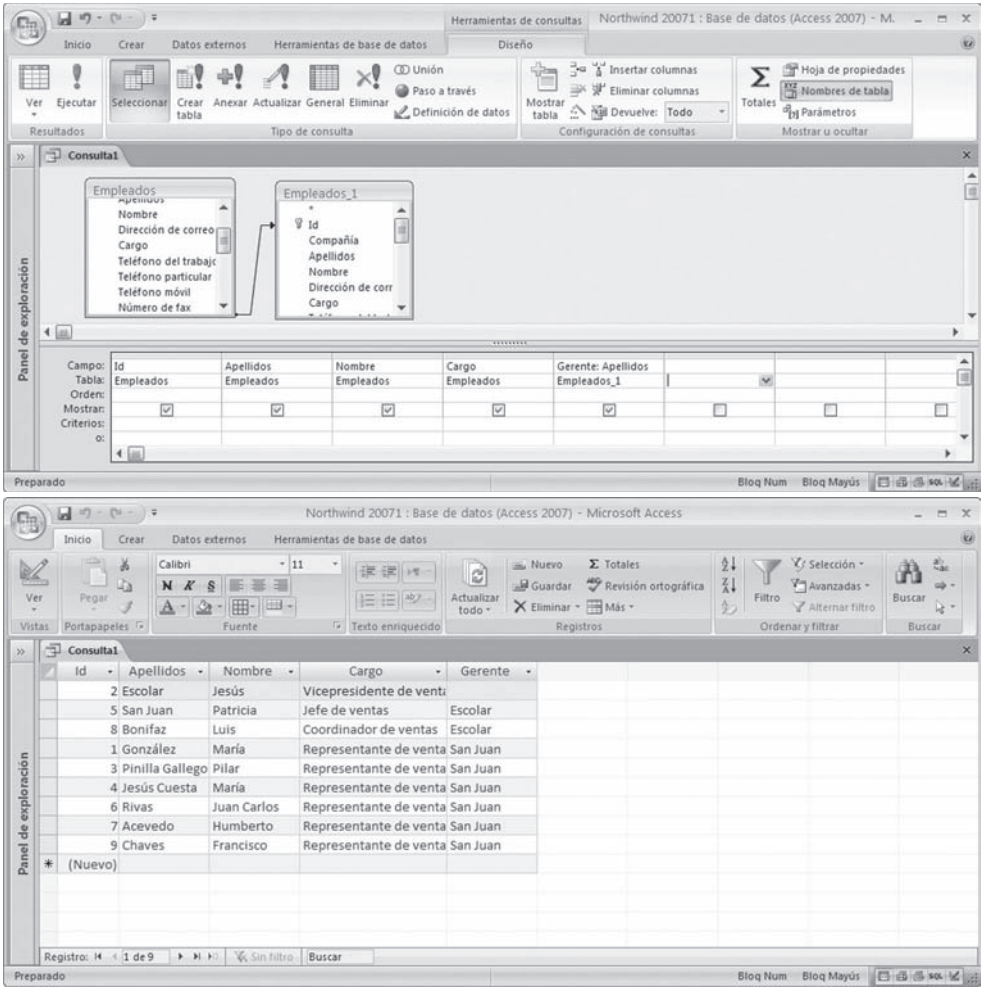


Figura 3-31 Pruebe esto 3-14 (Autocombinaciones), diseño de consulta (arriba) y resultados de la consulta (abajo).

13. Para completar este ejercicio, cierre el panel Diseño de consulta, al hacer clic en el cuadro Cerrar, en la esquina superior derecha del panel, o haga clic con el botón derecho del ratón en la ficha que muestra el nombre de la consulta (probablemente Consulta1) y seleccione Cerrar. Cuando un mensaje le pregunte si quiere guardar la consulta, haga clic en No. Después puede cerrar Microsoft Access, si lo desea.

(*continúa*)

Resumen de Pruebe esto

En los 14 ejercicios Pruebe esto de este capítulo exploró las consultas de Microsoft Access de una manera diseñada para mostrar las funciones básicas que utilizará con más frecuencia. Es obvio que existen muchas funciones más por explorar. Pero es el momento de pasar a SQL, el tema del siguiente capítulo.

✓ Autoexamen Capítulo 3

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. Un lenguaje de consultas mediante formularios

- **A** Fue desarrollado por primera vez por IBM en la década de 1980.
- **B** Describe cómo debe procesarse una consulta y no cuáles deben ser los resultados.
- **C** Se parece a SQL.
- **D** Usa una GUI (interfaz gráfica de usuario).
- **E** Se demostró que es claramente superior en estudios controlados.

2. Los tipos de objetos en Microsoft Access que se relacionan estrictamente con la administración de la base de datos (en contraste con el desarrollo de una aplicación) son

- **A** Las tablas.
- **B** Las consultas.
- **C** Los formularios.
- **D** Las macros.
- **E** Los módulos.

3. Cuando se elimina una tabla del panel Relaciones de Microsoft Access, ¿qué ocurre después?

- **A** Es eliminada inmediatamente de la base de datos.
- **B** Permanece intacta en la base de datos y sólo es retirada del panel Relaciones.
- **C** Se mantiene en la base de datos, pero se eliminan todas las filas de datos.
- **D** También se eliminan las relaciones que pertenecen a la tabla.

4. Las relaciones en el panel Relaciones de Microsoft Access representan _____ en la base de datos.

5. Una columna en los resultados de una consulta de Microsoft Access se forma a partir de

- **A** Una columna de una tabla.
- **B** Una columna de una consulta.
- **C** Una constante.
- **D** Un cálculo.
- **E** Todos los anteriores.

6. Cuando se ejecuta una consulta sin criterios incluidos, el resultado es

- **A** Un mensaje de error.
- **B** No se exhiben filas.
- **C** Se exhiben todas las filas de la tabla.
- **D** Ninguno de los anteriores.

7. Cuando no se incluye el ordenamiento de las filas en una consulta a la base de datos, las filas devueltas por la consulta están en orden _____.

8. En una consulta, el criterio de búsqueda REGIÓN NO = "CA" O REGIÓN NO = "NV" mostrará

- **A** Un mensaje de error.
- **B** Todas las filas de la tabla.
- **C** Sólo las filas en que REGIÓN es igual a "CA" o "NV".
- **D** Todas las filas de la tabla, excepto las que tienen NULL en REGIÓN.
- **E** Todas las filas de la tabla, excepto las que tienen un valor "CA" o "NV" en REGIÓN.

9. Los criterios en líneas diferentes de una consulta de Microsoft Access se conectan con el operador lógico _____.

10. El conector de combinación entre las tablas en una consulta de Microsoft Access puede

- **A** Crearse de manera manual al arrastrar una columna de una tabla o vista a una columna de otra tabla o vista.
- **B** Heredarse de los metadatos definidos en el panel Relaciones.
- **C** Ser alterado para definir combinaciones externas izquierda, derecha y completa.
- **D** Generar un producto cartesiano, si no se define entre dos tablas o vistas de la consulta.
- **E** Todos los anteriores.

11. Cuando se emplea una combinación externa, los datos de columna de las tablas (o vistas) en que no se encontraron filas que coincidieran contendrán _____.

12. Una función de obtención de totales

- **A** Combina los datos de varias columnas.
- **B** Combina los datos de varias filas.
- **C** Puede aplicarse a columnas de tablas, pero no a columnas calculadas.
- **D** Requiere que cada columna de una consulta sea una función de totalización o aparezca por nombre en la lista AGRUPAR POR de la consulta.
- **E** Todos los anteriores.

13. Las autocombinaciones de una consulta son un método para resolver una _____.

14. El nombre de una columna calculada en los resultados de la consulta es _____ cuando no es proporcionado en la definición de la consulta.

15. Las tablas se pueden combinar

- **A** Usando sólo una clave principal en una tabla y una clave externa en otra.
- **B** Mediante cualquier columna en cualquier tabla (en teoría).
- **C** Sólo consigo mismas.
- **D** Sólo con otras tablas.
- **E** Sólo mediante una fórmula de un producto cartesiano.

Capítulo 4

Introducción a SQL

Habilidades y conceptos clave

- Breve historia de SQL
- Introducción a Oracle SQL
- ¿Dónde están los datos?
- Lenguaje de consulta de datos (DQL): la instrucción SELECT
- Lenguaje de manipulación de datos (DML)
- Instrucciones del lenguaje de definición de datos (DDL)
- Instrucciones del lenguaje de control de datos (DCL)

En este capítulo se presenta el lenguaje de consulta estructurada (Structured Query Language, SQL), que se ha convertido en el idioma universal de las bases de datos relacionales, porque tiene soporte de casi todos los DBMS en uso en la actualidad. Es evidente que las razones de esta amplia aceptación son el tiempo y el esfuerzo dedicados al desarrollo de sus funciones y normas, que hace a SQL muy portátil entre los diferentes productos de RDBMS.

En este capítulo, se emplea Oracle y su esquema de ejemplo HR (Human Resources) para demostrar SQL. Un *esquema* es el conjunto de objetos de una base de datos que pertenecen a un usuario específico (en este caso del usuario HR). Oracle Database 10*g* XE, que requiere cuota por licencia, puede descargarse sin costo de www.oracle.com/technology/software/products/database/index.html. Oracle Database 10*g* XE incluye un esquema de ejemplo de recursos humanos (HR) que se emplea en los ejemplos y en los ejercicios Intente esto de este capítulo. Aprenderá más si prueba las instrucciones de SQL, para que valga la pena el esfuerzo de descargar e instalar el software. En www.oracle.com/pls/xe102/homepage encontrará la documentación específica para 10*g* XE.

NOTA

Debido a que Oracle proporciona 10*g* XE sin cobrar una cuota por licencia, impone restricciones importantes acerca del uso del producto. Si planea utilizarlo para otros fines, además de aprender SQL y probar Oracle, debe leer con cuidado la información de la licencia ofrecida con el producto.

Excepto donde se indica, todos los comandos y funciones presentados en este capítulo cumplen las normas actuales de SQL y, por lo tanto, deben funcionar correctamente en cualquier DBMS que permita SQL. No obstante, sin el esquema HR de Oracle, tendrá que crear tablas de ejemplo como las que proporciona Oracle y llenarlas con datos para ejecutar las instrucciones exactas incluidas en este capítulo. Para mejorar la comprensión, todas las ins-

trucciones de SQL se presentan en mayúsculas. Sin embargo, Oracle no distingue mayúsculas para los comandos de SQL o los nombres de los objetos de la base de datos, de modo que, en sus prácticas, puede escribir los comandos en cualquier combinación de mayúsculas y minúsculas. Pero recuerde que en los datos en Oracle se distinguen las mayúsculas, de modo que cuando escriba un valor de datos que se guardará en la base de datos o si va a buscar datos en la base de datos, debe escribir la forma correcta.

NOTA

Oracle ha publicado Database 11*g*. No obstante, al momento de escribir este libro, la Express Edition (XE) sólo está disponible en la versión Database 10*g*, y ésta es la versión disponible en la página Web para descargas del software Database de Oracle. Si se ofrece una versión más reciente de XE, la encontrará en la página Web, y es muy probable que funcione bien para seguir los ejemplos y los ejercicios Pruebe esto de este libro; aunque, por supuesto, la interfaz de usuario puede ser diferente.

Como se mencionó en el capítulo anterior, SQL es un lenguaje basado en comandos. Las instrucciones de SQL se forman en cláusulas que utilizan *palabras clave* y *parámetros*. Las palabras clave utilizadas suelen ser palabras reservadas para el DBMS, lo que significa que no pueden utilizarse para asignar nombre a los objetos de la base de datos. Por lo general, las cláusulas deben aparecer en una secuencia prescrita. Las instrucciones de SQL deben terminar con un punto y coma (;). Al programa que emplee para conectarse a la base de datos e interactuar con ella se le denomina *cliente de SQL*. Oracle ofrece otros clientes, entre ellos SQL*Plus, iSQL*Plus y SQL Developer, pero en este capítulo se emplea Oracle Application Express porque viene con Oracle 10*g* XE y, por lo tanto, está preparado para usarse en cuanto se instala este último.

Algunos clientes de SQL no ejecutan una instrucción SQL a menos que termine con un punto y coma o una diagonal (la diagonal es una extensión de Oracle para el estándar). Pero casi ninguno de los clientes GUI o basados en Web como Oracle Application Express requieren un carácter de terminación, porque se hace clic en un botón o un ícono para indicar al cliente que está preparado para ejecutar una instrucción. Más allá de las restricciones mencionadas, SQL tiene forma libre, en donde uno o más espacios separan los elementos de lenguaje, y se permiten cambios de línea entre cualesquiera dos elementos (pero no a la mitad de un elemento).

Las instrucciones de SQL se dividen en las categorías siguientes:

- **Lenguaje de consulta de datos (DQL, Data Query Language)** Instrucciones que consultan la base de datos pero no modifican un dato u objeto de ella. Esta categoría contiene la instrucción SELECT. No todos los vendedores incluyen esta diferencia; muchos incorporan DQL en DML, como se explica a continuación.
- **Lenguaje de manipulación de datos (DML, Data Manipulation Language)** Instrucciones que modifican los datos guardados en los objetos de la base de datos (es decir, las tablas). Esta categoría contiene las instrucciones INSERT, UPDATE y DELETE.

- **Lenguaje de definición de datos (DDL, Data Definition Language)** Instrucciones que crean y modifican objetos de la base de datos. Mientras que DML y DQL funcionan con los datos dentro de los objetos de la base de datos, DDL trabaja con los propios objetos. En otras palabras, DDL administra los *contenedores* de los datos, mientras que DML administra los datos *dentro* de los contenedores. Esta categoría incluye las instrucciones CREATE, ALTER y DROP.
- **Lenguaje de control de datos (DCL, Data Control Language)** Instrucciones que administren los privilegios que tienen los usuarios en relación con la base de datos y los objetos guardados en ella. Esta categoría incluye las instrucciones GRANT y REVOKE.

Instrucciones representativas en cada una de estas categorías se presentan en las secciones que siguen. Pero primero echemos un vistazo a la historia del lenguaje.

Breve historia de SQL

El precursor de SQL, llamado QUEL, surgió en las especificaciones para System/R, la base de datos relacional experimental de IBM, a fines de la década de 1970. Sin embargo, otros dos productos, con nombres diferentes para sus lenguajes de consulta, vencieron a IBM en el mercado con los primeros productos de bases de datos relacionales comerciales: Relational Software, de Oracle, y Relational Technology, de Ingres. IBM publicó SQL/DS en 1982, con el lenguaje de consulta llamado lenguaje de consultas estructuradas en inglés (Structured English Query Language, SEQUEL). No obstante, cuando IBM se enteró que SEQUEL era una marca registrada de Hawker Siddeley Aircraft Company, del Reino Unido, cambió el nombre a SQL.

En 1986, ANSI (American National Standards Institute) y, en 1987 ISO (International Organization for Standardization) formaron comités para las normas de SQL. Dos años después, se publicó la primera especificación de normas, conocida como SQL-89. Tres años más tarde, la norma se expandió para convertirse en SQL-92, que contenía casi 600 páginas. La tercera generación, publicada en 1999, fue llamada SQL-99, o SQL3. Se publicaron revisiones adicionales en 2003 (SQL:2003) y 2006 (SQL:2006), y todavía se trabaja en las normas de SQL. Las revisiones publicadas en 1999 y posteriores incorporan muchas de las funciones de objetos requeridas para que SQL funcione en una base de datos de objetos-relacional, al igual que extensiones del lenguaje para hacer a SQL computacionalmente completo (al incluir estructuras de bucle, ramificación y construcciones en mayúsculas) y funciones adicionales como el lenguaje extensible de marcado (Extensible Markup Language, XML). La mayor parte de los productos RDBMS actuales cumplen con la norma, en un grado u otro.

Casi todos los vendedores han agregado extensiones al SQL, en parte porque querían diferenciar sus productos y en parte por exigencias del mercado que los llevaron a implemen-

tar las funciones antes de que existieran las normas. Un ejemplo es el soporte a los tipos de datos DATE y TIMESTAMP. Las fechas son muy importantes en el procesamiento de datos de negocios, pero los desarrolladores de los productos RDBMS originales eran científicos y académicos, no especialistas en computación de negocios, de modo que no se previó esa necesidad. Como resultado, los primeros dialectos de SQL no tenían soporte especial para fechas. Conforme surgieron productos comerciales, los vendedores respondieron a presiones de sus clientes más importantes y se apresuraron a incluir soporte de fechas. Por desgracia, esto hizo que cada uno lo hiciera a su modo. Cuando traslade instrucciones de SQL de una marca a otra, tenga cuidado con las diferencias en los dialectos. SQL es muy compatible y portátil entre diferentes productos, pero los sistemas de base de datos completos rara vez se pueden mover sin ciertos ajustes.

Introducción a Oracle SQL

Como ya se mencionó, Oracle ofrece varias herramientas diferentes de cliente (clientes de SQL) para administrar la formación y ejecución de instrucciones de SQL y la presentación de los resultados. Se denominan herramientas de *cliente* porque suelen funcionar en la estación de trabajo del usuario de la base de datos y pueden conectarse de manera remota a bases de datos que funcionan en otros sistemas de cómputo, que suelen ser servidores compartidos. Es frecuente que las herramientas de cliente también se instalen en el servidor, junto con la base de datos, para facilitar la administración, lo que permite al DBA registrado en el servidor consultar la base de datos sin necesidad de una estación de trabajo de cliente. No obstante, para las ediciones Personal y Express de Oracle, la propia base de datos, junto con las herramientas de clientes, se instala en una estación de trabajo o dispositivo portátil del usuario individual.

Los ejemplos y los ejercicios Pruebe esto del capítulo se concentran en Oracle. Sin embargo, si emplea un RDBMS diferente, el vendedor del producto también ofrecerá herramientas de cliente. Por ejemplo, Microsoft SQL Server ofrece una herramienta GUI (SQL Server Management Studio) y una herramienta de línea de comandos (OSQL). Casi todos los productos DBMS comerciales tienen ediciones exprés (simplificadas) que se instalan y usan sin adquirir una licencia, y también existen productos de código abierto, como MySQL y PostgreSQL (un derivado de Ingres). No obstante, como ya se mencionó, si emplea un producto diferente, tendrá que crear por cuenta propia tablas que se parezcan al esquema de ejemplo HR de Oracle y llenarlas con datos.

Una vez que haya instalado Oracle 10*g* XE, puede iniciar Application Express al seleccionar Inicio | Todos los programas | Base de Datos Oracle 10g Express Edition | Ir a Página Inicial de Base de Datos. Se iniciará su navegador Web predeterminado (suele ser Microsoft Internet Explorer, pero puede ser otro, como Mozilla Firefox) y se mostrará una página de conexión como la que se presenta en la figura 4-1.

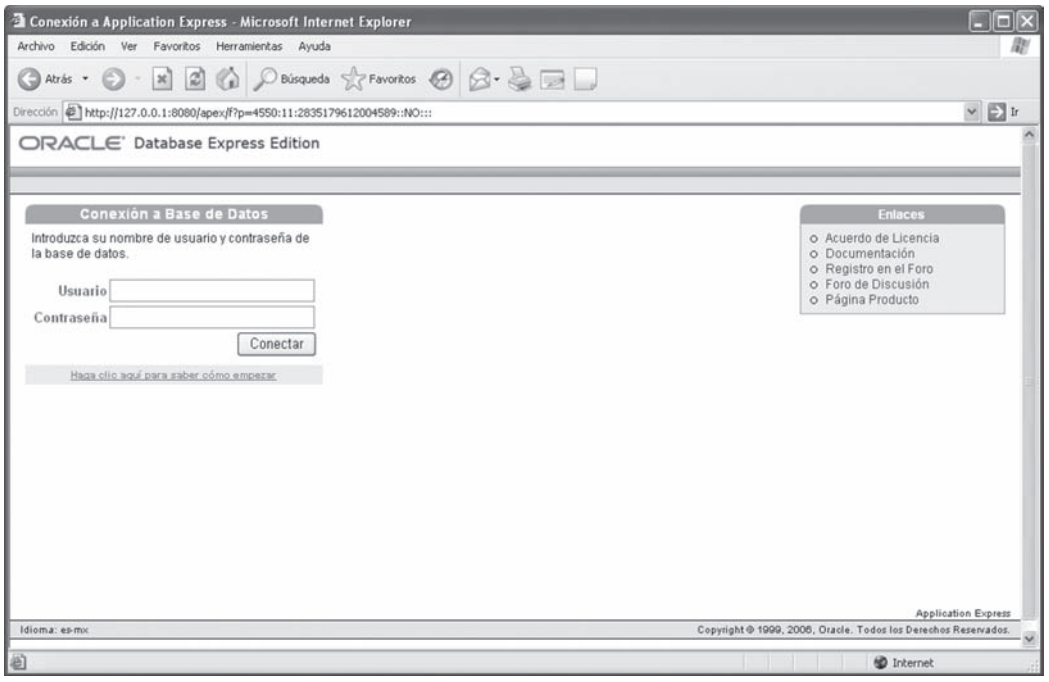


Figura 4-1 La página Conexión a Application Express de Oracle.

Pruebe esto 4-1 Desbloqueo de la cuenta HR y conexión como HR

La cuenta del usuario HR está bloqueada de manera predeterminada, de modo que necesitará registrarse como SYSTEM (la cuenta maestra de una base de datos de Oracle) y desbloquearla antes de utilizarla. Necesitará la contraseña que proporcionó cuando instaló Oracle 10*g* XE para registrarse como SYSTEM. Recuerde que aunque en Oracle las cuentas del usuario no diferencian mayúsculas y minúsculas, las contraseñas sí lo hacen. En este ejercicio desbloqueará la cuenta HR, le asignará una contraseña y luego iniciará sesión como el usuario HR.

Paso a paso

1. Si todavía no tiene en ejecución Oracle Application Express, actívelo desde su menú Inicio.
2. Escriba **SYSTEM** en el campo Usuario, en la página de conexión.

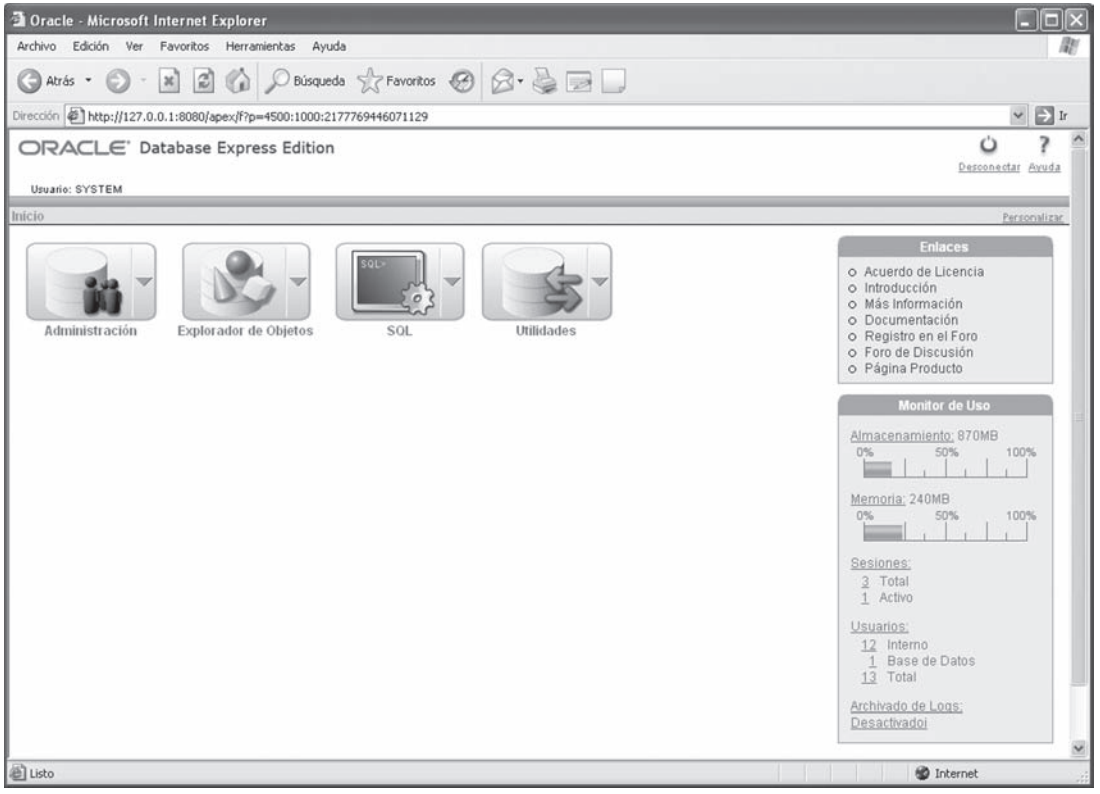


Figura 4-2 La página principal de Application Express.

3. En el campo Contraseña, escriba la contraseña que eligió para la cuenta SYSTEM cuando instaló 10*g* XE.
4. Haga clic en el botón Conectar para abrir la página principal de Application Express, presentada en la figura 4-2. Las opciones disponibles se explican más adelante en esta sección.
5. Haga clic en la flecha junto al ícono Administración y elija Usuarios de Base Datos. A continuación se desplegará la lista de cuentas actuales de usuarios de la base de datos, y es muy probable que el único listado sea HR.
6. Haga clic en el ícono del usuario HR para mostrar la página Gestionar Usuario de Base de Datos para esa cuenta, igual que en la figura 4-3.

(*continúa*)

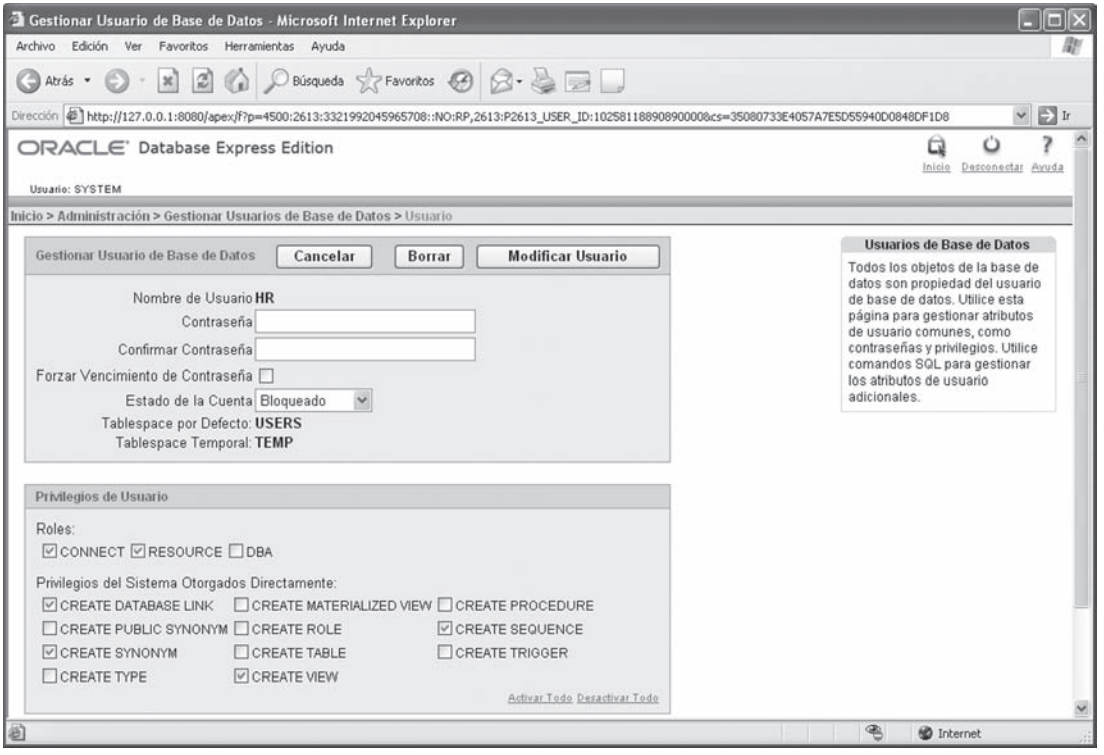


Figura 4-3 La página Gestionar Usuario de Base de Datos.

7. Seleccione una contraseña para la cuenta HR y escríbala en los campos Contraseña y Confirmar contraseña.
8. En el menú desplegable del campo Estado de cuenta, seleccione Desbloqueado.
9. Haga clic en el botón Modificar Usuario. Regresará a la página anterior (la lista de cuentas de usuarios) con un mensaje de confirmación mostrado en un cuadro gris.
10. Haga clic en el vínculo Desconectar, cerca de la esquina superior derecha de la página. Se mostrará una página que confirma la desconexión.
11. Haga clic en el vínculo Conectar, para regresar a la página Conexión que se mostró en la figura 4-1.
12. Escriba en los campos adecuados el nombre de usuario (HR) y la contraseña que eligió y haga clic en el botón Conectar. Se volverá a mostrar la página principal de Application Express, tal como se mostró en la figura 4-2.

Resumen de Pruebe esto

En este ejercicio Pruebe esto desbloqueó la cuenta del usuario HR incluida en la base de datos 10*g* XE de Oracle, le asignó una contraseña y después se conectó como usuario HR. Ahora está preparado para emplear el esquema de muestra HR para explorar SQL.

En la página principal de Application Express (figura 4-2), debe observar cuatro (o tal vez cinco) íconos; cada uno de ellos proporciona un grupo de funciones. He aquí un breve compendio de cada opción:

- **Administración** Esta opción proporciona herramientas para administrar el DBMS (incluidas las especificaciones de almacenamiento y memoria), gestionar las cuentas de usuario (como lo hizo al desbloquear la cuenta HR) y vigilar la base de datos con el fin de identificar y resolver problemas de rendimiento.
- **Explorador de Objetos** Esta opción contiene herramientas para crear y examinar los objetos de la base de datos (tablas, vistas, índices, etc.). El Explorador de Objetos se emplea en el tema siguiente de este capítulo para analizar los objetos dentro del esquema HR.
- **SQL** La opción SQL ofrece tres herramientas de SQL: Comandos SQL, que se emplea mucho en este capítulo para modificar y consignar opciones de SQL y observar los resultados; Archivos de Comandos SQL, que le permite modificar, guardar y ejecutar archivos que contienen varios comandos de SQL; y Generador de consultas, una herramienta gráfica de consulta cuyo concepto es similar a la herramienta Consulta, de Access, que se analizó con detalle en el capítulo 3.
- **Utilidades** Esta opción aporta herramientas para mover datos entre tablas y archivos externos a la base de datos, generar opciones de DDL para los objetos existentes en la base de datos, preparar diversos informes y administrar la papelera de reciclaje que 10*g* XE proporciona para los objetos descartados de la base de datos.
- **Creador de Aplicaciones** Esta opción no se muestra en la figura 4-2 porque no está disponible para todos los usuarios. Sin embargo, se ha habilitado para el usuario HR, de modo que es probable que la vea en la página principal. Creador de Aplicaciones ofrece herramientas para desarrollar y administrar aplicaciones basadas en Web que empleen la base de datos. Como esta opción se refiere a la programación de una aplicación y no a la administración de una base de datos, está más allá del alcance de este libro.

De todas las herramientas contenidas en 10*g* XE, la herramienta Comandos SQL se emplea casi de manera exclusiva en este capítulo. De la página principal, seleccione la flecha junto al ícono SQL y después elija Comandos SQL para abrir la página que se expone en la figura 4-4. El uso de la página es muy sencillo. Usted escribe comandos SQL en el área vacía

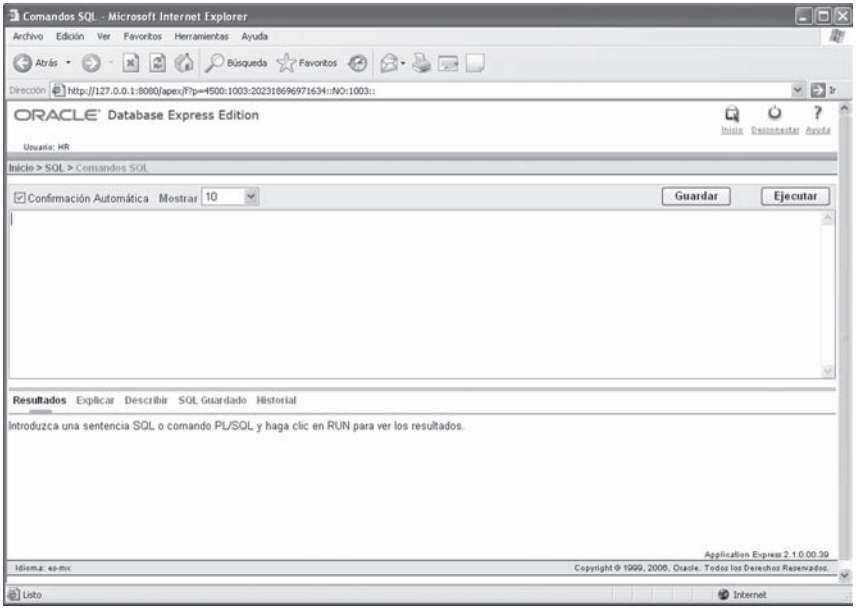


Figura 4-4 La página Comandos SQL de Oracle 10*g* XE.

de la mitad superior de la página, hace clic en el botón Ejecutar y los resultados de la consulta se muestran en la mitad inferior de la página.

Sin embargo, debe estar consciente de las demás opciones de la página:

- *La casilla de verificación Confirmación Automática* (arriba del área donde se introducen los comandos SQL) determina si los cambios en la base de datos se consignan automáticamente o no. La confirmación automática se cubre con mayor detalle en la sección "Soporte de transacciones (COMMIT y ROLLBACK)" más adelante, en el capítulo.
- *La especificación Mostrar* (junto a la casilla Consignación Automática) determina la cantidad máxima de filas que aparecerán en los grupos de resultados en la mitad inferior de la página. Lo predeterminado son 10, lo que significa que sólo verá las primeras 10 filas de resultados de cualquier comando de SQL que ejecute. Necesitará establecer este valor más alto para algunos de los ejemplos presentados en el capítulo.
- *El botón Guardar* le permite guardar una consulta, y asignarle un nombre para poder localizarla más tarde, cuando quiera reutilizarla.
- *La opción Explicar* ofrece una explicación de la manera en que el motor de SQL ejecutará la consulta. Se analiza con mayor detalle en la sección "Afinación del desempeño" del capítulo 11.

- *La opción Describir* indica cómo usar el comando **DESCRIBE** de Oracle para ver la definición de los objetos guardados en la base de datos. En la sección siguiente se analiza el comando **DESCRIBE**.
- *La opción SQL Guardado* le permite encontrar y recuperar los comandos SQL que conservó mediante el botón Guardar.
- *La opción Historial* aporta una lista de las instrucciones de SQL que ha ejecutado antes, y le permite seleccionar una de ellas para reutilización.

¿Dónde están los datos?

Aunque este capítulo se centra en SQL, no puede escribir instrucciones SQL sin comprender al menos de manera básica las tablas que contienen los datos que desea consultar. Al utilizar una herramienta de consulta gráfica, como el recurso Consulta, de Access o Creador de Aplicaciones, de Oracle 10*g* XE, conforme crea una consulta se representan de manera gráfica los objetos de la base de datos y sus definiciones. Sin embargo, cuando utiliza comandos SQL y necesita consultar las definiciones de un objeto de la base de datos, debe hacerlo con una herramienta separada. Los métodos básicos para conseguir esto consisten en emplear las *vistas de catálogo* (vistas especiales ofrecidas por un RDBMS que presentan los metadatos que documentan el contenido de la base de datos), o una herramienta gráfica especialmente diseñada para presentar los metadatos de la base de datos. En las secciones que aparecen a continuación se analizan estas opciones.

Localización de los objetos de una base de datos mediante vistas de catálogo

Oracle proporciona un grupo pormenorizado de vistas de catálogo que sirven para mostrar los nombres y las definiciones de todos los objetos disponibles en la base de datos para el usuario. Casi todos los otros RDBMS tienen una opción similar, pero por supuesto los nombres de las vistas son distintos. Al usar una instrucción SELECT para cualquiera de estas vistas, se muestra información sobre los objetos de su base de datos. Para conocer información completa de las vistas de catálogo disponibles, consulte las referencias de la base de datos Oracle 10g, disponible con un vínculo de www.oracle.com/pls/db102/homepage. Éstas son las vistas más útiles:

- **USER_TABLES** Contiene una fila de información para cada tabla en el esquema del usuario. Esta vista contiene muchas columnas, pero una de las más interesantes, TABLE_NAME, es la primera de la vista. Una vez que conozca los nombres de las tablas, aplique a cada una el comando **DESCRIBE** para revelar más información sobre las definiciones

de las tablas. En la figura 4-5 se presenta un ejemplo de selección de todo en la vista USER_TABLES. Ésta es la instrucción SQL:

```
SELECT *
  FROM USER_TABLES;
```

NOTA

La instrucción SELECT de SQL, mostrada en la figura 4-5, se describe con mayor detalle a lo largo de este capítulo.

- **USER_TAB_COLUMNS** Contiene una fila de información de cada columna de las tablas contenidas en el esquema del usuario actual. Igual que la vista USER_TABLES, contiene muchas columnas, pero las más útiles son TABLE_NAME, COLUMN_NAME, DATA_TYPE, DATA_LENGTH, DATA_PRECISION, DATA_SCALE, NULLABLE y DATA_DEFAULT. Si lee el análisis de los tipos de datos del capítulo 2, el contenido de estas columnas será evidente a partir de sus nombres.

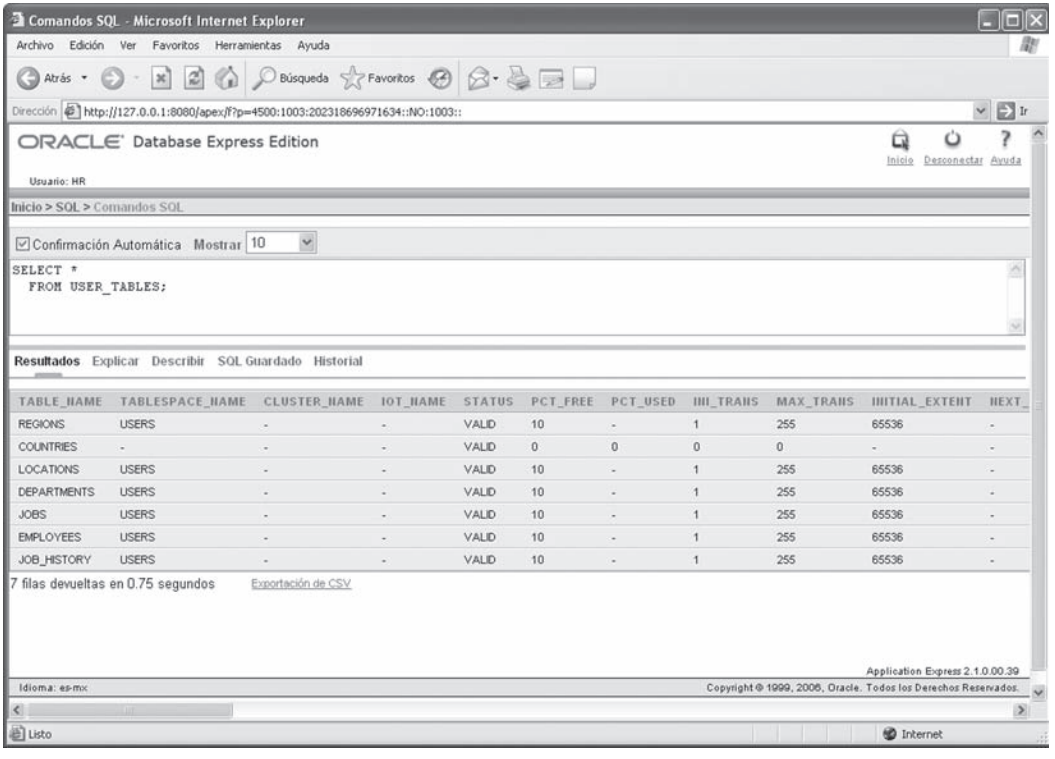


Figura 4-5 Selección de todas las columnas de USER_TABLES.

- **USER_VIEWS** Contiene una fila de información de cada vista en el esquema del usuario, lo que incluye, entre otras cosas, el nombre de la vista y el texto de la instrucción SQL que forma la vista. Observe que no se incluyen las vistas de catálogo porque pertenecen a un esquema diferente llamado SYS.

Como un modo alterno para explorar la vista USER_TAB_COLUMNS con el fin de hallar la definición de una tabla o vista individual, Oracle contiene el comando **DESCRIBE**. La sintaxis es muy sencilla: sólo escriba la palabra clave **DESCRIBE** seguida por el nombre de la tabla o la vista y ejecute el comando. Este comando es particular de Oracle, pero funciona en todos los clientes SQL de Oracle. En la figura 4-6 se presenta el comando **DESCRIBE** aplicado (mediante la página Comandos SQL) a la tabla EMPLOYEES del esquema HR. El comando es muy sencillo:

```
DESCRIBE EMPLOYEES
```

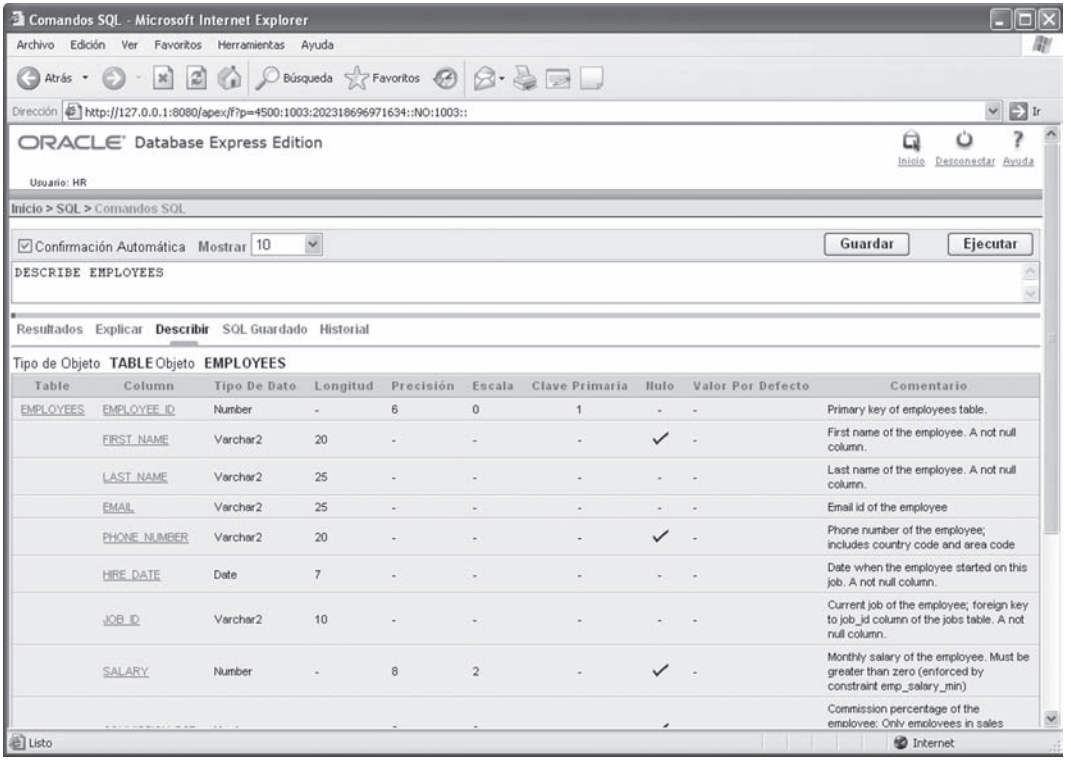


Figura 4-6 El comando DESCRIBE ejecutado para la tabla EMPLOYEES.

Observación de los objetos de una base de datos mediante el Explorador de objetos

Para quienes prefieren no escribir comandos de SQL, Oracle ofrece varias herramientas de GUI, como el Explorador de Objetos, incluido en Application Express de 10*g* XE. Para otras ediciones de Oracle, el producto SQL Developer ofrece opciones similares, y otros vendedores de software también ofrecen distintas herramientas. Casi todos los otros vendedores de RDBMS también ofrecen recursos gráficos, como SQL Server Management Studio de Microsoft.

Pruebe esto 4-2 Uso del Explorador de Objetos de Application Express

En este ejercicio Pruebe esto, empleará el Explorador de Objetos de Application Express.

Paso a paso

1. Si no está ejecutando Application Express, inícielo y conéctese a la cuenta del usuario HR; si ya está en ejecución, haga clic en el vínculo Inicio, cerca de la esquina superior derecha de la página para ir a la página principal.
2. Haga clic en el ícono Explorador de Objetos, en la página principal. También puede emplear la flecha junto al ícono para activar el menú desplegable y seleccionar las opciones Examinar y Tablas. (Examinar las tablas es lo predeterminado cuando hace clic en el ícono.)
3. Las tablas definidas en el esquema actual (HR) se presentan a lo largo del margen izquierdo. Haga clic en la tabla EMPLOYEES. Los metadatos de la tabla son recuperados de las vistas de catálogo mediante el Explorador de objetos y mostrados gráficamente en la parte principal de la página, como se aprecia en la figura 4-7. Observe todas las opciones mostradas sobre los metadatos de la tabla que permiten a un usuario autorizado modificar la definición de la tabla u observar los datos que contiene.
4. Haga clic en el vínculo Inicio (cerca de la esquina superior derecha de la página) para regresar a la página principal.

Resumen de Pruebe esto

En este ejercicio Pruebe esto empleó el Explorador de Objetos, de Application Express, para ver la definición de la tabla EMPLOYEES (la misma tabla para la que se mostró el comando **DESCRIBE** en la figura 4-6). Recuerde que el Explorador de Objetos está diseñado princi-

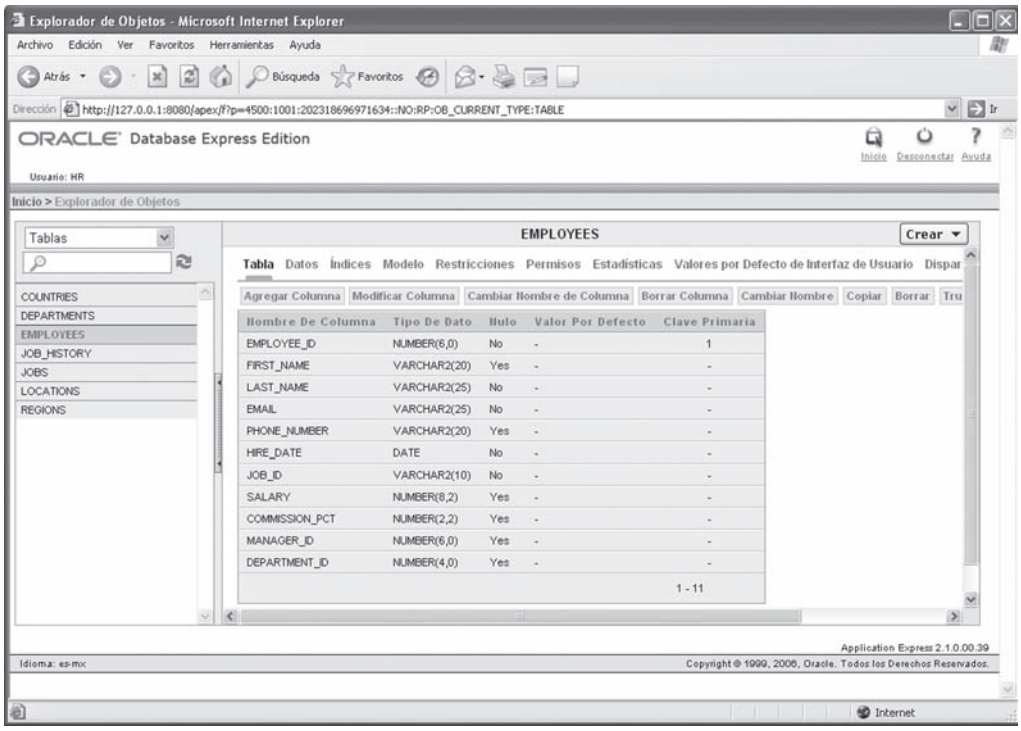


Figura 4-7 El Explorador de Objetos, de Application Express, con los metadatos de la tabla EMPLOYEES exhibidos.

palmente para mostrar y permitir cambios en las definiciones de objetos (los metadatos) y no en los datos reales guardados en las tablas de la base de datos. En la sección siguiente, se analiza en detalle la instrucción SELECT y cómo se emplea para ver los datos guardados en las tablas de la base de datos.

Lenguaje de consulta de datos (DQL): la instrucción SELECT

La instrucción SELECT recupera datos de la base de datos. A continuación se presentan las cláusulas de la instrucción, que se ejemplifican en las secciones siguientes:

- **SELECT** Muestra una lista de las columnas que van a ser devueltas en los resultados.
- **FROM** Presenta una lista de las tablas o vistas de las que se van a seleccionar los datos.

- **WHERE** Ofrece las condiciones para la selección de filas en los resultados.
- **ORDER BY** Especifica el orden en que se van a devolver las filas.
- **GROUP BY** Agrupa las filas para diversas funciones de obtención de totales.

Aunque en SQL se acostumbra escribir las palabras clave en mayúsculas, en casi ninguna implementación es necesario. El intérprete SQL del RDBMS suele reconocer las palabras clave escritas en mayúsculas, minúsculas o ambas. En Oracle SQL, todos los nombres de objetos de la base de datos (tablas, vistas, sinónimos y demás) se pueden escribir de cualquier modo, pero Oracle los cambia automáticamente a mayúsculas durante el procesamiento, porque todos los nombres de objetos de la base de datos de Oracle se guardan en mayúsculas en los metadatos de Oracle, a menos que los nombres estén entre comillas. No obstante, tenga cuidado con otras versiones de SQL. Por ejemplo, Sybase ASE y Microsoft SQL Server pueden establecerse en un modo que reconozca mayúsculas y minúsculas; por ello, los nombres de objetos escritos de otro modo son tratados como objetos *diferentes*. Asimismo, MySQL distingue mayúsculas y minúsculas en las plataformas que funcionan de esa manera, como Unix y Linux. Cuando se distingue mayúsculas y minúsculas, las siguientes serían consideradas tablas *distintas*: EMPLEADOS, Empleados, empleados.

En los temas siguientes se aportan descripciones y ejemplos de modos para usar la instrucción SELECT con el fin de recuperar datos de la base de datos. Esto no pretende ser un examen exhaustivo de las capacidades de la instrucción SELECT, sino más bien un compendio para que se entere de sus extensas capacidades. Todas las figuras utilizadas para ilustrar los ejemplos emplean el esquema de ejemplo HR de Oracle 10*g* XE y la opción Comandos SQL dentro de Application Express. El mejor modo de aprender SQL es probarlo, de modo que lo invito a aplicar estos ejemplos mientras lee.

Listado de todas las filas y columnas

El símbolo de asterisco (*) se utiliza en lugar de una lista de columnas para elegir todas las columnas de una tabla o vista. Ésta es una función útil para presentar con rapidez listas de datos, pero debe evitarse en instrucciones que se reutilizarán, porque cualquier columna nueva se seleccionará automáticamente la próxima vez que se ejecute la instrucción, lo que pone en peligro la independencia lógica de los datos. Observe también que, en la sintaxis de SQL, se hace referencia a tablas, vistas y *sinónimos* (un alias de tabla o vista) de la misma manera. Esto se debe a que los nombres de estos objetos provienen del mismo *espacio de nombre*, lo que significa, por ejemplo, que el nombre de una tabla debe ser único entre todas las tablas, vistas y sinónimos definidos en un esquema específico. En la figura 4-8 se presenta una instrucción SQL que emplea una cláusula SELECT * para mostrar todas las filas y columnas de la tabla EMPLOYEES, junto con una parte de los resultados de la consulta. He aquí la instrucción SQL:

```
SELECT *
  FROM EMPLOYEES;
```

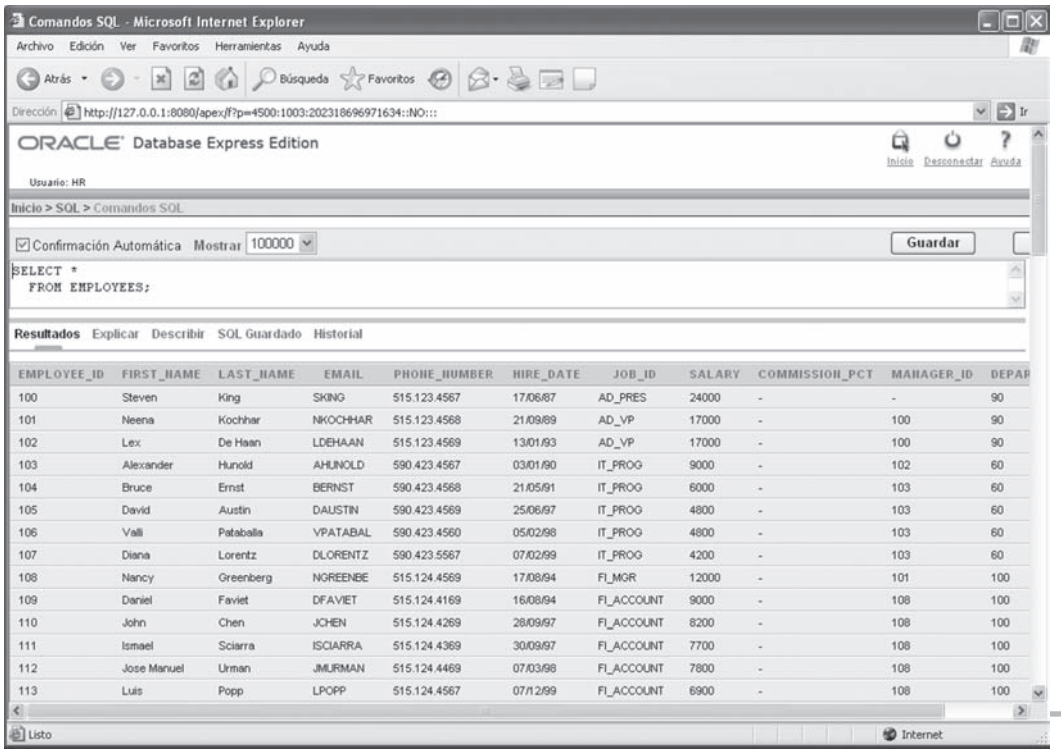


Figura 4-8 Empleo de SELECT * para mostrar todas las filas y columnas de la tabla EMPLOYEES.

Ésta es la forma más sencilla de la instrucción SELECT, y emplea sólo las cláusulas SELECT y FROM. Observe que se modificó la especificación de Mostrar, en la página Comandos SQL, de las 10 filas predeterminadas a un valor mucho más alto (100 000) porque la tabla EMPLOYEES contiene 107 filas y, si se emplea la opción predeterminada, sólo aparecerían en la consulta las primeras 10 filas. Ninguna pantalla de computadora es lo bastante amplia para mostrar un resultado tan grande, de modo que tendrá que emplear la barra de desplazamiento a lo largo del borde derecho de la página para recorrer todas las filas del grupo de resultados.

Delimitación de las columnas que se muestran

Para especificar las columnas que se van a seleccionar, en lugar de un asterisco, incorpore una lista separada con comas después de la palabra clave **SELECT**. Tenga en cuenta que la lista de la cláusula SELECT en realidad proporciona *expresiones* que describen las columnas que se prefieren en los resultados de la consulta, y aunque muchas veces estas expresiones

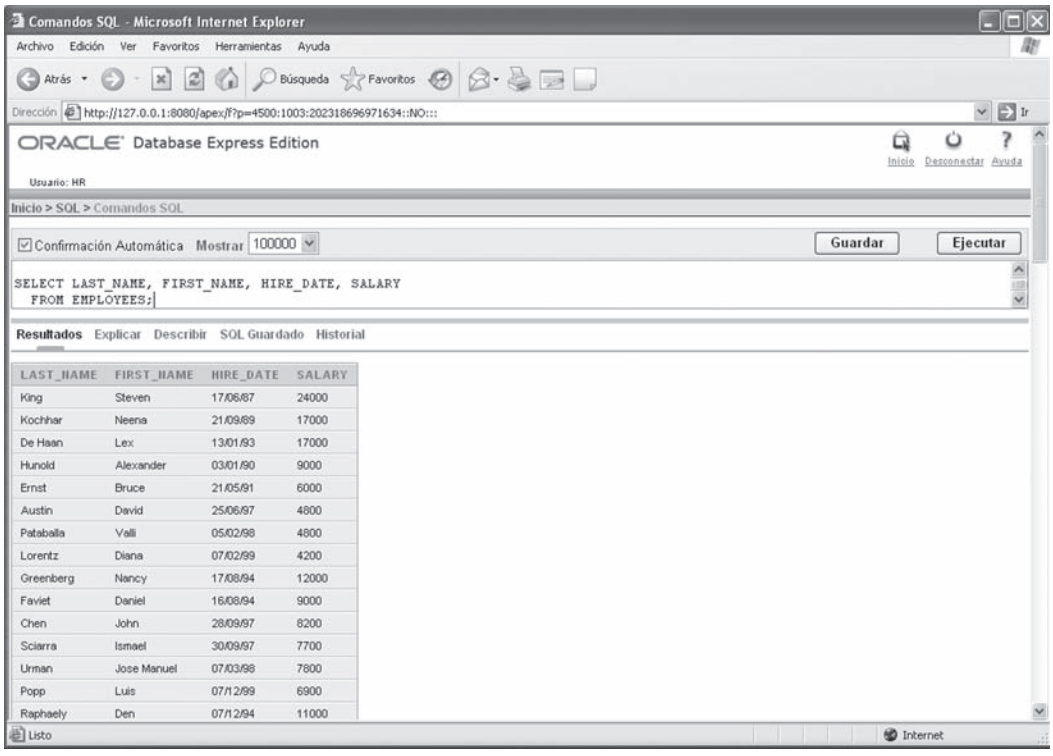


Figura 4-9 Selección de cuatro columnas de la tabla EMPLOYEES.

son sólo nombres de columnas de tablas o vistas, también puede ser cualquier constante o fórmula que SQL pueda interpretar y convertir en valores para la columna. Los ejemplos que aparecen más adelante en el capítulo le indican cómo usar fórmulas y constantes para formar columnas de la consulta. En la figura 4-9 se presenta el SQL para seleccionar las columnas LAST_NAME, FIRST_NAME, HIRE_DATE y SALARY de la tabla EMPLOYEES, junto con resultados parciales de la consulta. Ésta es la instrucción SQL:

```
SELECT LAST_NAME, FIRST_NAME, HIRE_DATE, SALARY
  FROM EMPLOYEES;
```

Ordenamiento de los resultados

Igual que con las consultas de Microsoft Access, con SQL nada garantiza la secuencia de las filas en los resultados, a menos que se especifique en la consulta la secuencia que se prefiere. En SQL, esto se consigue al incorporar una lista separada por comas después de la palabra

clave **ORDER BY**. En la figura 4-10 se muestra el SQL del ejemplo anterior, con la secuencia añadida. Ésta es la instrucción SQL:

```
SELECT LAST_NAME, FIRST_NAME, HIRE_DATE, SALARY
  FROM EMPLOYEES
 ORDER BY LAST_NAME, FIRST_NAME;
```

Observe estos detalles:

- Una secuencia ascendente es lo predeterminado para cada columna, pero se puede agregar la palabra clave **ASC** después del nombre de la columna para desplegar una secuencia ascendente, y **DESC** para una descendente.
- No es necesario que las columnas incluidas en la lista **ORDER BY** aparezcan en los resultados de la consulta (es decir, la lista **SELECT**). Sin embargo, ésta no es la mejor ingeniería humana.

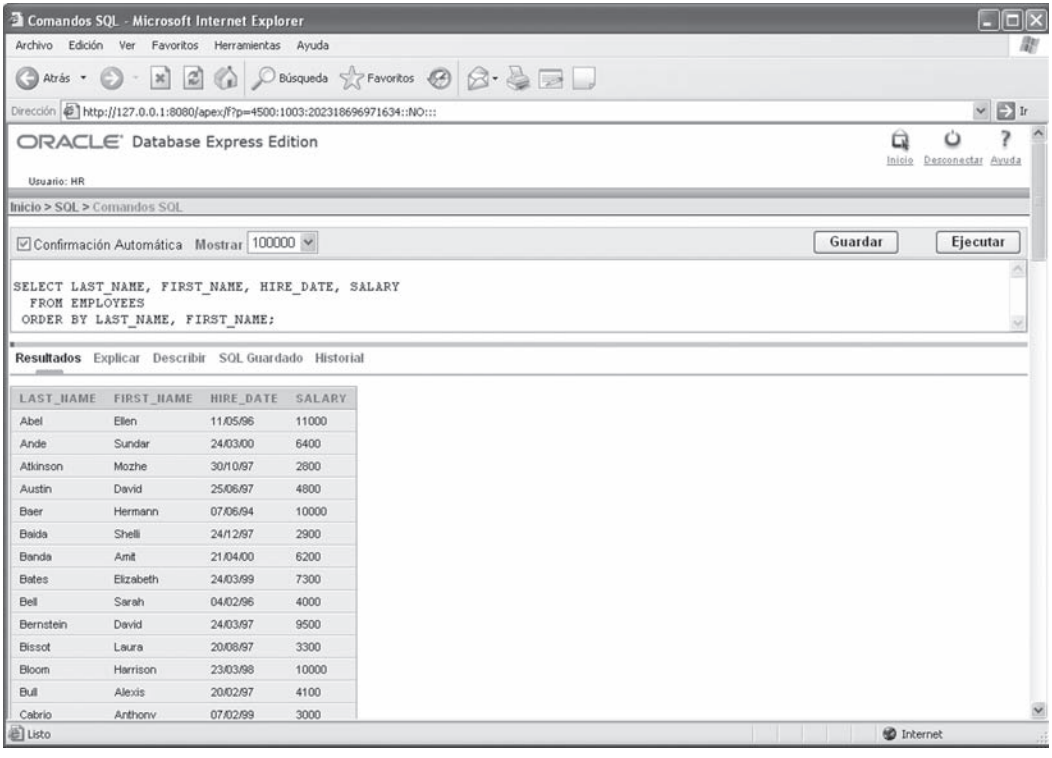


Figura 4-10 Consulta de la tabla EMPLOYEES con la cláusula ORDER BY incorporada.

- En lugar de los nombres de columnas, es posible mostrar la posición relativa de las columnas en los resultados. Sin embargo, el número proporcionado no tiene correlación con la posición de la columna en la tabla o vistas de origen. No se recomienda esta opción en el SQL formal porque alguien que cambia la consulta en una ocasión posterior puede organizar las columnas con base en la lista SELECT y no comprender que, al hacerlo, modifica las columnas utilizadas para ordenar los resultados. En este ejemplo, se puede emplear otra cláusula ORDER BY para obtener los mismos resultados de la consulta: ORDER BY 1,2.

Selección de las filas que se despliegan

SQL emplea la cláusula WHERE para seleccionar las filas que se despliegan. Sin una cláusula WHERE, se muestran todas las filas encontradas en las tablas y vistas de origen. Cuando se incluye una cláusula WHERE, se aplican las reglas del álgebra booleana (que debe su nombre al experto en lógica George Boole) para evaluar la cláusula WHERE de cada fila de datos. Sólo se muestran en los resultados de la consulta las filas para las que la cláusula WHERE se evalúa con un *verdadero* lógico.

Como verá en los ejemplos siguientes, las pruebas de condiciones individuales se deben evaluar como *verdadera* o *falsa*. Los operadores condicionales permitidos son los mismosque se presentaron en el capítulo 3 **(=, <, <=, >, >=** y **<>).** Si se prueban varias condiciones en una sola cláusula WHERE, los resultados de estas condiciones se pueden combinar mediante operadores lógicos como **AND**, **OR** y **NOT**. Por claridad, puede (y debe) incorporar paréntesis en instrucciones complejas y si desea controlar el orden en que se evalúan las condiciones. Se emplea un orden de precedencia muy complicado cuando aparecen varios operadores lógicos en una instrucción. No obstante, es mucho más sencillo recordar que siempre se evalúan primero las condiciones entre paréntesis, y que se deben incluir suficientes paréntesis para que no haya duda del orden en que se evalúan las condiciones.

Una cláusula WHERE sencilla

En la figura 4-11 se presenta una cláusula WHERE sencilla que sólo selecciona las filas en donde SALARY es igual a 11000. La instrucción SQL utilizada es:

```
SELECT LAST_NAME, FIRST_NAME, HIRE_DATE, SALARY
  FROM EMPLOYEES
 WHERE SALARY = 11000
 ORDER BY LAST_NAME, FIRST_NAME;
```

El operador BETWEEN

SQL incluye el operador **BETWEEN** para ayudar a hallar rangos de valores. Los puntos finales se *incluyen* en las filas devueltas. En la figura 4-12 se presenta el uso del operador

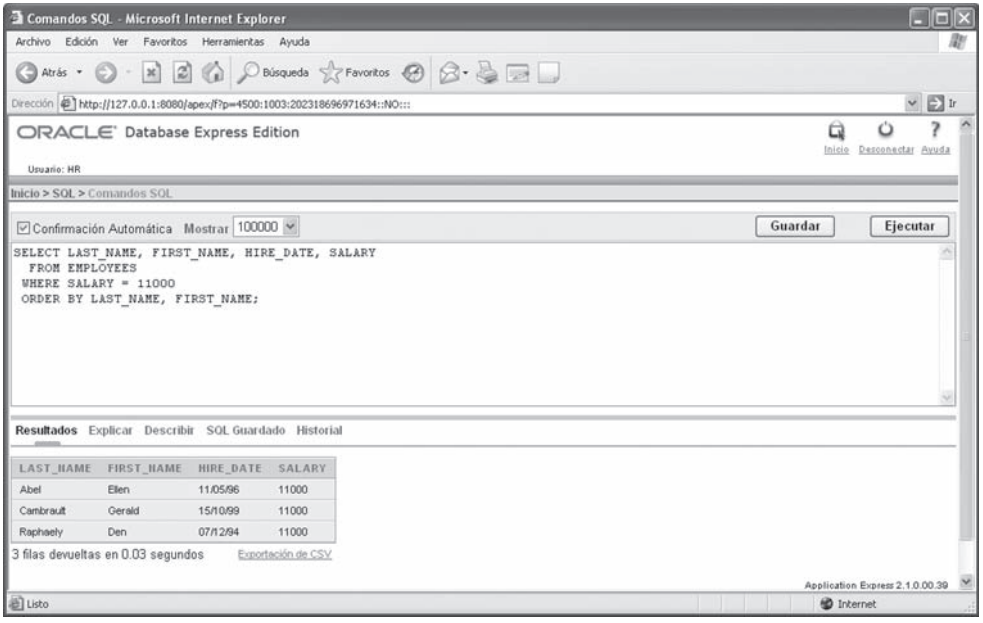


Figura 4-11 SELECT con una cláusula WHERE sencilla.

BETWEEN para encontrar todas las filas donde SALARY es mayor o igual a 10000 y donde es menor o igual a 11000. Ésta es la instrucción SQL:

```
SELECT LAST_NAME, FIRST_NAME, HIRE_DATE, SALARY
  FROM EMPLOYEES
 WHERE SALARY BETWEEN 10000 AND 11000
 ORDER BY LAST_NAME, FIRST_NAME;
```

Éste es un modo alterno de redactar una cláusula WHERE equivalente:

```
WHERE FIRST_NAME LIKE 'Pete%'
  ORDER BY LAST_NAME, FIRST_NAME;
```

El operador LIKE

Para buscar en columnas de caracteres, SQL ofrece el operador **LIKE**, que compara la cadena de caracteres de la columna con un modelo y devuelve un *verdadero* lógico si la columna coincide con el modelo, y *falso* en caso contrario. Se utiliza el carácter de guión bajo (_) como un *comodín* de posición, lo que significa que coincide con cualquier carácter en esa posición de la cadena de caracteres que se evalúa. Se aplica el signo de porcentaje (%) como un comodín que no es de posición, lo que significa que sustituye a cualquier cantidad de caracteres de

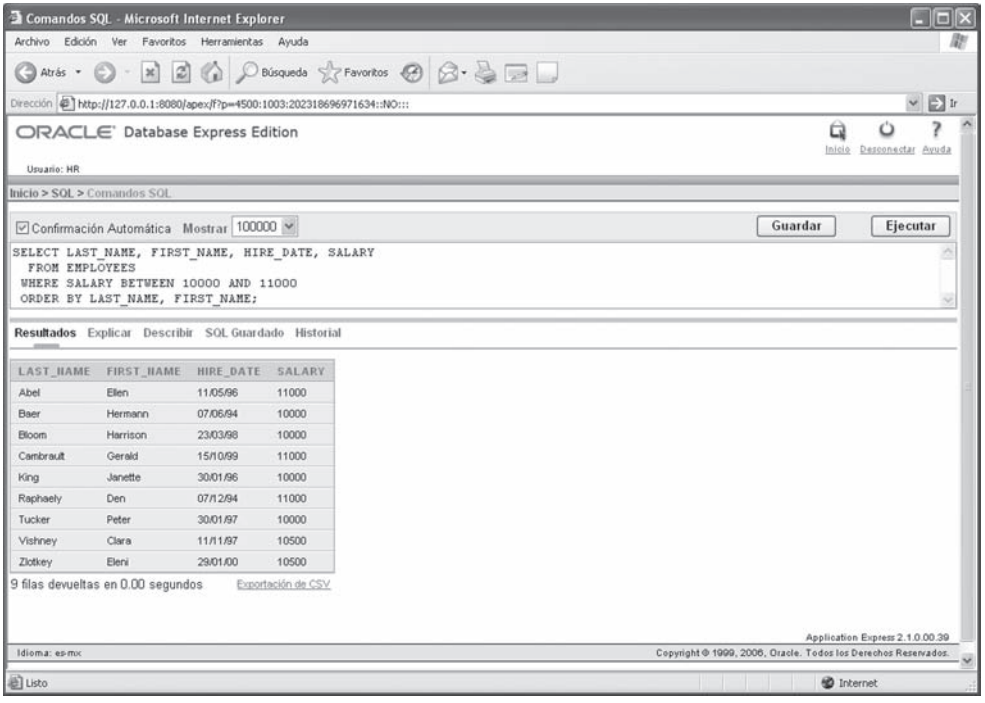


Figura 4-12 SELECT que emplea el operador BETWEEN.

cualquier longitud. Observe que Microsoft Access emplea una característica similar, pero los caracteres comodines son diferentes (coinciden con los de DOS y Visual Basic): el signo de interrogación (?) es el comodín de posición y el asterisco (*) es el comodín que no es de posición. En la tabla siguiente se muestran algunos ejemplos:

Modelo	Interpretación
%hora	Relaciona cualquier cadena de caracteres que termina con *hora*.
Hora%	Relaciona cualquier cadena de caracteres que comienza con *Hora*.
%Hora%	Relaciona cualquier cadena de caracteres que contiene *Hora* (ya sea al inicio, a la mitad o al final).
H_ra	Relaciona cualquier cadena de exactamente cuatro caracteres, en donde el primer carácter es *H* y el tercero y cuarto caracteres son *r* y *a*.
%H_ra%	Relaciona cualquier cadena de caracteres que contenga el carácter *H* seguido por cualquier carácter, que a su vez es seguido por los caracteres *r* y *a* y continúa con cualquier cantidad de caracteres.

En la figura 4-13 exhibe el uso del operador **LIKE** para mostrar sólo las filas donde la columna FIRST_NAME comienza con el texto *Pete*. Ésta es la instrucción SQL:

```
SELECT LAST_NAME, FIRST_NAME, HIRE_DATE, SALARY
  FROM EMPLOYEES
 WHERE FIRST_NAME LIKE 'Pete%'
 ORDER BY LAST_NAME, FIRST_NAME;
```

Condiciones combinadas que emplean OR

Como ya se mencionó, se pueden combinar varias condiciones mediante el operador **OR**. En la figura 4-14 se muestra una cláusula WHERE que selecciona las filas que tienen una columna FIRST_NAME que comienza con *Pete* o una columna SALARY que está entre 10000 y 11000 inclusive. La instrucción SQL es:

```
SELECT LAST_NAME, FIRST_NAME, HIRE_DATE, SALARY
  FROM EMPLOYEES
 WHERE FIRST_NAME LIKE 'Pete%'
    OR SALARY BETWEEN 10000 AND 11000
 ORDER BY LAST_NAME, FIRST_NAME;
```

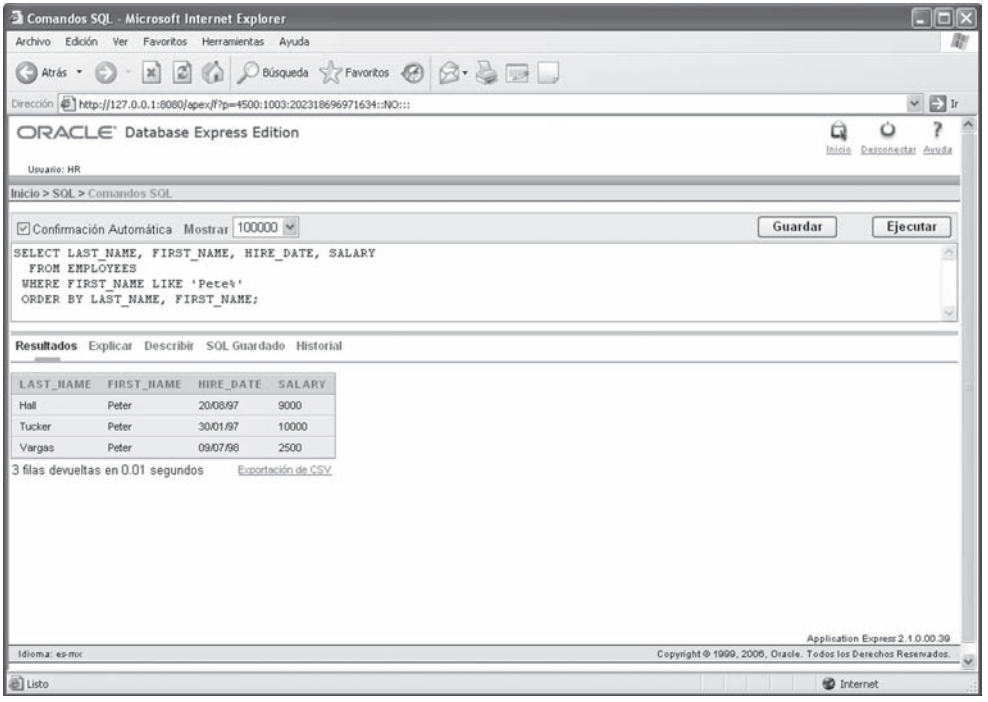


Figura 4-13 SELECT que emplea el operador LIKE.

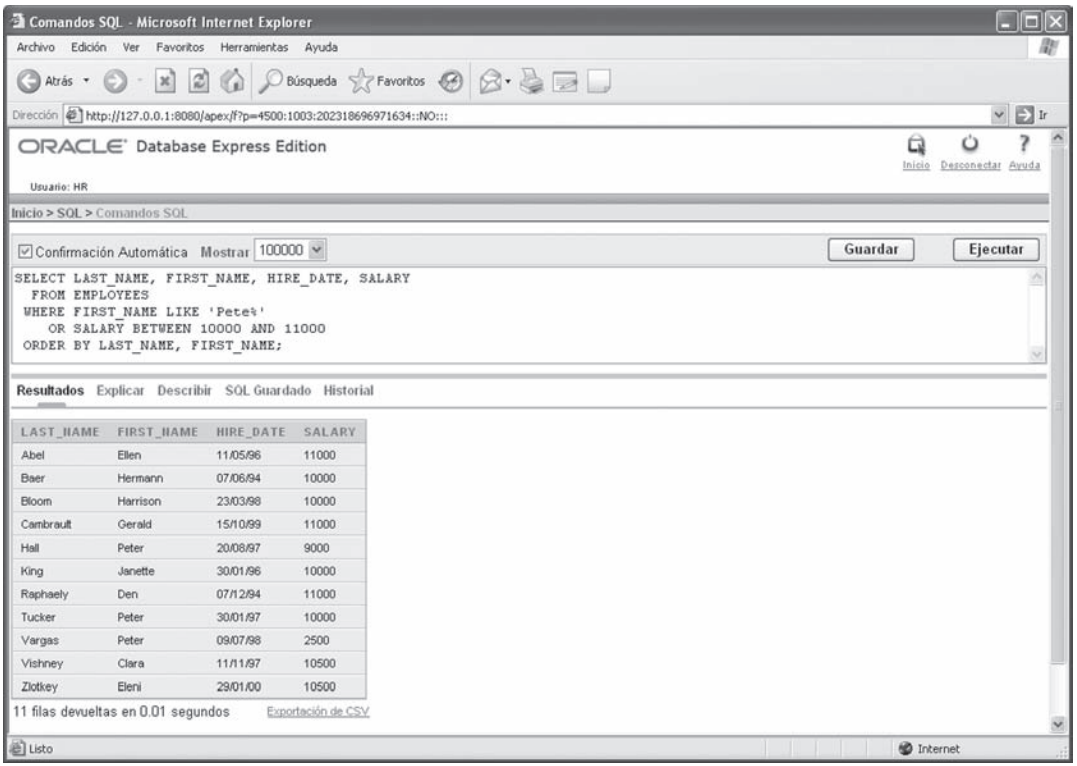


Figura 4-14 SELECT con condiciones combinadas que emplean el operador OR.

Tome en cuenta que, en el ejemplo anterior, se devolvieron tres filas cuando se buscaron nombres que sólo comenzaran con *Pete*. Sin embargo, como en el ejemplo más reciente se empleó el operador **OR**, se obtuvieron no sólo las tres filas que coinciden con *Pete*, sino también ocho filas más que coinciden con el rango de salario incluido, pero no con el criterio *Pete*.

En la figura 4-15 cambia el operador **OR** del ejemplo anterior (figura 4-14) al operador **AND**. Observe que ahora sólo se devolvió una fila, porque ambas condiciones deben ser verdaderas para que aparezca una fila en los resultados de la consulta, y sólo existe una fila con esas condiciones en la tabla. La instrucción SQL es:

```
SELECT LAST_NAME, FIRST_NAME, HIRE_DATE, SALARY
  FROM EMPLOYEES
 WHERE FIRST_NAME LIKE 'Pete%'
   AND SALARY BETWEEN 10000 AND 11000
 ORDER BY LAST_NAME, FIRST_NAME;
```

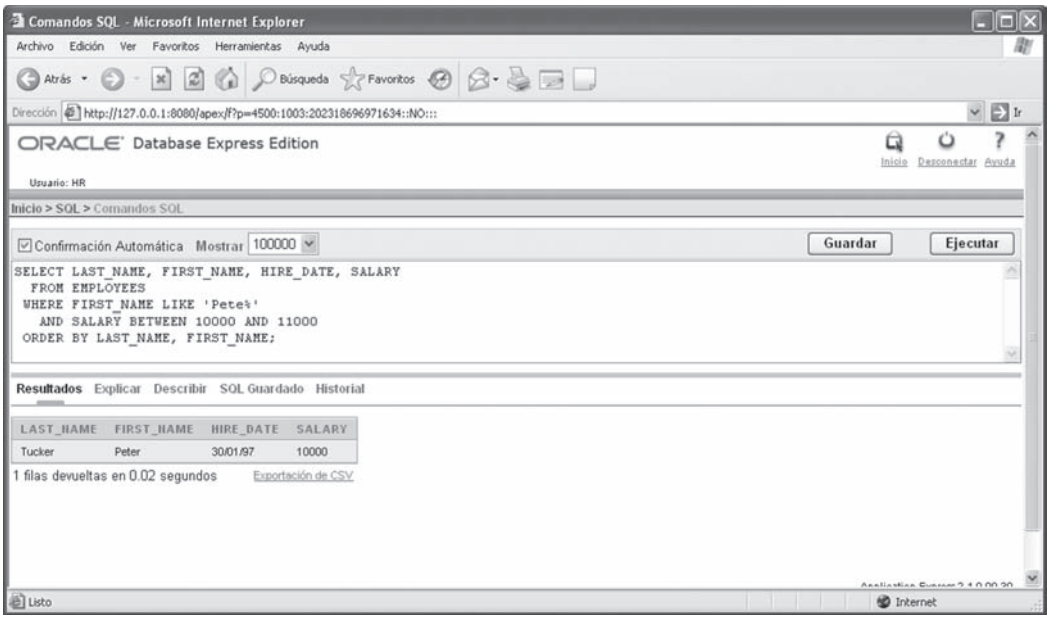


Figura 4-15 SELECT con condiciones combinadas que emplean el operador AND.

La selección secundaria

Una función muy poderosa de SQL es la *selección secundaria* (o *consulta secundaria*); como lo indica su nombre, se refiere a la instrucción SELECT que contiene una instrucción SELECT subordinada. Éste es un modo muy flexible de seleccionar datos.

Supongamos que necesita una lista de todos los empleados que trabajan en ventas. El dilema es que la tabla DEPARTMENTS en el esquema de ejemplo HR contiene varios departamentos de ventas, entre ellos Sales, Government Sales y Retail Sales. Es posible colocar literales para esos tres nombres de departamento en sus identificaciones correspondientes en la cláusula WHERE de la instrucción SELECT. Sin embargo, en este caso el problema que se enfrenta es el mantenimiento de la consulta si después se le agrega o elimina un departamento relacionado con ventas. Un método más seguro consiste en emplear una consulta SQL para hallar las identificaciones aplicables cuando se ejecuta la consulta, y después emplear esa lista de identificaciones para hallar los empleados. La consulta para hallar las identificaciones de un departamento es muy sencilla:

```
SELECT DEPARTMENT_ID
   FROM DEPARTMENTS
 WHERE DEPARTMENT_NAME LIKE '%Sales%'
```

Si se pone la instrucción SELECT anterior en la cláusula WHERE de una consulta que muestra la información de los empleados que interesan, se llega a la consulta presentada en la figura 4-16. Observe que la sintaxis SQL requiere que la selección secundaria esté entre paréntesis:

```
SELECT LAST_NAME, FIRST_NAME, HIRE_DATE, SALARY
  FROM EMPLOYEES
 WHERE DEPARTMENT_ID IN
       (SELECT DEPARTMENT_ID
          FROM DEPARTMENTS
         WHERE DEPARTMENT_NAME LIKE '%Sales%')
 ORDER BY LAST_NAME, FIRST_NAME;
```

Se dice que la instrucción utilizada en este ejemplo contiene una selección secundaria *no correlacionada* porque la SELECT interna (es decir, la que está dentro de la cláusula WHERE) se puede ejecutar primero y los resultados se emplean cuando se ejecuta la SELECT externa. También existe lo que se conoce como selección secundaria (o consulta secundaria)

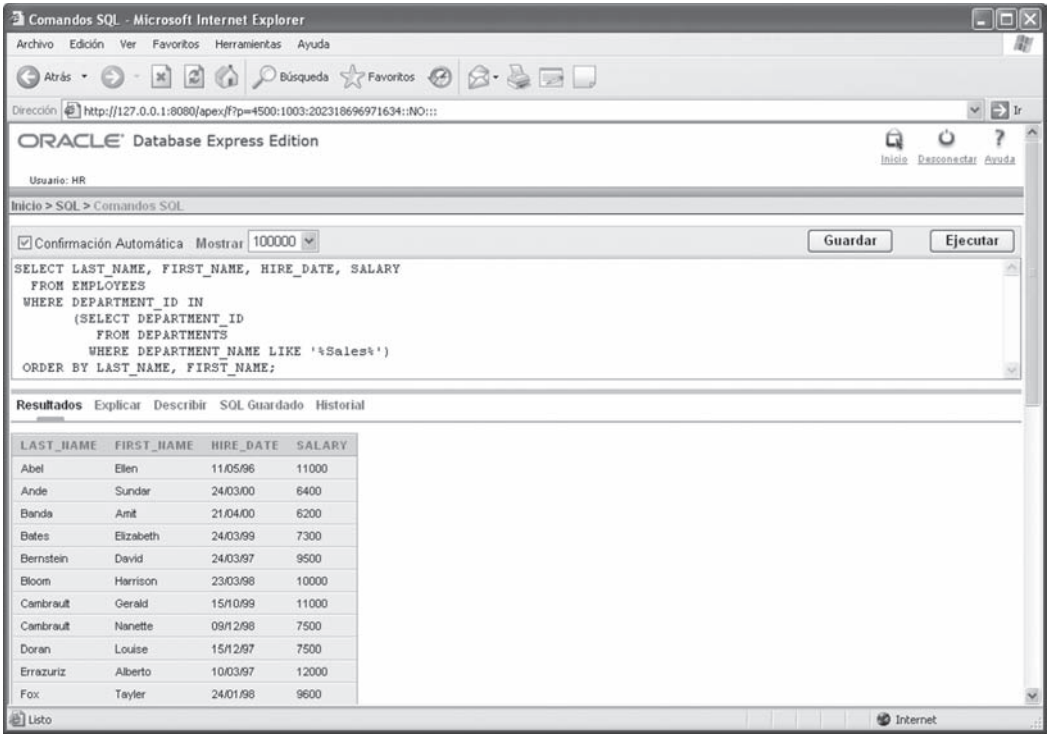


Figura 4-16 SELECT con una selección secundaria.

correlacionada, en que la consulta externa debe invocarse varias veces, una vez para cada fila encontrada en la consulta interna. Considere este ejemplo:

```
SELECT LAST_NAME, FIRST_NAME, SALARY, DEPARTMENT_ID
  FROM EMPLOYEES A
 WHERE SALARY >
        (SELECT AVG(SALARY)
           FROM EMPLOYEES B
          WHERE A.DEPARTMENT_ID = B.DEPARTMENT_ID);
```

Esta consulta encuentra todos los empleados cuyo salario mínimo es superior al promedio de su departamento. La SELECT interna detecta el salario promedio de cada departamento. Después se ejecuta la SELECT externa en cada fila devuelta de la SELECT interna (es decir, para cada departamento) con el fin de hallar a todos los empleados de ese departamento cuyo salario es superior al promedio del área. Es probable que reconozca la función **AVG**, que fue presentada en el capítulo 3, como PROMEDIO. En un ejemplo futuro de SQL se repasará el uso de las funciones de obtención de totales.

Combinación de tablas

Como aprendió en el capítulo 3, necesita unir tablas (o vistas) cuando necesita datos de más de una tabla en los resultados de su consulta. En SQL, usted especifica las combinaciones al incluir las tablas o vistas que se habrán de combinarse en una lista separada con comas en la cláusula FROM de la instrucción SELECT, o mediante la más reciente cláusula JOIN, junto con la cláusula FROM. En esta sección, explorará con detalle esas opciones.

El producto cartesiano

Al especificar combinaciones, es importante indicar al RDBMS cómo relacionar las filas en las tablas (o vistas) que se combinan. No obstante, SQL no le recuerda que lo haga. Si lo olvi-

Pregunta al experto

P: **¿Existen problemas de rendimiento relacionados con selecciones secundarias?**

R: Sí, algunos problemas de rendimiento se pueden asociar con selecciones secundarias. En general, cuantas más filas devuelva la SELECT interna, mayor será el riesgo de un problema de desempeño. Sobre todo, esto ocurre con las selecciones secundarias correlacionadas, porque la SELECT externa debe ejecutarse para cada fila devuelta por la selección secundaria

da, obtendrá un producto cartesiano (que debe su nombre al matemático francés René Descartes), como se aprecia en esta instrucción SQL y la figura 4-17:

```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, DEPARTMENT_NAME
  FROM EMPLOYEES, DEPARTMENTS;
```

Cada vez que escriba una consulta nueva, debe aplicar una prueba de "legitimidad" a los resultados. La consulta SQL de la figura 4-17 parece bien, en la superficie, pero al desplazarse a la parte inferior del grupo de resultados, verá que la consulta devolvió 2889 filas. Si considera que sólo hay 107 empleados, comprende que algo salió muy mal. ¿Cómo puede la consulta producir 2889 filas si sólo combina los empleados y los departamentos? La respuesta: esta consulta no incluyó una especificación de combinación en la cláusula WHERE o en la cláusula JOIN, de modo que el RDBMS generó un producto cartesiano, unió a cada empleado con *cada* departamento, y 27 departamentos por 107 empleados produce 2889 (27 * 107) filas. ¡Perdón!

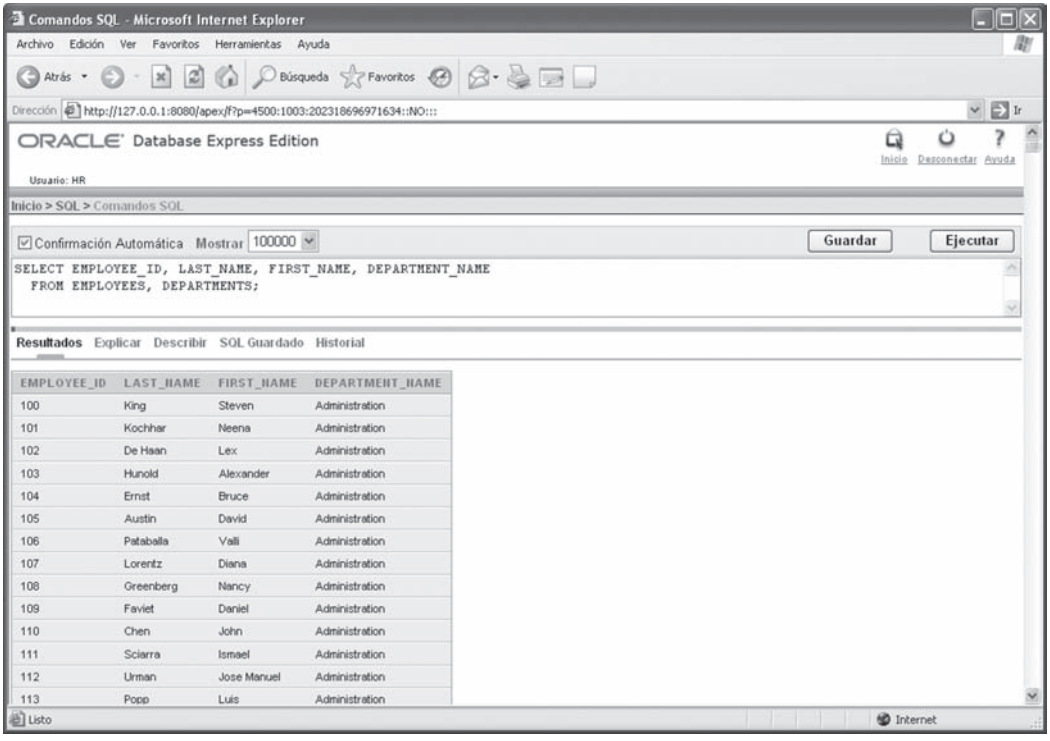


Figura 4-17 Una combinación que genera un producto cartesiano.

La combinación interna de dos tablas

En la figura 4-18 se muestra la corrección, que incluye la adición de una cláusula WHERE que indica al DBMS que haga coincidir la columna DEPARTMENT_ID de la tabla EMPLOYEES (la clave externa) con la columna DEPARTMENT_ID de la tabla DEPARTMENTS (la clave principal). Ésta es la instrucción SQL corregida:

```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, DEPARTMENT_NAME
  FROM EMPLOYEES, DEPARTMENTS
 WHERE EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID;
```

Esto produce un resultado mucho más razonable, con 106 filas. Si se desplaza por los resultados, verá que cada empleado sólo se presenta una vez. Observe que falta una fila, porque hay 107 empleados. La razón de esto, y la modificación a la consulta para que se muestren los 107 empleados, se cubren en el tema siguiente, “La combinación externa”.

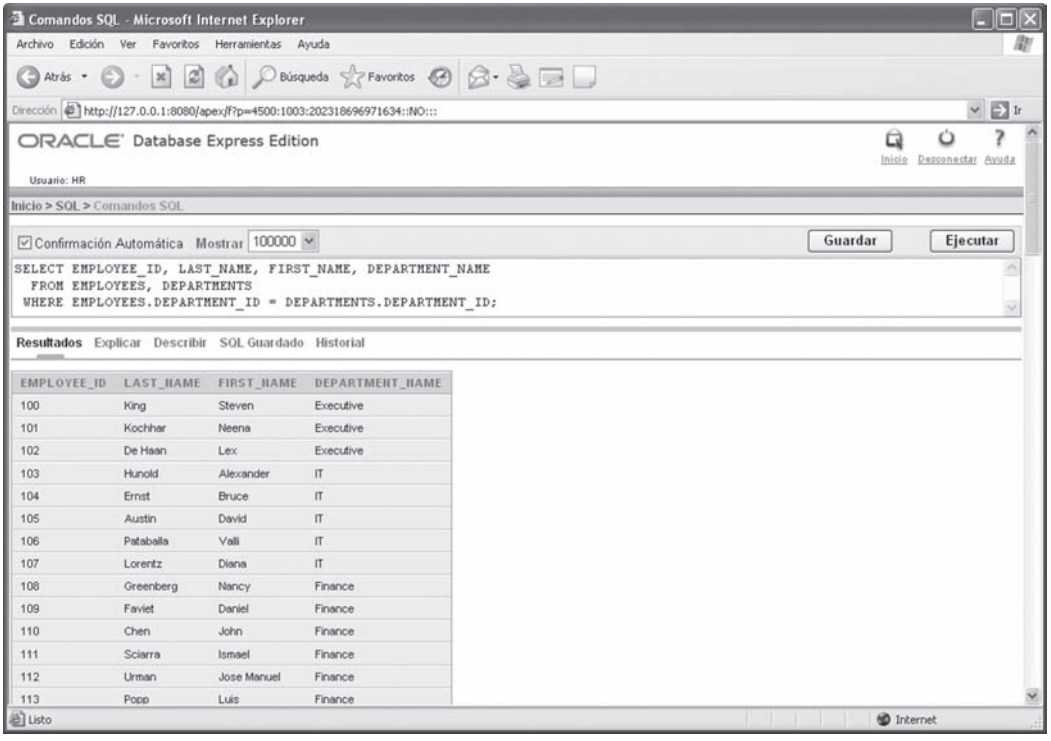


Figura 4-18 Combinación interna de dos tablas mediante la cláusula WHERE.

En SQL, el método original consistía en colocar la condición de combinación en la cláusula WHERE. Sin embargo, se ha agregado una cláusula JOIN a la norma de SQL y ahora es el método preferido para redactar condiciones de combinación. La cláusula JOIN no sólo mejora la legibilidad al separar la condición de combinación de otras combinaciones diseñadas para descartar filas no deseadas del grupo de resultados, sino también es más flexible, como verá en los ejemplos siguientes. En la figura 4-19 se presenta la instrucción SQL de la figura 4-18 reescrita para utilizar la cláusula JOIN. Observe que los resultados de la consulta son exactamente iguales. Ésta es la instrucción modificada de SQL:

```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, DEPARTMENT_NAME
 FROM EMPLOYEES JOIN DEPARTMENTS
   ON EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID;
```

La combinación externa

Las consultas presentadas en las figuras 4-18 y 4-19 devolvieron sólo 106 empleados aunque existen 107 en la tabla EMPLOYEES. Este resultado se debe a que se ejecutó una combinación interna. Sólo se devolvieron filas cuando se encontró una fila de departamento que

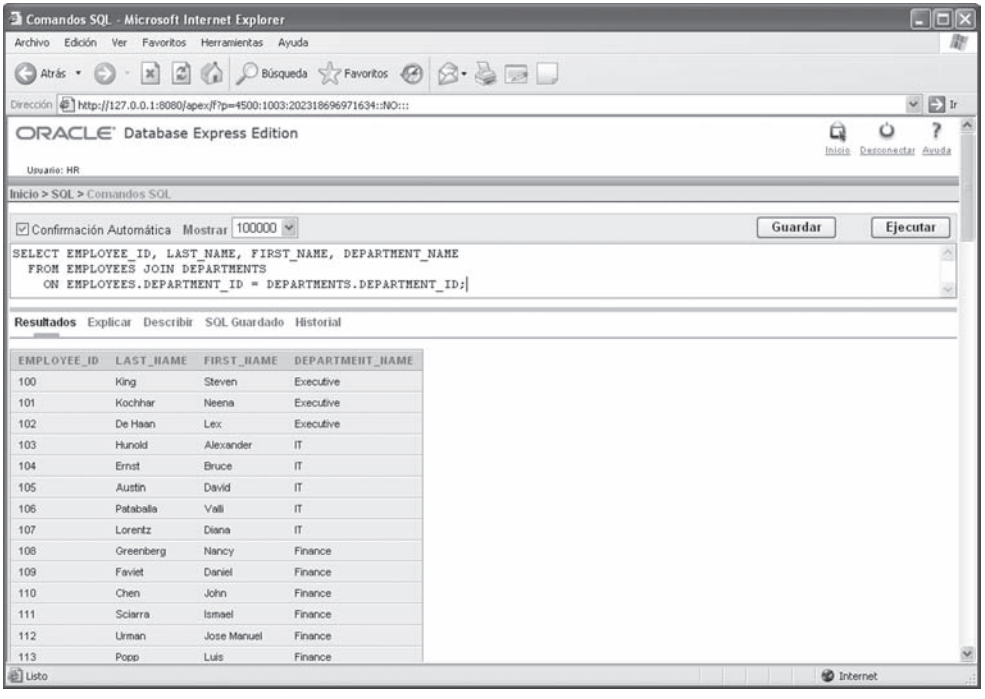


Figura 4-19 Combinación interna de dos tablas mediante la cláusula JOIN.

coincidiera con un empleado; y hay una empleada (Kimberly Grant) que no está asignada a un departamento. Este problema se corrige al cambiar la combinación interna por una combinación externa. Al utilizar una combinación externa, se recuperan todas las filas de la tabla EMPLOYEES, aunque no se encuentre una fila que coincida en la tabla DEPARTMENTS para algunos empleados.

La sintaxis de las combinaciones externas puede resultar un poco confusa, porque existen tres variaciones: combinaciones externas izquierdas, derechas y completas. Sin embargo, la situación se aclara si recuerda que el modificador sólo indica de cuál tabla de la cláusula JOIN se van a devolver todas las filas (sin tomar en cuenta si hay filas que coincidan en la otra tabla). Una combinación externa izquierda (la forma más común) devuelve todas las filas de la tabla nombrada a la izquierda (antes) de la palabra clave **JOIN**; una combinación externa derecha devuelve todas las filas de la tabla nombrada a la derecha (después) de la palabra clave **JOIN**; y una combinación externa completa devuelve todas las filas de ambas tablas. En la figura 4-20 se expone la combinación utilizada en la figura 4-19, modificada a una combina-

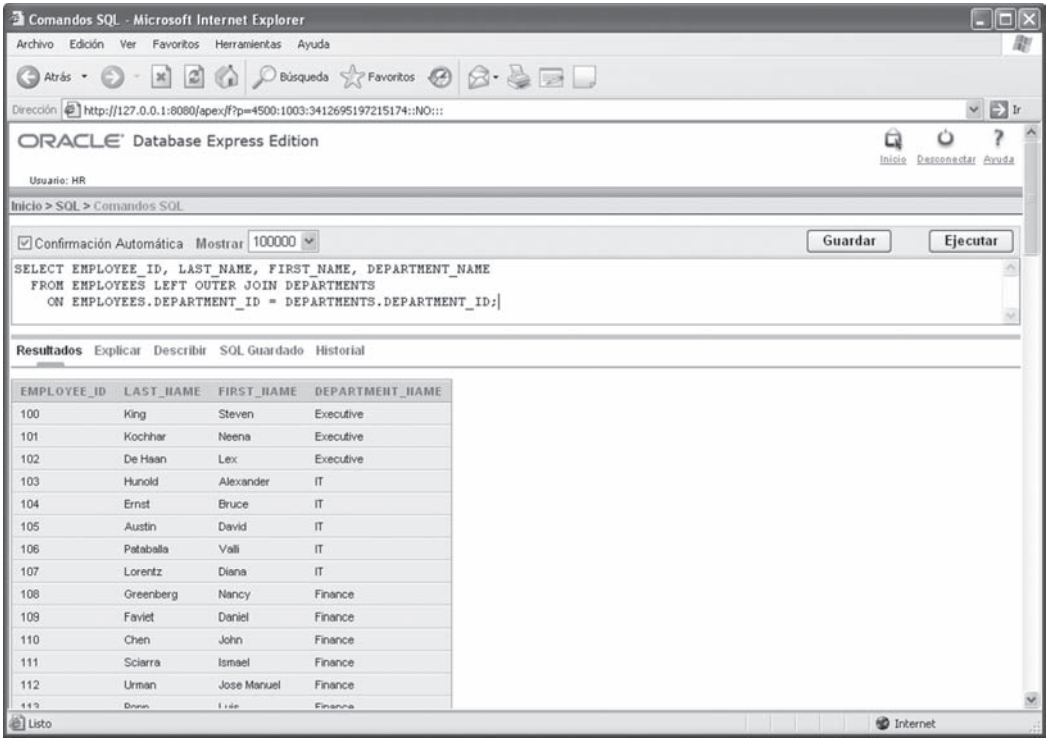


Figura 4-20 Combinación externa izquierda de las tablas EMPLOYEES y DEPARTMENTS.

ción externa izquierda para que se incluyan todos los empleados en los resultados, incluso los que no han sido asignados a departamentos, mediante esta instrucción SQL:

```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, DEPARTMENT_NAME
  FROM EMPLOYEES LEFT OUTER JOIN DEPARTMENTS
    ON EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID;
```

Delimitación de los resultados de una combinación

La cláusula WHERE se utiliza fácilmente para agregar condiciones con el fin de limitar las filas devueltas de una consulta que también incluye combinaciones. En la figura 4-21 se presenta una modificación a la consulta de la figura 4-20, de modo que sólo se devuelven los empleados que trabajan en departamentos de ventas. Éste es el SQL modificado:

```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, DEPARTMENT_NAME
  FROM EMPLOYEES LEFT OUTER JOIN DEPARTMENTS
    ON EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID
 WHERE DEPARTMENT_NAME LIKE '%Sales%';
```

Pregunta al experto

P: He visto Oracle SQL con una combinación externa especificada en la cláusula WHERE y que emplea un signo de más entre paréntesis. ¿Por qué el SQL se escribió de ese modo?

R: Ésta es una sintaxis patentada de Oracle para las combinaciones externas. Igual que casi todos los vendedores, Oracle fue obligada por la demanda del mercado a agregar soporte a combinaciones externas antes de que la sintaxis fuera incluida en la norma de SQL. Oracle incorporó el soporte a la sintaxis OUTER JOIN de la norma de SQL en Oracle 9i versión 2. Antes de esa versión, el único modo para especificar una combinación externa era mediante una sintaxis patentada que requería que se agregara el símbolo (+) a la condición de combinación (en el lado derecho para una combinación externa izquierda y en el lado izquierdo para una combinación externa derecha). Por lo tanto, la combinación externa del ejemplo anterior se escribiría así:

```
SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, DEPARTMENT_NAME
  FROM EMPLOYEES, DEPARTMENTS
 WHERE EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID(+);
```

No sólo ya no debe escribir instrucciones de SQL nuevas con esta sintaxis, porque sólo funciona con Oracle SQL, sino que también debe esforzarse por convertir el SQL existente a la sintaxis OUTER JOIN de la norma porque, con el tiempo, Oracle eliminará el soporte a la sintaxis patentada. Por ejemplo, la sintaxis de combinación externa patentada de SQL Server (que emplea un asterisco a la izquierda o a la derecha del signo de igual en la condición de combinación) fue dejada de lado en SQL Server 2005, lo que provocó muchos problemas a quienes querían convertir a la versión más reciente, a menos que ejecutaran la base de datos nueva en el modo de compatibilidad.

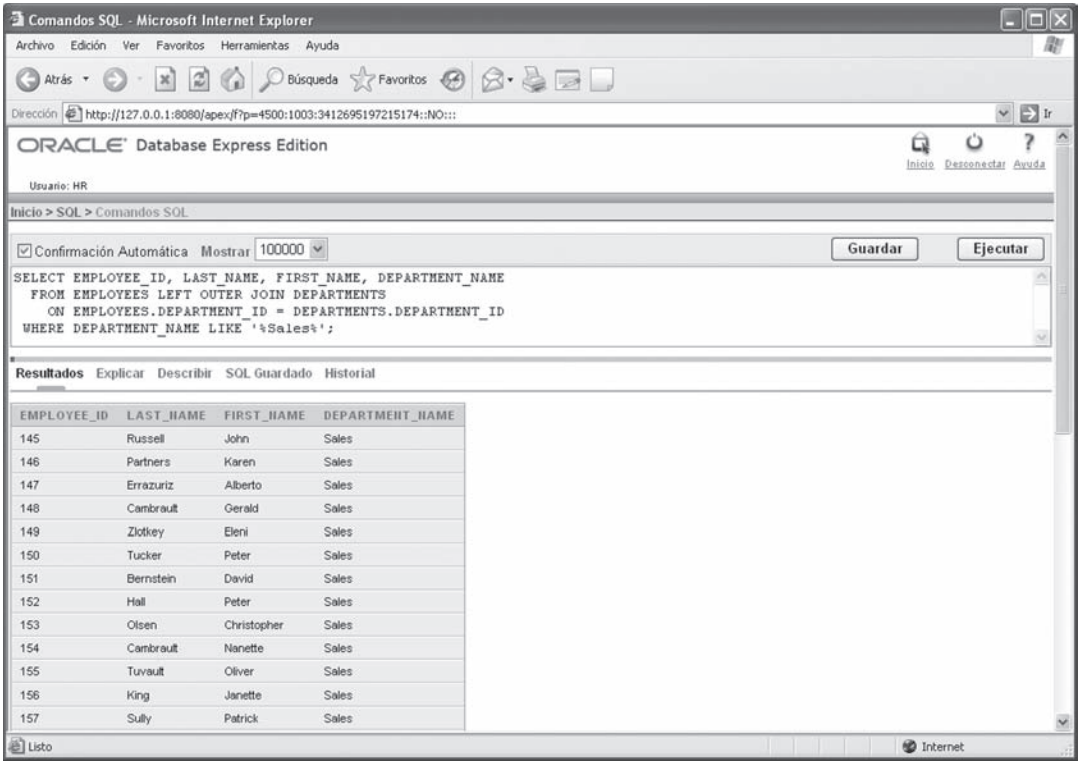


Figura 4-21 Combinación externa con una condición de la cláusula WHERE agregada.

La autocombinación

Cuando una tabla tiene una relación recursiva, necesita combinar la tabla consigo misma para seguir la relación en los resultados de su consulta. La tabla EMPLOYEES posee ese tipo de relación porque la columna MANAGER_ID contiene el valor EMPLOYEE_ID del jefe de cada empleado. En nuestro ejemplo, cada empleado tiene un gerente en la tabla, excepto el propietario de la compañía (Steven King), de modo que la consulta está escrita mediante una combinación externa, igual que en la figura 4-22. Por cierto, es muy común en las relaciones recursivas que algunas filas no tengan elemento primarios; de lo contrario, nunca podría insertar la primera fila de la tabla. Ésta es la instrucción SQL:

```
SELECT A.EMPLOYEE_ID, A.LAST_NAME, A.FIRST_NAME,
       B.FIRST_NAME || ' ' || B.LAST_NAME AS MANAGER_NAME
  FROM EMPLOYEES A LEFT OUTER JOIN EMPLOYEES B
       ON A.MANAGER_ID = B.EMPLOYEE_ID;
```

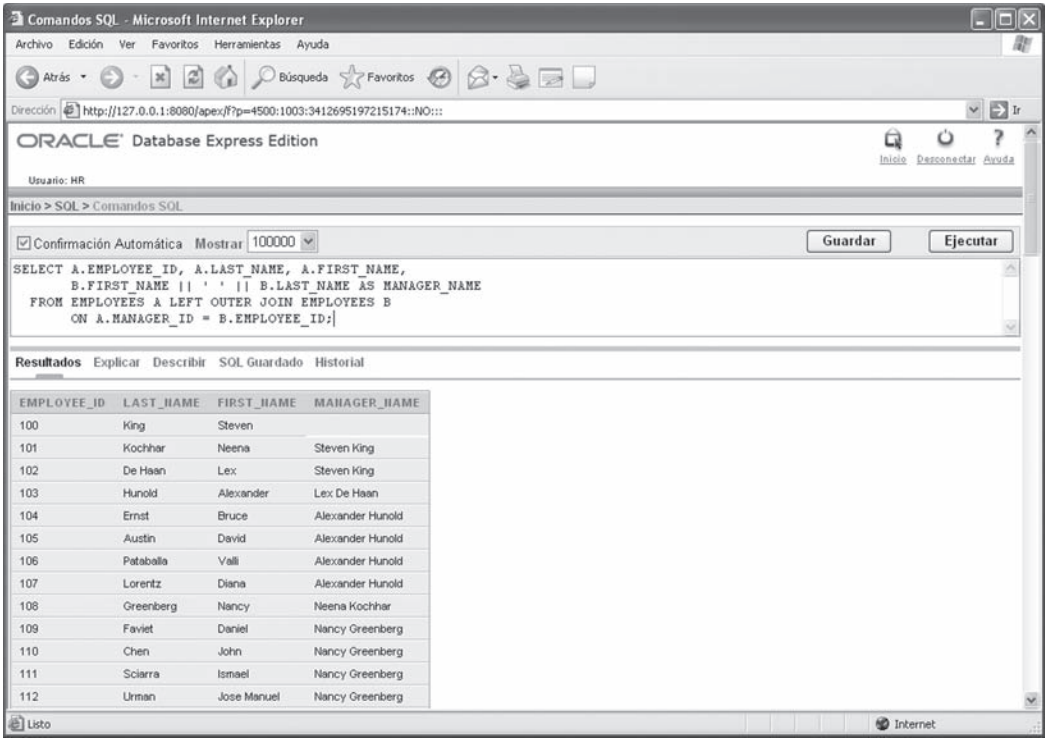


Figura 4-22 SELECT que contiene una autocombinación.

Observe que agregamos otra complicación a este ejemplo al unir el nombre y el apellido del gerente con un espacio entre ellos para formar la columna MANAGER_NAME en los resultados. El operador de unión de la norma SQL es ||; en cambio, SQL Server requiere +. El nombre de columna se asigna mediante la palabra clave **AS** seguida por el nombre deseado. (En realidad **AS** es opcional en Oracle SQL, de modo que puede dejar un espacio después de la expresión de la columna y agregar el nombre de columna deseada. No obstante, es mejor incluirla siempre, porque algunas implementaciones de SQL la requieren.) Si no se asignara de este modo un nombre de columna, el RDBMS tendría que generarlo (cada columna en el grupo de resultados debe tener un nombre de columna válido), de modo que es mejor asignar uno cada vez que se forma una columna en una consulta mediante una expresión, en lugar de un simple nombre de columna.

Funciones de obtención de totales

Como recordará del capítulo 3, las funciones de obtención de totales combinan los valores en varias filas. En esta sección, las explorará con detalle.

Funciones de obtención de totales sencillas

En la figura 4-23 se emplean funciones de obtención de totales para hallar los salarios mínimo, máximo y promedio de todos los empleados, junto con un conteo de la cantidad total de empleados. Ésta es la instrucción SQL utilizada:

```
SELECT MIN(SALARY), MAX(SALARY), AVG(SALARY), COUNT(*)
  FROM EMPLOYEES;
```

Debido a que no se emplea una cláusula GROUP BY para agrupar filas, toda la tabla se considera un grupo, de modo que sólo se devuelve una fila en el grupo de resultados. Es posible que haya observado el valor devuelto por la función **AVG(SALARY)**; el motor de SQL no redondea los resultados a menos que se pida, de modo que al ejemplo siguiente se le incorporó una función **ROUND** para mejorar la legibilidad de los resultados.

Combinación de columnas totalizadas y normales (error)

Si agrega DEPARTMENT_ID a la consulta sin incluir una cláusula GROUP BY, devuelve un mensaje de error, igual que en la figura 4-24 (la función de grupo no es de grupo único), que es muy críptico. Lo que se trata de señalar es que la consulta contiene un solo grupo (la tabla completa) porque no hay una cláusula GROUP BY, y también incluye cuando menos

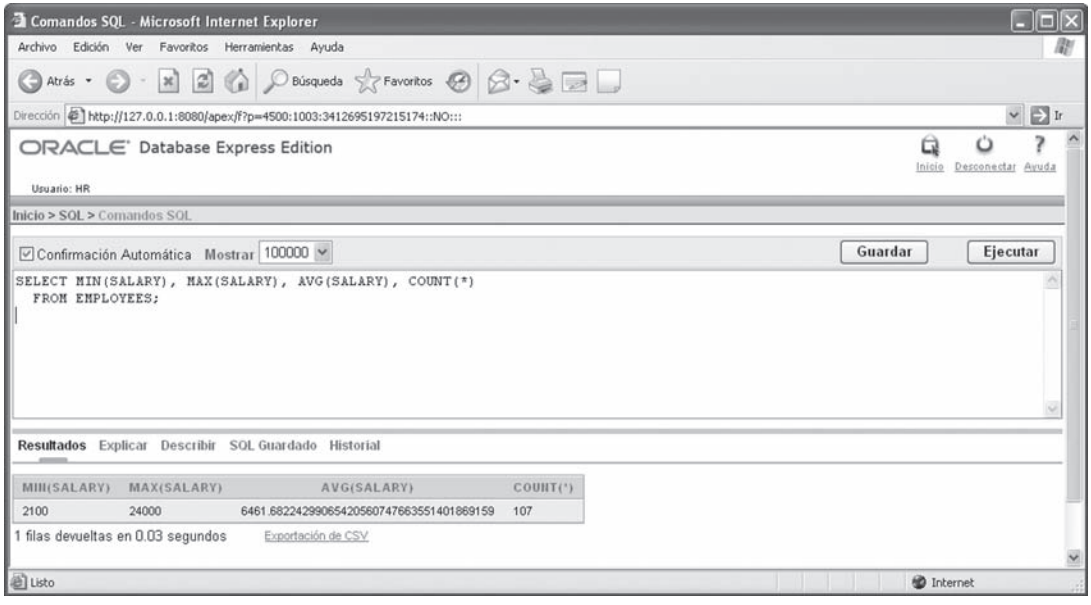


Figura 4-23 SELECT con funciones de obtención de totales sencillas.

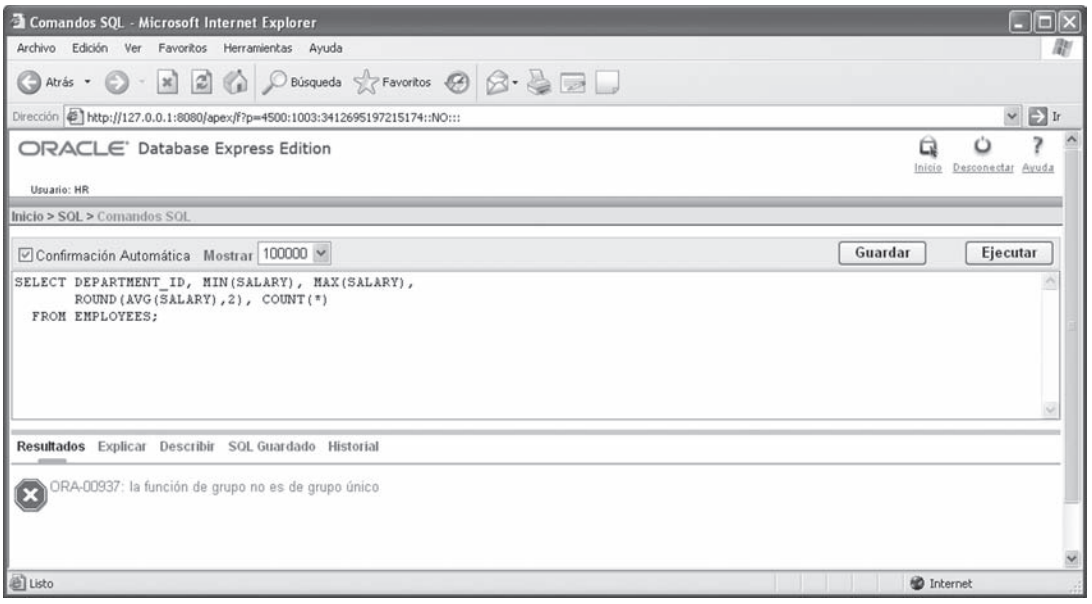


Figura 4-24 Error provocado por combinar funciones de obtención de totales y columna normales sin agrupar.

una expresión de columna que no es una función de grupo. En este caso, señala que DEPARTMENT_ID no es una función de grupo.

Observe que se incluyó una función **ROUND** en la columna AVG(SALARY) para redondear el promedio a dos lugares decimales con el fin de que los resultados sean más legibles que los mostrados en la figura 4-23. La función **ROUND** *no* es una función de obtención de totales; sólo redondea el valor de una sola columna. Sin embargo, es perfectamente aceptable aplicar una función a los resultados de otra función, a lo que se le conoce como *anidar* funciones. No hay límite para las cosas inteligentes que se pueden hacer con SQL:

```
SELECT DEPARTMENT_ID, MIN(SALARY), MAX(SALARY),
       ROUND(AVG(SALARY),2), COUNT(*)
  FROM EMPLOYEES;
```

Funciones de obtención de totales con GROUP BY

La consulta de la figura 4-24 es ilógica porque, en esencia, pide al RDBMS que muestre cada valor de DEPARTMENT_ID pero, al mismo tiempo, sólo muestre una fila que contenga los valores de las otras columnas (formadas con funciones de obtención de totales). Para remediar la situación, se debe indicar al RDBMS que se busca *agrupar* las filas por DEPARTMENT_

ID y, para cada *grupo*, mostrar DEPARTMENT_ID junto con los resultados de las columnas totalizadas (los salarios mínimo, máximo y promedio del departamento y la cuenta de la cantidad de empleados del departamento). La consulta corregida se presenta aquí y en la figura 4-25:

```
SELECT DEPARTMENT_ID, MIN(SALARY), MAX(SALARY),
       ROUND(AVG(SALARY),2), COUNT(*)
  FROM EMPLOYEES
 GROUP BY DEPARTMENT_ID;
```

La cláusula GROUP BY sólo devuelve una fila por departamento, pero esas filas no necesariamente están en una secuencia de Id de departamento; y al analizar la figura 4-25, se aprecia que las filas no están en ninguna secuencia en particular. La lección aquí es que siempre debe incluir GROUP BY cuando quiera que las filas en los resultados de la consulta se devuelvan en una secuencia específica.

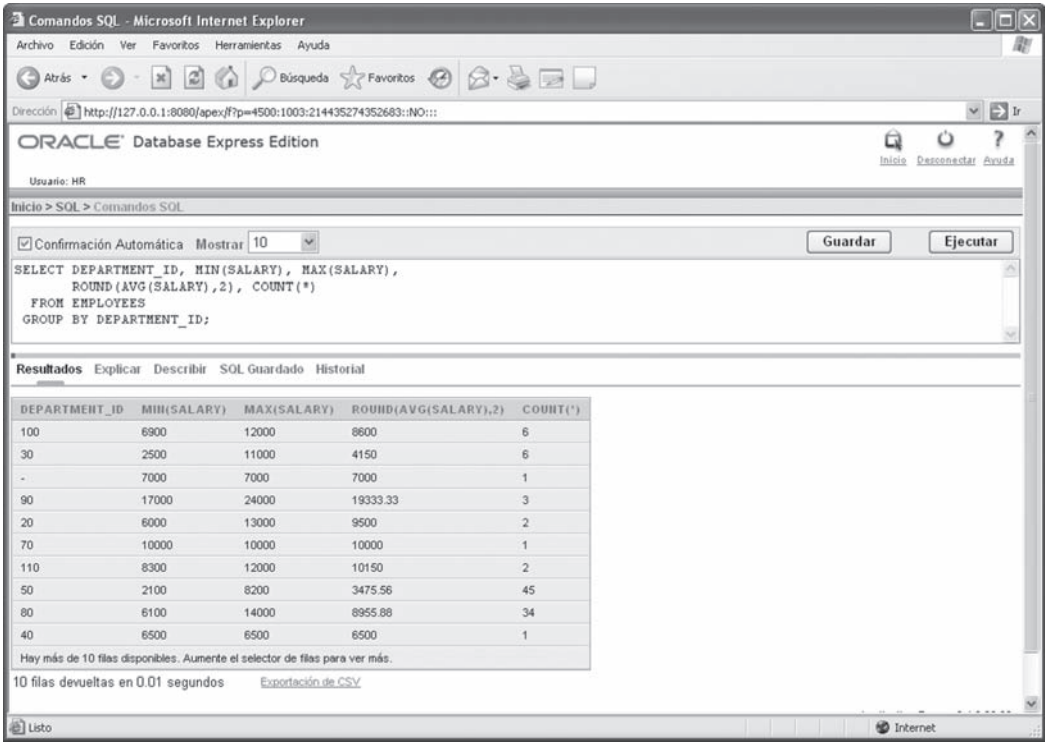


Figura 4-25 SELECT con funciones de obtención de totales y una cláusula GROUP BY.

Pregunta al experto

P: **Recuerdo que la cláusula ORDER BY proporciona una lista de nombres de columnas que se van a usar para aplicar una secuencia a las filas del grupo de resultados de una consulta. ¿Puedo usarla para clasificar las columnas formadas mediante funciones y otras expresiones?**

R: Sí, puede utilizar las columnas del grupo de resultados formadas mediante expresiones en la cláusula ORDER BY. Simplemente repita la expresión completa en la cláusula ORDER BY. Por ejemplo, si quisiera cambiar la consulta presentada en la figura 4-25 para que muestre las filas en una secuencia descendente, a partir del salario máximo en el departamento, lo consigue al agregar esta cláusula ORDER BY:

```
ORDER BY MAX(SALARY) DESC
```

Lenguaje de manipulación de datos (DML)

Los tipos de instrucciones del lenguaje de manipulación de datos (Data Manipulation Language, DML) son INSERT, UPDATE y DELETE. Estos comandos le permiten agregar, cambiar y eliminar filas de datos en las tablas. Antes de analizar cada uno de estos tipos de instrucciones, necesita comprender el concepto de transacciones y cómo las permite el RDBMS.

Soporte a transacciones (COMMIT y ROLLBACK)

En el RDBMS, una *transacción* es una serie de una o más instrucciones SQL tratadas como una sola unidad. Una transacción debe funcionar o fracasar por completo, lo que significa que los cambios que hace una transacción en cualquier base de datos deben volverse permanentes cuando la transacción concluye correctamente. Por otra parte, estos cambios deben eliminarse por completo de la base de datos si la transacción falla antes de su conclusión. Por ejemplo, puede comenzar una transacción con un proceso que crea un pedido nuevo y después, al final del proceso, cuando se ha introducido toda la información del pedido, culminar la transacción. Es importante que los demás usuarios de la base de datos no observen fragmentos de un pedido hasta que se haya introducido y confirmado por completo.

SQL permite transacciones con las instrucciones COMMIT y ROLLBACK. Ocurren algunas variaciones en la sintaxis y manejo de estos comandos entre los diferentes vendedores de RDBMS. Casi ninguno de ellos requiere argumentos con una instrucción COMMIT o ROLLBACK, de modo que la instrucción es simplemente la palabra clave seguida por el signo de punto y coma que concluye cada instrucción SQL.

En Oracle, se inicia una transacción de manera implícita para una sesión de un usuario de la base de datos en cuanto el usuario usa una instrucción que cambia datos (es decir, una instrucción INSERT, UPDATE y DELETE, pero no una instrucción SELECT). En cualquier momento, el usuario puede usar COMMIT, lo que vuelve permanentes todos los cambios concluidos en la base de datos y, por lo tanto, los hace visibles para cualquier usuario. El usuario también puede usar ROLLBACK, que revierte cualquier cambio realizado en la base de datos. Las instrucciones COMMIT y ROLLBACK no sólo concluyen una transacción, sino también preparan la escena para una nueva. Debe recordar otra complicación: en Oracle, ocurre una confirmación *automática* cuando el usuario se desconecta de la base de datos, y antes de cualquier instrucción de DDL (que se cubre más adelante en el capítulo).

Una opción a las transacciones implícitas es el modo *confirmación automática*, que, en esencia, pone cada instrucción SQL en su propia transacción. Cuando está activa la confirmación automática, cualquier instrucción que modifica datos se consigna automáticamente en cuanto se completa con éxito la instrucción. Al principio del capítulo se describió la casilla de verificación Confirmación Automática, en el cliente de Application Express, que activa y desactiva el modo de confirmación automática en una sesión de la base de datos. Otra manera de modificar el modo en Oracle consiste en ejecutar los comandos **SET AUTOCOMMIT ON** y **SET AUTOCOMMIT OFF**. Sin embargo, estos comandos no están permitidos en el cliente Application Express de Oracle, tal vez debido al soporte explícito al utilizar la casilla de verificación Confirmación Automática.

En contraste, en Sybase ASE y Microsoft SQL Server, la confirmación automática es el modo predeterminado. El usuario de la base de datos debe usar una instrucción BEGIN TRANSACTION para iniciar una transacción de manera explícita. Una vez comenzada, los cambios realizados en la base de datos se vuelven permanentes con una instrucción COMMIT TRANSACTION o pueden revertirse mediante una instrucción ROLLBACK TRANSACTION. Algunos RDBMS, como Microsoft Access y MySQL, no proporcionan soporte a transacciones.

La instrucción INSERT

En SQL, la instrucción INSERT se emplea para agregar nuevas filas de datos a las tablas. Una instrucción INSERT también inserta filas mediante una vista, siempre y cuando se cumplan las condiciones siguientes:

- Si la vista combina varias tablas, todas las columnas a las que se hace referencia con la instrucción INSERT deben ser de la misma tabla. Dicho de otro modo, INSERT sólo puede afectar una tabla.
- La vista debe incluir todas las columnas obligatorias de la tabla base. Si en la vista no aparecen las columnas con restricciones NOT NULL, es imposible asignar valores a esas columnas y, por lo tanto, también es imposible usar la vista para realizar una INSERT.

La instrucción INSERT adopta dos formas básicas: una en donde los valores de las columnas son indicados en la instrucción misma, y la otra donde los valores son seleccionados de una tabla o vista mediante una selección secundaria. A continuación se analizan ambas.

INSERT con una cláusula VALUES

La forma de la instrucción INSERT que incluye la cláusula VALUES sólo puede crear una fila cada vez que se ejecuta, porque los valores de esa fila de datos se proporcionan en la propia instrucción. En la figura 4-26 se nuestra un ejemplo que incorpora una fila nueva a la tabla EMPLOYEES. Ésta es la instrucción SQL:

```
INSERT INTO EMPLOYEES
  (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER, HIRE_DATE,
   JOB_ID, SALARY, COMMISSION_PCT, MANAGER_ID, DEPARTMENT_ID)
VALUES (911921, 'Werdna', 'Leppo', 'leppo@whatever.com', null, SYSDATE,
        'IT_PROG', 15000, 0.0, 103, 60);
```

Observe la lista de columnas después de la palabra clave **INSERT**. Esta lista separada por comas es opcional, pero si se proporciona, siempre debe ponerse entre paréntesis. Si omite la lista, los valores de las columnas deben ofrecerse en el orden correcto (es decir, el mismo orden en que están físicamente ordenadas las columnas en la tabla) y no es posible omitir nin-

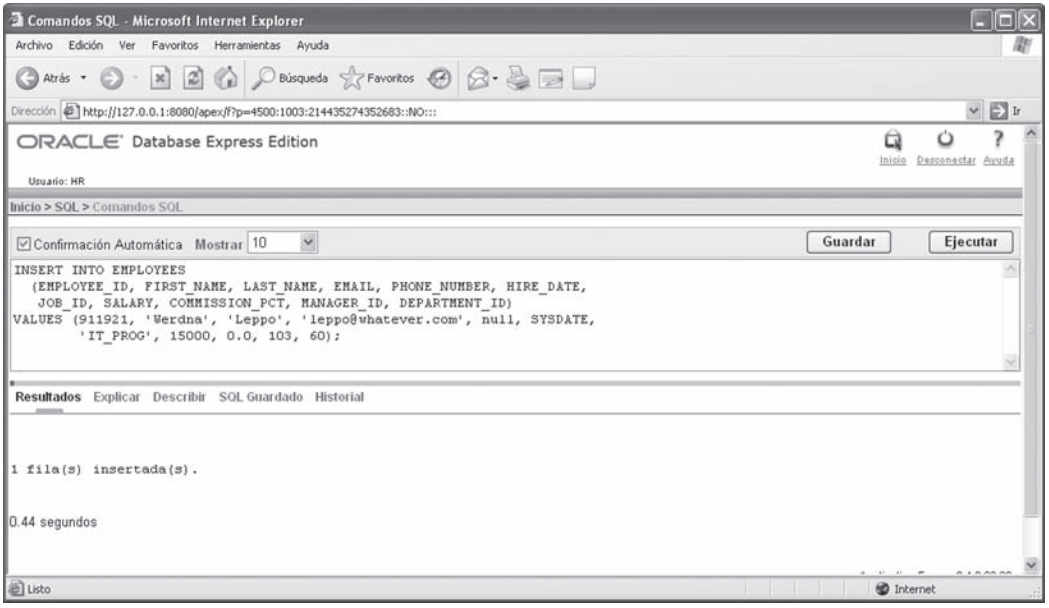


Figura 4-26 INSERT que emplea la cláusula VALUES.

gún valor de columna. La instrucción puede funcionar de manera errónea, si alguien agrega columnas a la tabla, incluso opcionales, de modo que *siempre* es una buena idea proporcionar la lista de columnas, aunque implique más trabajo crear una. Después de la lista de columnas está la palabra clave **VALUES** y luego una lista de los valores de las columnas. Esta lista separada por comas también debe ponerse entre paréntesis. Los elementos de la lista VALUES tienen una correspondencia uno a uno con la lista de columnas (si se proporcionó una) o con las columnas definidas en la tabla o la vista (si no se proporcionó una lista de columnas). Se puede emplear la palabra clave **null** (o **NULL**) para asignar valores nulos a las columnas de la lista. SYSDATE es una pseudocolumna proporcionada en las bases de datos de Oracle que siempre contiene la fecha y la hora actuales.

INSERT con una consulta secundaria

La forma de una instrucción INSERT que incluye una consulta secundaria crea una fila en la tabla de destino para cada fila recuperada de la tabla o vista de origen. Se utiliza una consulta secundaria para recuperar la información que se insertará. En el ejemplo siguiente, se emplean las filas de una tabla imaginaria llamada EMPLOYEE_INPUT para insertar datos en la tabla EMPLOYEES:

```
INSERT INTO EMPLOYEES
    (EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, PHONE_NUMBER,
     HIRE_DATE, JOB_ID)
       SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME,
              EMAIL, PHONE_NUMBER, SYSDATE, JOB_ID
         FROM EMPLOYEE_INPUT;
```

Si quiere probar esta instrucción INSERT, encontrará las utilizadas para crear la tabla EMPLOYEE_INPUT en la sección "Instrucciones del lenguaje de definición de datos (DDL)" un poco más adelante en el capítulo.

La instrucción UPDATE

La instrucción UPDATE en SQL se utiliza para actualizar los valores de datos de las columnas de una tabla (o vista) incluidas en la instrucción. Se puede añadir una cláusula WHERE para limitar el alcance de la instrucción a las filas que coinciden con sus condiciones; de lo contrario, la instrucción trata de actualizar todas las filas de la tabla (o vista) nombrada en la instrucción. En la figura 4-27 se presenta un ejemplo de la instrucción UPDATE que modifica el número telefónico del empleado 921. Ésta es la instrucción SQL:

```
UPDATE EMPLOYEES
   SET PHONE_NUMBER = '301.555.1212'
 WHERE EMPLOYEE_ID = 921;
```

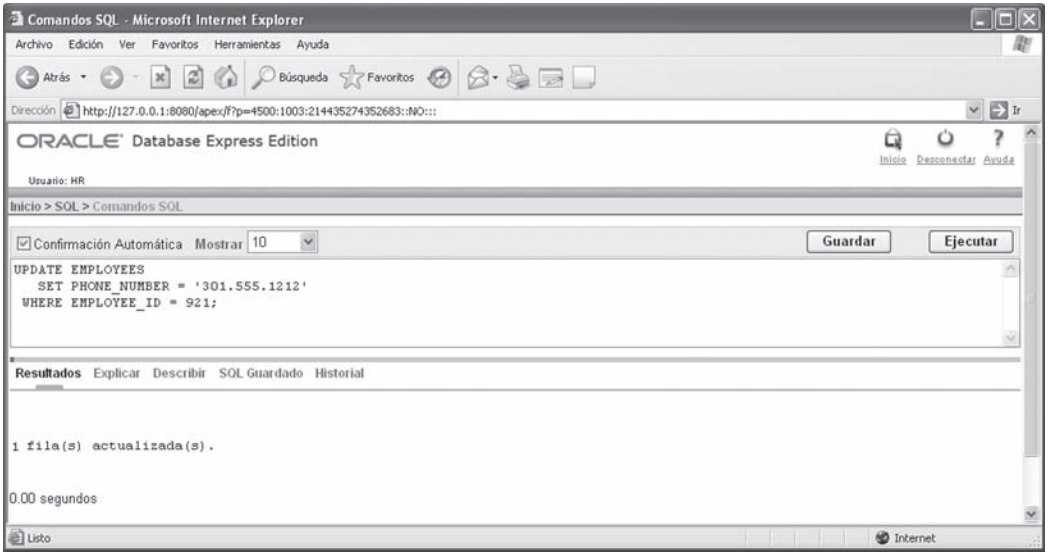


Figura 4-27 Instrucción UPDATE para la tabla EMPLOYEES.

Para cada columna que se va actualizar, se emplea una cláusula SET para nombrar la columna y el valor nuevo de la columna. El valor nuevo proporcionado puede ser una constante, el nombre de otra columna o cualquier expresión que SQL pueda resolver para el valor de una columna. Si la cláusula SET hace referencia a varias columnas, los nombres y valores de las columnas deben estar en una lista separada con comas. La instrucción UPDATE puede incluir una cláusula WHERE para limitar las filas afectadas por la instrucción. Si se omite la cláusula WHERE, la instrucción UPDATE intenta actualizar cada fila de la tabla (o vista). Si olvida este punto importante, no olvide la amigable instrucción ROLLBACK, que revierte los resultados de la actualización (por supuesto, a menos que esté en el modo actualización automática).

La instrucción DELETE

La instrucción DELETE elimina una o más filas de una tabla. También puede hacer referencia a una vista, pero sólo si ésta se basa en una tabla única (en otras palabras, no es posible hacer referencia a vistas que combinan varias tablas). Una instrucción DELETE no hace referencia a columnas porque borra automáticamente todos los datos de éstas para cualquier fila eliminada. Es posible incluir una cláusula WHERE para limitar las filas afectadas por la instrucción DELETE; si se omite la cláusula WHERE, la instrucción intenta eliminar todas las filas en la tabla a que se hace referencia. En la figura 4-28 se muestra un ejemplo de una ins-

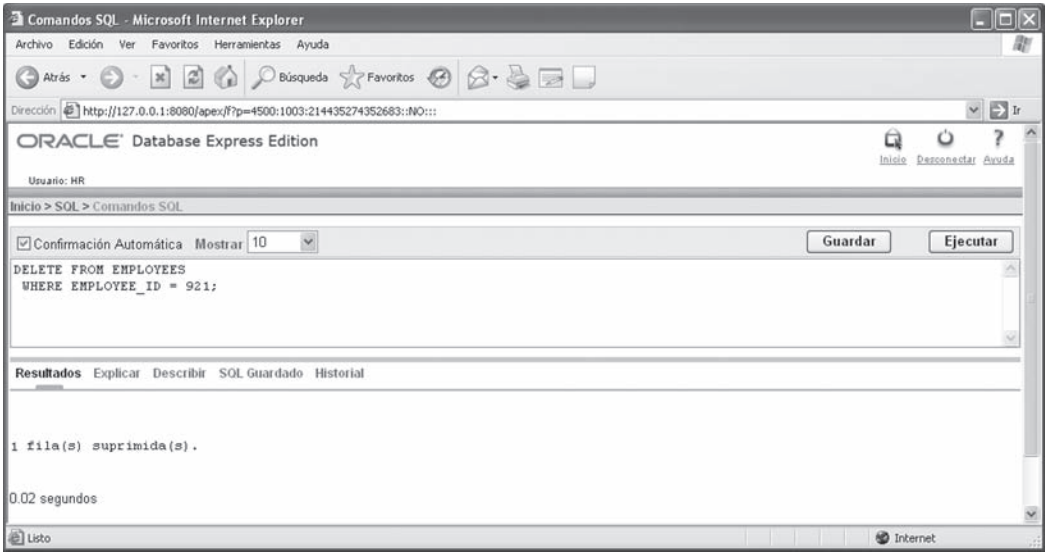


Figura 4-28 Instrucción DELETE para la tabla EMPLOYEES.

trucción DELETE que elimina al empleado 921 de la tabla EMPLOYEES. Aquí está la instrucción SQL:

```
DELETE FROM EMPLOYEES
 WHERE EMPLOYEE_ID = 921;
```

Instrucciones del lenguaje de definición de datos (DDL)

Las instrucciones del lenguaje de definición de datos (Data Definition Language, DDL) definen los objetos de la base de datos, pero no insertan ni actualizan los datos guardados dentro de esos objetos. (Las instrucciones de DML sirven para ese propósito.) En SQL se emplean tres comandos básicos dentro de DDL:

- **CREATE** Crea un nuevo objeto de la base de datos del tipo mencionado en la instrucción.
- **DROP** Descarta (destruye) un objeto existente en la base de datos del tipo indicado en la instrucción.
- **ALTER** Modifica la definición de un objeto existente en la base de datos del tipo señalado en la instrucción.

En las secciones siguientes se examinan los tipos de instrucciones DDL de uso más frecuente. Encontrará mucha diversidad en las instrucciones DDL que ofrecen los distintos vendedores, de modo que debe enterarse de los detalles de la documentación de cada uno.

La instrucción CREATE TABLE

La instrucción CREATE TABLE incorpora una tabla nueva a la base de datos. Éste es un ejemplo que crea la tabla EMPLOYEE_INPUT con las mismas definiciones de columnas que la tabla EMPLOYEES:

```
CREATE TABLE EMPLOYEE_INPUT (
 EMPLOYEE_ID      NUMBER(6)      NOT NULL,
 FIRST_NAME       VARCHAR2(20)   NULL,
 LAST_NAME        VARCHAR2(25)   NOT NULL,
 EMAIL            VARCHAR2(25)   NOT NULL,
 PHONE_NUMBER     VARCHAR2(20)   NULL,
 HIRE_DATE        DATE           NOT NULL,
 JOB_ID           VARCHAR2(10)   NOT NULL,
 SALARY           NUMBER(8,2)    NULL,
 COMMISSION_PCT   NUMBER(2,2)    NULL,
 MANAGER_ID       NUMBER(6)      NULL,
 DEPARTMENT_ID    NUMBER(4)      NULL)
 ;
```

Observe que se proporciona una lista de columnas separadas por comas, junto con el tipo de datos y la especificación NULL o NOT NULL para cada columna. Debe recordar, del capítulo 2, que los vendedores de RDBMS permiten una amplia variedad de tipos de datos. Los presentados aquí se aplican a Oracle. Tenga cuidado con las especificaciones NULL y NOT NULL. En casi todos los RDBMS, entre ellos Oracle, NULL es la opción predeterminada. Sin embargo, en otros puede ser NOT NULL. Por lo tanto, es más seguro, aunque también requiere más trabajo especificar siempre NULL o NOT NULL. Por cierto, casi todos los RDBMS requieren que las columnas de clave principal se especifiquen como NOT NULL. Verá cómo crear una restricción de clave principal en la columna EMPLOYEE_ID de esta tabla en la sección “Restricciones de clave principal”, más adelante en el capítulo.

Existen muchas extensiones de vendedores para la instrucción CREATE TABLE, más allá de la lista de columnas básica de este ejemplo. Por ejemplo, en Oracle, se incluye la cláusula STORAGE para especificar la cantidad de espacio físico asignado a la tabla, y existe una cláusula TABLESPACE para especificar el espacio que contendrá los datos de la tabla.

La instrucción ALTER TABLE

La instrucción ALTER TABLE sirve para modificar muchos aspectos de la definición de una tabla de una base de datos. Una vez más, existe una amplia variación en la implementación entre los diversos vendedores de RDBMS; en términos generales, se hacen los siguientes tipos de cambios mediante la instrucción ALTER TABLE:

- Agregar columnas a la tabla.
- Eliminar columnas de la tabla.
- Modificar el tipo de datos para las columnas de una tabla existente.
- Cambiar los atributos del almacenamiento físico de la tabla.
- Incorporar, eliminar o modificar las restricciones.

Debido a que la implementación de restricciones es el modo en que se imponen reglas de negocios a las bases de datos, aquí se analizarán con detenimiento. Es importante que nombre las restricciones, porque en casi todas las implementaciones de SQL, esos nombres aparecen en los mensajes de error generados cuando ocurren violaciones a las restricciones.

Restricciones referenciales

Éste es un ejemplo de una definición de restricción referencial mediante la instrucción ALTER TABLE:

```
ALTER TABLE EMPLOYEE_INPUT
  ADD CONSTRAINT EMP_INPUT_DEPT_FK
      FOREIGN KEY (DEPARTMENT_ID)
      REFERENCES DEPARTMENTS (DEPARTMENT_ID);
```

En este ejemplo se agregó una instrucción referencial llamada EMP_DEPT_FK a la tabla EMPLOYEES con el propósito de definir la columna DEPARTMENT_ID como una clave externa para la columna de clave principal (DEPARTMENT_ID) de la tabla DEPARTMENTS. De este modo implementa las relaciones que identificó en el diseño lógico de la base de datos.

Restricciones de clave principal

Las restricciones de clave principal aseguran que las columnas designadas así para la tabla nunca tengan valores duplicados. Casi todos los RDBMS, entre ellos Oracle, crean un índice único para ayudar a imponer las restricciones de clave principal. Un *índice* es un objeto especial de la base de datos que contiene el valor de la clave de una o más columnas de tabla y apuntadores a las filas de la tabla que coinciden con el valor de la clave. Los índices se usan

para realizar una búsqueda rápida en una tabla con base en el valor de la clave. Ésta es la definición de la restricción de clave principal de la tabla EMPLOYEES:

```
ALTER TABLE EMPLOYEE_INPUT
 ADD CONSTRAINT EMPLOYEES_PK
     PRIMARY KEY (EMPLOYEE_ID)
     USING INDEX;
```

Restricciones de unicidad

Además de imponer claves principales, puede imponer la unicidad de otras columnas en una tabla mediante una restricción de unicidad. Una tabla sólo puede tener una restricción de clave principal, pero puede tener todas las restricciones de unicidad que sean necesarias. Casi todos los RDBMS, Oracle incluido, emplean un índice de unicidad para ayudar a imponer las restricciones de unicidad. Por ejemplo, puede emplear una restricción de unicidad para asegurar que dos empleados no tengan la misma dirección de correo electrónico del modo siguiente:

```
ALTER TABLE EMPLOYEE_INPUT
  ADD CONSTRAINT EMPLOYEES_UNQ_EMAIL
      UNIQUE (EMAIL);
```

Esta misma restricción se elimina mediante esta instrucción:

```
ALTER TABLE EMPLOYEE_INPUT
 DROP CONSTRAINT EMPLOYEES_UNQ_EMAIL;
```

Restricciones de comprobación

Las restricciones de comprobación se utilizan para imponer cualquier regla de negocios que se pueda aplicar a una sola columna de una tabla. La condición incluida en la restricción siempre debe ser verdadera cuando se cambian los datos de la columna en la tabla; de otro modo, fracasa la instrucción SQL y se muestra un mensaje de error. El ejemplo siguiente implementa una restricción de comprobación que asegura que la columna SALARY de la tabla EMPLOYEES siempre sea mayor que cero:

```
ALTER TABLE EMPLOYEES
  ADD CONSTRAINT EMPLOYEES_CHK_SALARY_MIN
     CHECK (SALARY > 0);
```

La misma restricción se retira con esta instrucción:

```
ALTER TABLE EMPLOYEES
 DROP CONSTRAINT EMPLOYEES_CHK_SALARY_MIN;
```

La instrucción CREATE VIEW

Debido a que una vista es simplemente una consulta guardada, cualquier consulta que se pueda ejecutar mediante una instrucción SELECT se puede guardar como una vista en la base

de datos. Los nombres de vista deben ser únicos entre todas las tablas, vistas y sinónimos en el esquema de la base de datos. En Oracle, es posible incluir la opción **OR REPLACE** para reemplazar una vista existente del mismo nombre. En el ejemplo siguiente se crea una vista para la consulta presentada en la figura 4-21:

```
CREATE OR REPLACE VIEW SALES_EMPLOYEES AS
   SELECT EMPLOYEE_ID, LAST_NAME, FIRST_NAME, DEPARTMENT_NAME
     FROM EMPLOYEES LEFT OUTER JOIN DEPARTMENTS
          ON EMPLOYEES.DEPARTMENT_ID = DEPARTMENTS.DEPARTMENT_ID
   WHERE DEPARTMENT_NAME LIKE '%Sales%';
```

La ejecución de la siguiente instrucción SQL seleccionará los datos de la vista, lo que producirá los mismos resultados exactos que los mostrados en la figura 4-21:

```
SELECT *
  FROM SALES_EMPLOYEES;
```

La instrucción CREATE INDEX

La instrucción CREATE INDEX genera un índice en una o más columnas de una tabla. Como ya se indicó, los índices proporcionan una búsqueda rápida en una tabla con base en una o más columnas de clave. Los índices en claves externas también mejoran mucho el rendimiento de las combinaciones. El RDBMS conserva automáticamente el índice cuando se agregan o eliminan filas en la base de datos, o cuando se actualizan los valores de una columna indizada. Sin embargo, los índices ocupan espacio de almacenamiento y su mantenimiento acapara recursos de procesamiento. Con el ejemplo siguiente se crea un índice en la columna DEPARTMENT_ID en la tabla EMPLOYEE_INPUT:

```
CREATE INDEX EMPLOYEE_INPUT_IX_DEPT_ID
    ON EMPLOYEE_INPUT (DEPARTMENT_ID);
```

Si los valores de la columna en el índice siempre serán únicos, se coloca la palabra clave **UNIQUE** entre las palabras clave **CREATE** e **INDEX**. O se puede incluir una restricción de unicidad a la tabla, lo que crea indirectamente el índice único. Los índices únicos suelen ser más eficientes que los que no lo son.

La instrucción DROP

La instrucción DROP se utiliza para eliminar objetos de la base de datos cuando ya no son necesarios. Para eliminar tablas, se puede agregar la cláusula CASCADE CONSTRAINTS (que se abrevia como CASCADE en ciertas implementaciones de SQL) para eliminar automáticamente cualquier restricción referencial en que participe la tabla. Cuando se descarta una tabla, también se descartan casi todos los objetos que dependen de ella (índices y restric-

ciones). No obstante, en casi todos los RDBMS permanecen las vistas que dependen de una tabla descartada, pero se marcan como no válidas para que no puedan utilizarse hasta que se vuelva a crear la tabla. Éstas son las instrucciones DROP que eliminan los objetos creados en los ejemplos anteriores:

```
DROP VIEW SALES_EMPLOYEES;
DROP INDEX EMPLOYEE_INPUT_IX_DEPT_ID;
DROP TABLE EMPLOYEE_INPUT CASCADE CONSTRAINTS;
```

NOTA

Tal vez tenga que ejecutar estas instrucciones de una en una. Ésta parece ser una peculiaridad del cliente de Oracle XE.

Instrucciones del lenguaje de control de datos (DCL)

Un *privilegio* de una base de datos es la autorización para hacer algo en ella. Al usuario de la base de datos que concede el privilegio se le llama *otorgante* y al usuario que recibe el privilegio se le denomina *concesionario*. Los privilegios caen en dos categorías amplias:

- **Privilegios del sistema** Permiten que el concesionario aplique una función general a la base de datos, como crear nuevas cuentas de usuario o conectarse a la base de datos.
- **Privilegios de objeto** Permiten al concesionario efectuar acciones específicas sobre objetos determinados, como seleccionar de la tabla EMPLOYEES o actualizar la tabla DEPARTMENTS.

Para reducir el tedio de administrar privilegios, casi todos los RDBMS permiten guardar un grupo de definiciones de privilegios como un solo objeto llamado *función*. Por lo tanto, las funciones pueden concederse a usuarios individuales, que heredan todos los privilegios contenidos en su rol. Los RDBMS que permiten funciones también suelen venir con varias funciones predefinidas. Por ejemplo, Oracle tiene una función llamada DBA que contiene los importantes privilegios del sistema y de objetos que un usuario necesita para administrar una base de datos.

La instrucción GRANT

Se conceden privilegios a los usuarios de SQL mediante la instrucción GRANT. En los ejemplos siguientes se presenta la sintaxis para conceder un privilegio del sistema y un privilegio de objeto a los usuarios de una base de datos. La cuenta del usuario que concede el privilegio debe poseer ese privilegio, de modo que muchos de los ejemplos siguientes no funcionarán a menos que esté conectado a la base de datos mediante la cuenta SYSTEM.

La siguiente instrucción concede el privilegio CREATE VIEW al usuario HR:

```
GRANT CREATE VIEW TO HR;
```

La instrucción siguiente concede privilegios para seleccionar, insertar y actualizar la tabla EMPLOYEES en el esquema HR al usuario HR_ADMIN. Recuerde que debe calificar el nombre de la tabla con el del esquema, si está conectado como un usuario diferente, como SYSTEM. Siempre debe calificar los objetos que pertenecen a otro esquema (usuario) cuando hace referencia a ellos en SQL. Ésta es la instrucción:

```
GRANT SELECT, INSERT, UPDATE
   ON HR.EMPLOYEES TO HR_ADMIN;
```

NOTA

La cuenta del usuario HR_ADMIN debe existir para que funcione esta instrucción. Si quiere probarla, utilice el ícono Administración, de la página principal, y seleccione Usuarios de Base de Datos | Crear Usuario, para generar la cuenta. En este capítulo no se cubre la creación de cuentas de usuarios porque en la norma de SQL no existe una sintaxis para eso, lo que significa que cada vendedor ofrece una solución patentada.

Casi todos los RDBMS que permiten privilegios también otorgan permiso al concesionario para conceder el privilegio a otros. En Oracle, la cláusula para hacerlo es WITH ADMIN OPTION para privilegios del sistema y WITH GRANT OPTION para privilegios de objetos. Sin embargo, se recomienda *enfáticamente* que se *evite* hacer esto. Es demasiado fácil perder el control de los privilegios cuando se permite a las personas que poseen un privilegio concederlo a los demás.

La instrucción REVOKE

Los privilegios concedidos pueden retirarse mediante la instrucción REVOKE. En el caso de los privilegios de objetos, si el usuario usa WITH GRANT OPTION, la revocación se aplica en cascada y las personas que se encuentran en el siguiente nivel de privilegios también los pierden. No siempre es cierto esto para los privilegios del sistema: consulte los detalles en los manuales de su RDBMS. Mejor aún, si nunca utiliza WITH GRANT OPTION y WITH ADMIN OPTION, nunca tendrá que preocuparse por este problema. Los privilegios presentados en la sección anterior se revocan con estos comandos:

```
REVOKE CREATE VIEW FROM HR;

REVOKE SELECT, INSERT, UPDATE
    ON HR.EMPLOYEES FROM HR_ADMIN;
```

Autoexamen Capítulo 4

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. SQL se divide en los subconjuntos siguientes:

- **A** Lenguaje de selección de datos (DSL).
- **B** Lenguaje de control de datos (DCL).
- **C** Lenguaje de consulta de datos (DQL).
- **D** Lenguaje de definición de datos (DDL).

2. SQL fue desarrollado por _____.

3. A un programa utilizado para conectarse a la base de datos e interactuar con ella se le llama _____.

4. Una instrucción SELECT sin una cláusula WHERE

- **A** Selecciona todas las filas de la tabla o vista de origen.
- **B** No devuelve filas del grupo de resultados.
- **C** Produce un mensaje de error.
- **D** Sólo presenta la definición de la tabla o vista.

5. En SQL, el orden de las filas en los resultados de la consulta

- **A** Es especificado por la cláusula SORTED BY.
- **B** Es impredecible, a menos que se especifique en la consulta.
- **C** Cuando no se especifica una secuencia, la opción predeterminada es descendente.
- **D** Sólo se puede especificar para las columnas en los resultados de la consulta.

6. El operador BETWEEN

- **A** Incluye los valores del punto final.
- **B** Selecciona las filas que se agregan a una tabla durante un intervalo.
- **C** También se puede escribir mediante los operadores **<=** y **NOT =**.
- **D** También se puede escribir mediante los operadores **<=** y **>=**.

7. El operador LIKE emplea _____ como comodines de posición y _____ como comodines que no están relacionados con la posición.

8. Una selección secundaria

- **A** Puede ser corrugada o no corrugada.
- **B** Permite una selección flexible de las filas.
- **C** No debe estar entre paréntesis.
- **D** Puede usarse para seleccionar valores para aplicarlos a las condiciones de la cláusula WHERE.

9. Una combinación sin una cláusula WHERE o JOIN

- **A** Produce un mensaje de error.
- **B** Genera una combinación externa.
- **C** Obtiene un producto cartesiano.
- **D** No devuelve filas en el grupo de resultados.

10. A una combinación que devuelve todas las filas de ambas tablas, se encuentren coincidencias o no, se le conoce como _____.

11. Una autocombinación

- **A** Incluye dos tablas diferentes.
- **B** Puede ser una combinación interna o externa.
- **C** Resuelve las relaciones recursivas.
- **D** Puede emplear una selección secundaria para limitar más las filas devueltas.

12. Una instrucción SQL que contiene una función de obtención de totales

- **A** Debe contener una cláusula GROUP BY.
- **B** También puede incluir columnas comunes.
- **C** No puede incluir cláusulas GROUP BY y ORDER BY.
- **D** También puede incluir columnas calculadas.

13. _____ hace que los cambios realizados por una transacción adquieran permanencia.

14. Una instrucción INSERT

- **A** Debe contener una lista de columnas.
- **B** Debe contener una lista de VALORES.
- **C** Puede crear varias filas de una tabla.
- **D** Puede contener una consulta secundaria.

15. Una instrucción UPDATE sin una cláusula WHERE

- **A** Produce un mensaje de error.
- **B** No actualiza las filas de una tabla.
- **C** Actualiza cada fila de una tabla.
- **D** Genera un producto cartesiano.

16. Una instrucción DELETE sin una lista de columnas

- **A** Produce un mensaje de error.
- **B** Sólo elimina datos de las columnas de la lista.
- **C** Elimina cada columna de la tabla.
- **D** Se utiliza para eliminar de una vista.

17. Una instrucción CREATE

- **A** Es una forma de DML.
- **B** Crea nuevos privilegios del usuario.
- **C** Genera un objeto de la base de datos.
- **D** Se puede revertir después mediante una instrucción DROP.

18. Una instrucción ALTER

- **A** Se utiliza para agregar una restricción.
- **B** Sirve para descartar una restricción.
- **C** Es útil para agregar una vista.
- **D** Se emplea para descartar una columna de una tabla.

19. El modo _____ hace que cada instrucción SQL se confirme en cuanto concluye.

20. Los privilegios de una base de datos

- **A** Se modifican con una instrucción ALTER PRIVILEGE.
- **B** Pueden ser del sistema o de objetos.
- **C** Se administran mejor cuando se integran en grupos mediante GROUP BY.
- **D** Son administrados mediante GRANT y REVOKE.

Parte II

Desarrollo de una base de datos

Capítulo 5

El ciclo de vida de una base de datos

Habilidades y conceptos clave

- El ciclo de vida tradicional.
- Ciclos de vida no tradicionales.
- El triángulo del proyecto.

Antes de ahondar en los detalles del diseño de una base de datos, le resultará útil comprender el marco conceptual en que se presenta el diseño. El *ciclo de vida* de una base de datos (o sistema computacional) abarca todos los eventos que ocurren desde el momento en que se reconoce la necesidad de una base de datos, pasa por su desarrollo y despliegue, y concluye el día en que la base de datos se retira del servicio.

Casi todas las empresas que desarrollan sistemas computacionales siguen un proceso formal que asegura que el desarrollo funcione sin contratiempos, que su costo sea eficiente y que el resultado sea un sistema que cumpla las expectativas. Las bases de datos nunca se diseñan e implementan en el vacío; junto con ellas siempre se desarrollan otros componentes del sistema, como la interfaz del usuario, los programas de aplicaciones y los informes. Todo el trabajo que se va a efectuar a largo plazo se suele dividir en *proyectos*, y cada proyecto tiene su propia lista finita de metas (también llamadas *resultados*), un marco de tiempo esperado para su conclusión, y un administrador o líder del proyecto que será responsable de entregarlo. Para comprender el ciclo de vida de una base de datos, también debe comprender el de todo el esfuerzo de desarrollo de sistemas y el modo en que se organizan y administran los proyectos. En este capítulo se analizan los procesos tradicionales y no tradicionales del desarrollo de sistemas.

No todas las bases de datos son creadas por empresas que utilizan proyectos y financiamiento formales. Sin embargo, las disciplinas resumidas en este capítulo le ayudan a reflexionar sobre su proyecto de base de datos y a formular las preguntas importantes antes de emprender un esfuerzo pormenorizado.

El ciclo de vida tradicional

El método tradicional para desarrollar sistemas computacionales sigue un proceso llamado *ciclo de vida del desarrollo del sistema (CVDS)*, que divide el trabajo en las fases presentadas en la figura 5-1. Es posible que existan tantas variaciones del CVDS como autores, vendedores de software para administración de proyectos y empresas que han optado por crear sus propias metodologías. Sin embargo, todas tienen los componentes básicos y, en ese sentido, todas están hechas de la misma tela. Es posible debatir los méritos de una variación sobre otra, pero eso simplemente confundiría el tema cuando sólo necesita un compendio básico.

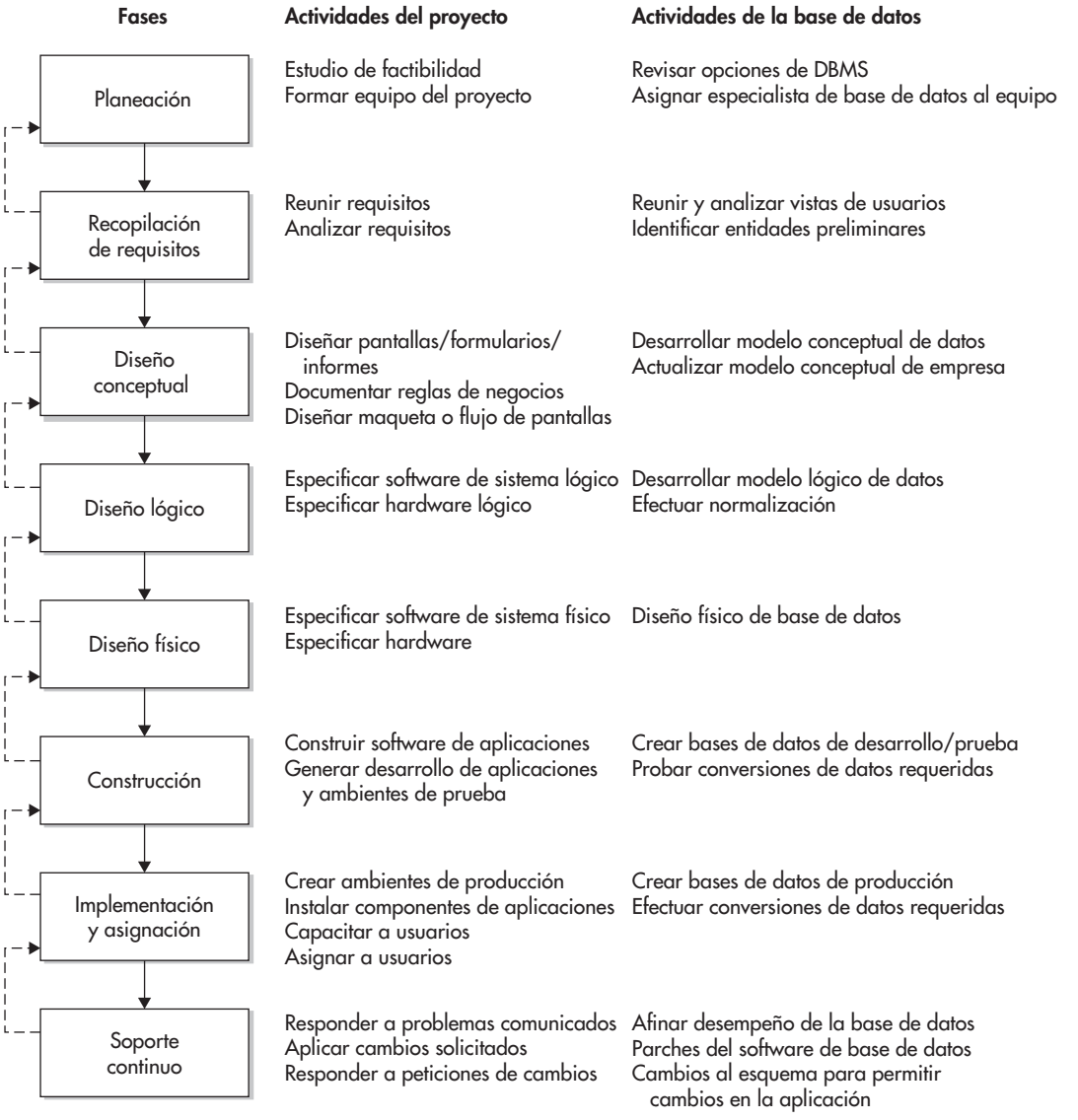


Figura 5-1 Ciclo de vida del desarrollo de sistemas (CVDS) tradicional.

Un buen texto sobre análisis de sistemas puede ofrecer mayores detalles en caso de que los necesite. En la figura 5-1 se muestran los pasos del CVDS tradicional en la columna izquierda, las actividades básicas del proyecto en la columna intermedia, y los pasos en la base de datos que permiten las actividades del proyecto en la columna derecha. Cada paso se explora con mayor detalle en las secciones siguientes. Observe que el proceso no es siempre unidirec-

cional; en ocasiones, descubrirá que falta información o está incompleta, y eso requiere que regrese a una fase y ajuste el trabajo realizado allí. Las líneas de guiones que apuntan a fases anteriores de la figura 5-1 sirven para recordar que es normal y esperada cierta cantidad de reelaboración durante un proyecto que sigue la metodología CVDS.

Planeación

Durante la fase de planeación, la organización debe alcanzar una comprensión de alto nivel acerca de dónde está en ese momento, dónde quiere estar, y un método o plan razonable para pasar de un lugar a otro. La planeación suele ocurrir durante un periodo más prolongado que en cualquier proyecto individual, y el plan general de sistemas de información para la organización forma la base para identificar cuáles proyectos deben lanzarse con el fin de lograr los objetivos generales. Por ejemplo, un objetivo a largo plazo en el plan podría ser "Aumentar 15% las utilidades". Para apoyar este objetivo, podría proponerse un proyecto para desarrollar un sistema de aplicaciones y de base de datos que registre la rentabilidad de los clientes.

Una vez propuesto un proyecto específico, se suele lanzar un *estudio de factibilidad* para determinar si es razonable esperar que el proyecto alcance (o ayude a lograr) el objetivo y si las estimaciones preliminares de tiempo, personal y materiales para el proyecto son compatibles con el marco de tiempo requerido y el presupuesto disponible. A menudo se emplea el rendimiento sobre la inversión o un cálculo similar para medir el valor esperado del proyecto propuesto para la organización. Si el estudio de factibilidad obtiene la aprobación de la directiva, el proyecto se incorpora al programa general de la organización y se forma el equipo del proyecto. La composición del equipo se modificará durante la vida del proyecto, y se incorporarán y descartarán personas conforme se requieran niveles específicos de conocimientos y personal. El único integrante persistente del equipo del proyecto será el *administrador del proyecto* (o líder del proyecto), quien es responsable de la administración general y la ejecución del proyecto.

Muchas organizaciones asignan a los proyectos un *especialista en bases de datos* (un administrador de base de datos o un modelador de datos) en el momento de crearse, como se observa en la figura 5-1. En un método *orientado a datos*, en donde se hace hincapié en estudiar los datos para descubrir el procesamiento que desde ocurrir con el fin de transformarlos de acuerdo con las necesidades del proyecto, es esencial la incorporación inmediata de una persona con talento para analizar los datos. En un método *orientado a los procesos*, en que se hace hincapié en el estudio de los procesos requeridos para descubrir cuáles deben ser los datos, es menos esencial un especialista en base de datos durante las fases iniciales del proyecto. La experiencia en la industria sugiere que los mejores resultados se obtienen al aplicar *ambos* métodos: uno orientado a datos y otro a los procesos. Sin embargo, rara vez se tienen el tiempo y el personal para hacerlo, de modo que el siguiente mejor resultado de un proyecto relacionado con bases de datos se obtiene del método orientado a datos. Todavía es necesario diseñar procesos, pero si primero estudiamos los datos, los procesos requeridos se vuelven

evidentes. Por ejemplo, al diseñar un sistema de rentabilidad de clientes, si se tienen datos de las ventas a éstos y se sabe que son más rentables los clientes que hacen menos pedidos pero de mayor volumen, se deduce que se requiere un proceso para clasificar a los clientes por volumen y tamaño de pedidos. Por otra parte, si sólo se sabe que se necesita un proceso que clasifique a los clientes, tal vez se necesite más trabajo para obtener los criterios que se deben emplear para clasificarlos.

En esta fase, las actividades de la base de datos incluyen la revisión de las opciones de DBMS y la determinación de que las tecnologías en uso en la actualidad cubren las necesidades generales del proyecto. Casi todas las organizaciones se deciden por un producto DBMS estándar (o tal vez dos), que emplea para todos los proyectos. En este momento, las metas del proyecto deben compararse con la tecnología actual para asegurar que se puede esperar razonablemente que el proyecto tenga éxito con esa tecnología. Si se requiere una versión más reciente del DBMS, o una completamente diferente, es durante la fase de planeación cuando se debe determinarlo, para que pueda comenzar la adquisición y la instalación del DBMS.

Recopilación de requisitos

Durante la fase de recopilación de requisitos, el equipo del proyecto desde reunir y documentar una descripción de alto nivel, y al mismo tiempo precisa, de lo que va a lograr el proyecto. Debe centrarse en *qué* más que en *cómo*; él cómo se desarrolla durante las fases siguientes del diseño. Es importante que los requisitos incluyan todo lo que se pueda saber sobre los procesos de negocios existentes y esperados, las reglas de negocios y las entidades. Cuanto más trabajo se realice en las etapas iniciales del proyecto, con menos contratiempos se avanzará en las etapas posteriores. Por otra parte, sin cierta tolerancia a lo desconocido (es decir, las áreas grises que todavía no tienen respuestas firmes), puede ocurrir una *parálisis del análisis*, en donde el proyecto completo se detiene mientras los analistas se esfuerzan por encontrar respuestas y aclaraciones que no aparecen.

Desde la perspectiva del diseño de una base de datos, los elementos de mayor interés durante la recopilación de requisitos son las vistas de usuarios. Recuerde que una *vista de usuario* es el método utilizado para presentar un grupo de datos al usuario de la base de datos de una manera adaptada a las necesidades de la persona o aplicación. En esta fase de desarrollo, las vistas de usuario adoptan la forma de informes, formularios, pantallas o páginas Web existentes o propuestos.

Se utilizan muchas técnicas en la recopilación de requisitos. Las más frecuentes se comparan y contrastan aquí: realización de entrevistas, realización de encuestas, observación y revisión de documentos. Ninguna técnica específica es claramente superior a otra, y es mejor determinar una combinación de técnicas que funcione bien para la organización en particular en lugar de confiar en una sobre las demás. Por ejemplo, si es mejor efectuar una encuesta y dar seguimiento por medio de entrevistas a personas importantes, o comenzar con entrevistas y utilizar los hallazgos de éstas para formular una encuesta, eso suele determinarse a partir de

lo que funciona mejor para la cultura y los métodos operativos de la organización. Con cada técnica detallada en las secciones secundarias siguientes, se incluyen algunas ventajas y desventajas para ayudar a la toma de decisiones.

Realización de entrevistas

Un método popular consiste en entrevistar a personas importantes que poseen información sobre lo que se espera conseguir con el proyecto. Sin embargo, uno de los errores comunes es entrevistar sólo a los directivos. Si no incluye a quienes en realidad van a usar las aplicaciones y las bases de datos nuevas, el proyecto puede terminar con la entrega de algo que no es práctico, porque la administración tal vez no comprende por completo todos los detalles de los requisitos necesarios para dirigir el negocio de la organización. En particular, necesita iniciar un diálogo con uno o más *expertos en la materia*: profesionales que tienen experiencia en el campo de la aplicación, pero que no suelen poseer conocimientos técnicos de sistemas computacionales.

Entre las ventajas de la recopilación de requisitos mediante entrevistas están:

- El entrevistador puede obtener respuestas a preguntas que no fueron formuladas. Suelen surgir temas colaterales que aportan información útil adicional.
- El entrevistador puede aprender mucho del lenguaje corporal del entrevistado. En persona es mucho más fácil detectar incertidumbre e intentos de engaño que en las respuestas escritas.

Algunas desventajas son:

- Las entrevistas consumen mucho más tiempo que otros métodos.
- Los entrevistadores con poca experiencia pueden "telegrafiar" las respuestas que esperan por el modo en que plantean las preguntas o por sus reacciones a las respuestas recibidas.

Realización de encuestas

Otro método popular consiste en redactar una encuesta para buscar respuestas a preguntas importantes en relación con los requisitos para un proyecto. La encuesta se envía a todas las personas que toman decisiones y a los posibles usuarios de las aplicaciones y las bases de datos que el proyecto espera entregar, y se analizan las respuestas de los conceptos que se van a incluir en los requisitos.

Algunas ventajas de la recopilación de requisitos mediante encuestas son:

- Se cubre mucho terreno en poco tiempo. Una vez redactada la encuesta, se requiere un mínimo esfuerzo adicional para distribuirla a un número aún mayor de personas, si es necesario.
- Las preguntas se plantean del mismo modo para cada participante.

Entre sus desventajas están:

- Las encuestas suelen tener tasas de respuestas muy deficientes. Se debe considerar afortunado si 10% responde sin tener que ser estimulado o amenazado con las consecuencias.
- Redactar preguntas imparciales es mucho más difícil de lo que se imagina.
- El equipo del proyecto no recibe los beneficios de las señales no verbales que aporta una entrevista.

Observación

Otra técnica popular para recopilar requisitos consiste en observar la operación del negocio y a las personas que utilizarán las aplicaciones y bases de datos nuevas.

A continuación se presentan algunas ventajas de la recopilación de requisitos mediante observaciones:

- Siempre y cuando observe de manera discreta, contemplará a las personas aplicando los procesos normales de uso cotidiano. Tome en cuenta que tal vez no sean los procesos que la administración considera que se siguen, o que ni siquiera sean los que aparecen en la documentación existente. En cambio, puede encontrarse con adaptaciones que fueron hechas para que los procesos funcionen en la realidad o para que sean más eficientes.
- Puede observar eventos que las personas no pensarían o no se atreverían a mencionar como respuesta a las preguntas en un cuestionario o una entrevista.

Entre sus desventajas están:

- Si las personas saben que están siendo observadas, cambian su conducta, y es posible que no obtenga una imagen precisa de sus procesos de negocios. A esto suele denominársele *efecto Hawthorne*, porque es un fenómeno que se vio por primera vez en la planta Hawthorne, de Western Electric, donde la producción no aumentó por mejorar las condiciones de trabajo, sino porque la administración mostró interés en esas mejoras.
- A menos que dedique mucho tiempo a la observación, puede ocurrir que nunca presencie las excepciones que subvierten los procesos de negocios existentes. Igual que ocurre en una antigua fábula, mientras aplana el terreno para las vacas, éstas escapan al área vecina por un hueco en la cerca.
- Viajar a las diferentes instalaciones de negocios puede aumentar considerablemente el gasto en el proyecto.

Revisión de documentos

Esta técnica incluye la localización y revisión de todos los documentos disponibles de las unidades y procesos empresariales existentes que serán afectados por los programas y bases de datos nuevos.

La recopilación de requisitos por medio de documentos tiene estas ventajas:

- La revisión de documentos suele consumir menos tiempo que cualquiera de los otros métodos.
- Los documentos suelen proporcionar un compendio del sistema que se analiza mejor que la información introductoria que se obtiene en una entrevista.
- Cada imagen y diagrama en verdad vale más que mil palabras.

Entre las desventajas que existen están las siguientes:

- Es posible que los documentos no reflejen las prácticas actuales. Los documentos suelen referirse a lo que *debe* ocurrir, en lugar de a lo *realmente* sucede.
- La documentación suele estar desactualizada.

Diseño conceptual

La fase de diseño conceptual incluye el diseño de los elementos externos de las aplicaciones y bases de datos. En realidad, muchas metodologías emplean el término *diseño externo* para esta fase del proyecto. En esta etapa finaliza el diseño de los informes, pantallas, formularios, página Web y otros recursos para introducir y presentar datos. Además, se documenta el flujo de la aplicación externa como un diagrama de flujo, una maqueta de dibujos o un diagrama de flujo de pantallas. Esto ayuda al equipo del proyecto a comprender el flujo lógico del sistema. Las técnicas para diagramar procesos se analizan con mayor detalle en el capítulo 7.

Durante esta fase, el especialista en base de datos (DBA o modelador de datos) asignado el proyecto actualiza el modelo conceptual de datos de la empresa, que suele existir en forma de un diagrama entidad-relación (ERD). Las entidades nuevas o modificadas descubiertas se incorporan al ERD, y también se indica cualquier regla de negocios adicional o modificada. Las vistas de usuarios, las entidades y las reglas de negocios son esenciales para el diseño lógico correcto de una base de datos que ocurre en la fase siguiente.

Diseño lógico

Durante el diseño lógico, se efectúa gran parte del diseño técnico de las aplicaciones y bases de datos incluidas en el proyecto. Muchas metodologías llaman a esta fase *diseño interno*, porque requiere el diseño de los elementos internos del proyecto que los usuarios de negocios nunca verán.

El trabajo que habrán de realizar las aplicaciones se divide en *módulos* (unidades individuales de programación de aplicaciones que se escribirán y probarán juntas), y se redacta una especificación detallada para cada unidad. Esta especificación debe ser lo bastante completa para que cualquier programador con las habilidades adecuadas pueda escribir el módulo y probarlo con poca o ninguna información adicional. Se suelen emplear diagramas de flujo de

datos o flujogramas (una técnica más antigua) para documentar el flujo lógico entre los módulos. El modelado de procesos se cubre con mayor detalle en el capítulo 7.

Desde la perspectiva de una base de datos, en esta fase el principal esfuerzo es la *normalización*, técnica desarrollada por E. F. (Ted) Cody para diseñar las tablas de una base de datos relacional que funcionan mejor para los sistemas basados en transacciones (es decir, los que insertan, actualizan y eliminan datos en las tablas de una base de datos relacional). La normalización, el tema más importante de todo el libro, se cubre con mayor detalle en el capítulo 6. Una vez concluida la normalización, se actualiza el modelo lógico de datos general para la empresa (suponiendo que existe una) para reflejar las entidades recién descubiertas.

Diseño físico

Durante la fase de diseño físico, el diseño lógico se ubica o se convierte en el hardware y los sistemas de software reales que servirán para implementar las aplicaciones y las bases de datos. Desde el lado de los procesos, poco o nada necesita hacerse si las especificaciones de aplicaciones se escribieron de manera que se pueden implementar directamente. Sin embargo, se requiere mucho trabajo para especificar el hardware en que se instalarán las aplicaciones y las bases de datos, lo que incluye estimaciones de la capacidad de los procesadores, dispositivos de disco y banda amplia de red en que funcionará el sistema.

En el lado de la base de datos, las relaciones normalizadas que fueron diseñadas en la fase previa de diseño lógico se implementan en el DBMS relacional que se va usar. En particular, se codifica o genera el lenguaje de definición de datos (DDL) para definir los objetos de las bases de datos, entre ellas las cláusulas SQL que definen el almacenamiento físico de tablas e índices. Se efectúan análisis preliminares de las consultas de base de datos requeridas para identificar los índices adicionales que pueden ser necesarios para lograr un desempeño aceptable de la base de datos. Un resultado esencial de esta fase es el DDL para la creación de los objetos de desarrollo de la base de datos que los diseñadores necesitan para probar las aplicaciones durante la fase de construcción siguiente. El diseño físico de una base de datos se cubre con detalle en el capítulo 8.

Construcción

Durante la fase de construcción, los desarrolladores de aplicaciones codifican y prueban unidades de programación individuales. Las unidades probadas son promovidas a un ambiente de prueba del sistema, en donde todo el sistema de aplicaciones y bases de datos se integra y prueba con minuciosidad. En la figura 5-2 se muestran los entornos que suelen usarse cuando se desarrolla, prueba e implementa un sistema de aplicaciones. Cada uno es un entorno completo de hardware y software que incluye todos los componentes necesarios para ejecutar el sistema de aplicaciones. Una vez concluida la prueba del sistema, éste se promueve a un entorno de aseguramiento de la calidad. Casi todas las organizaciones medianas y grandes tienen

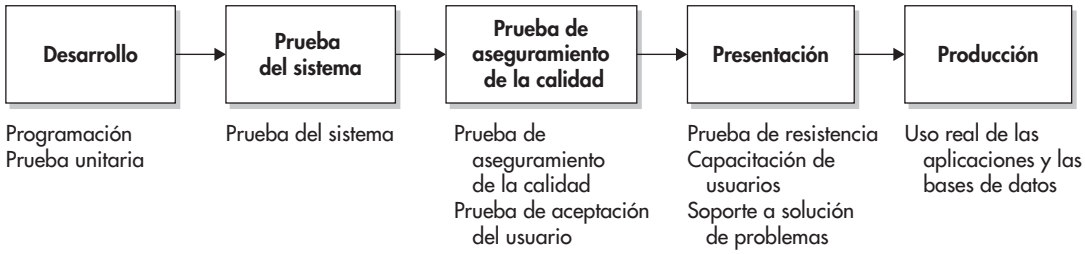


Figura 5-2 Ambientes de desarrollo de hardware/software.

un departamento de aseguramiento de la calidad que prueba el sistema de aplicaciones para asegurarse de que se apega a los requisitos mencionados. Algunas organizaciones también hacen que los usuarios de negocios prueben el sistema para corroborar que también satisface sus necesidades. Cuanto antes se encuentran errores en un sistema computacional, menos costoso resultará repararlos. Después de que aseguramiento de la calidad ha aprobado un sistema de aplicaciones, es promovido a un entorno de presentación. Éste debe imitar lo más posible el entorno de producción. En este entorno se efectúan pruebas de resistencia para asegurar que las aplicaciones y la base de datos funcionarán razonablemente cuando se desplieguen para un uso productivo real. También se suele efectuar aquí la capacitación final de los usuarios, porque será muy similar al entorno real que pronto utilizarán.

Cuando comienza la construcción, el trabajo principal del DBA ya está concluido. Sin embargo, como cada parte del sistema de aplicaciones es trasladada de un ambiente al siguiente, también deben migrarse los componentes de la base de datos que necesitan las aplicaciones. Es de esperar que se escriba una secuencia de comandos que despliegue los componentes de la base de datos para el entorno de desarrollo y que esa secuencia de comandos se vuelva a utilizar en cada entorno posterior. Sin embargo, pueden ocurrir complicaciones cuando se pretende mejorar una base de datos existente o se reemplaza un sistema de almacenamiento de datos más antiguo, porque los datos deben convertirse de estructuras de almacenamiento antiguas a nuevas. Los datos trascienden a los sistemas. Por lo tanto, la conversión de datos entre las versiones antigua y nuevas de los sistemas es trivial, y va desde simples adiciones de tablas y columnas nuevas hasta complejas conversiones que requieren cuantiosos esfuerzos de programación.

Implementación y lanzamiento

La *implementación* es el proceso de instalar los nuevos componentes del sistema de aplicaciones (programas de aplicaciones, formularios o páginas Web, informes, objetos de base de datos, etc.) en el sistema real y efectuar cualquier conversión de datos requerida. El *lanzamiento* es el proceso de ubicar grupos de usuarios de negocios en la nueva aplicación. En ocasiones

un proyecto nuevo se implementa *en frío*, lo que significa que la versión nueva es novedosa para todos los usuarios. Sin embargo, con las aplicaciones más complicadas o las destinadas a grandes cantidades de usuarios se suele aplicar una implementación *en fases* para reducir el riesgo. Las versiones antigua y nueva de la aplicación deben funcionar en paralelo durante un tiempo mientras los grupos de usuarios (divididos por ubicación del trabajo físico o por departamentos) son capacitados y trasladados a la aplicación nueva. A este método suele denominársele jocosamente *método en caliente* (porque contrasta con el método en frío).

Soporte continuo

Una vez que se ha implementado un nuevo sistema de aplicaciones y base de datos en un entorno de producción, el soporte de aplicaciones se suele traspasar a un equipo de soporte de producción. Este equipo debe estar preparado para aislar y resolver las complicaciones que surjan, como problemas de desempeño, resultados anormales o inesperados, fallas completas o las inevitables solicitudes de mejoramientos. Con estos últimos, lo mejor es clasificarlos, asignarles prioridades e incorporarlos en proyectos futuros. Sin embargo, deben repararse de inmediato los errores genuinos (también denominados *defectos* en la terminología de tecnología de la información) encontrados en una aplicación o base de datos existente. Cada defecto se vuelve un miniproyecto, al que deben aplicarse de nuevo todas las fases del CVDS. Cuando menos, debe actualizarse la documentación conforme se realizan cambios. Tal como se indicó en la figura 5-2, la etapa de presentación ofrece un lugar ideal para la validación de los errores y sus reparaciones, y permite reparar los errores en paralelo con el siguiente mejoramiento importante del sistema de aplicaciones, que tal vez ya se haya iniciado en el entorno de desarrollo.

Siempre y cuando no ocurran errores importantes durante el diseño de la base de datos, durante esta fase suele ser menor el soporte requerido. Algunas tareas necesarias serían:

- Deben aplicarse parches cuando se determina que los problemas son defectos en el software RDBMS del vendedor.
- Una afinación del rendimiento, como trasladar archivos de datos o incorporar índices, puede ser necesaria para evitar problemas de rendimiento.
- Debe vigilarse el espacio y aumentarse el almacenamiento conforme crece la base de datos.
- Algunas reparaciones de defectos de aplicaciones pueden requerir columnas nuevas o alterar las columnas existentes. Si las pruebas se efectuaron bien, no ocurren errores importantes que requieran cambios extensos en la base de datos. Se necesitan algunas modificaciones en una aplicación debido a cambios en los estatutos o los reglamentos más allá del control de la organización, y estos cambios pueden llevar a modificaciones extensas en las aplicaciones y las bases de datos.

Pregunta al experto

P: Hace poco he escuchado del proceso unificado racional (Rational Unified Process, RUP). ¿Cómo se integra esto con el CVDS?

R: El proceso unificado rational es una estructura iterativa de procesos para desarrollo de software, originalmente desarrollada por Rational Software Corporation, que se convirtió en una división de IBM en 2003. Las organizaciones que utilizan el conjunto de herramientas de Rational para el desarrollo de aplicaciones también suelen emplear estructuras de procesos de la compañía. RUP fue diseñado para que lo ajuste la organización que lo utiliza, de modo que no existen dos implementaciones iguales. A diferencia del CVDS, las iteraciones RUP están diseñadas dentro del marco conceptual. Si bien las fases de RUP (concepción, elaboración, construcción y transición) tienen nombres un tanto diferentes de las fases del CVDS, las tareas se clasifican en seis disciplinas de ingeniería que coinciden mucho con el CVDS clásico (modelado de negocios, requisitos, análisis y diseño, implementación, prueba y despliegue).

Ciclos de vida no tradicionales

Como respuesta a la idea de que los proyectos de CVDS ocupan demasiado tiempo y consumen demasiados recursos, el uso de algunos métodos no tradicionales se ha integrado en las rutinas de algunas organizaciones. Dos de los más destacados son *creación de prototipos* y *desarrollo rápido de aplicaciones (Rapid Application Development, RAD)*.

Creación de prototipos

La *creación de prototipos* conlleva el desarrollo rápido de una aplicación mediante grupos iterativos de los pasos de diseño, desarrollo e implementación como método para determinar los requisitos del usuario. Se requiere una extensa participación de los usuarios de negocios durante el proceso de desarrollo. En su forma extrema, el proceso de creación de prototipos comienza con una reunión efectuada durante un día de actividades común para revisar la iteración más reciente de la aplicación, y después el equipo de desarrollo analiza eso hasta muy avanzado el día. Luego se revisa la próxima iteración durante el siguiente día laboral.

Algunas técnicas de creación de prototipos se abren paso hasta una versión de producción de la aplicación y la base de datos. En esta variación, se incorporan en las iteraciones niveles cada vez más complejos de detalle, hasta que se convierten en aplicaciones completamente funcionales. Si se elige este método, recuerde que la creación de prototipos nunca concluye, e incluso después de la implementación y la asignación, cualquier mejoramiento futuro es considerado otro prototipo. La desventaja más común de esta técnica de implementación es el desgaste del equipo de desarrollo.

Otra variación de la creación de prototipos limita el esfuerzo a la definición de los requisitos. Una vez que se determinan los requisitos y las partes relacionadas con el usuario del diseño conceptual (es decir, las vistas de usuarios), se aplica una metodología CVDS tradicional para concluir el proyecto. IBM introdujo una versión de esta metodología llamada *Diseño conjunto de aplicaciones* (Joint Application Design, JAD), que tuvo mucho éxito en situaciones donde los requisitos del usuario no se pudieron determinar mediante técnicas más tradicionales. El mayor impedimento de esta variante de creación de prototipos es no establecer ni mantenerse a la altura de las expectativas de los patrocinadores empresariales del proyecto. El prototipo es más o menos una fachada, muy parecido a un escenario en el que, al frente, los edificios tienen un aspecto real pero no hay nada sustancial detrás de ellos. Las personas sin conocimientos técnicos no comprenden lo que se requiere para desarrollar la lógica y las estructuras de almacenamiento de datos que forman los trabajos internos de la aplicación, y se sienten decepcionadas cuando comprenden que lo que parecía un sistema de aplicaciones funcional y completo es sólo una cáscara vacía. No obstante, cuando se aplica correctamente, esta técnica alcanza un éxito notable para determinar requisitos de usuarios que describen con precisión el sistema de aplicaciones que quieren y necesitan los usuarios de negocios.

Desarrollo rápido de aplicaciones

El *desarrollo rápido de aplicaciones* (RAD) es un proceso de desarrollo de software que permite crear esquemas de aplicaciones funcionales en tan sólo 60 a 90 días. Se suelen establecer soluciones intermedias mediante la *regla 80/20*, en que se supone que 80% del trabajo requerido puede concluirse en 20% del tiempo. Por ejemplo, se puede omitir el manejo de excepciones complicadas para entregar más pronto un sistema funcional. Si se repite el proceso sobre el mismo grupo de requisitos, a final de cuentas el sistema cumple 100% de los requisitos de un modo similar a la creación de prototipos.

El RAD no es útil para controlar calendarios con presupuestos de proyectos, y en realidad requiere un administrador de proyectos con gran talento para manejar los tiempos y controlar los costos. Es más útil en situaciones en que un calendario rápido es más importante que la calidad del producto (medida en términos de apegarse a todos los requisitos conocidos).

El triángulo del proyecto

La motivación que apoya el crecimiento de los procesos de desarrollo no tradicionales es la presión que padece la administración para desarrollar mejores aplicaciones de negocios con mayor rapidez y a un menor costo. En otras palabras, se propone la entrega rápida de software de aplicaciones económico y de elevada calidad. Sin embargo, a pesar de las afirmaciones de algunas personas que venden herramientas y metodologías de desarrollo, sencillamente no es posible maximizar los tres objetivos.

En la figura 5-3 se presenta una representación gráfica del dilema mediante un *triángulo del proyecto*. Los tres puntos del triángulo representan los tres objetivos: calidad, tiempo de

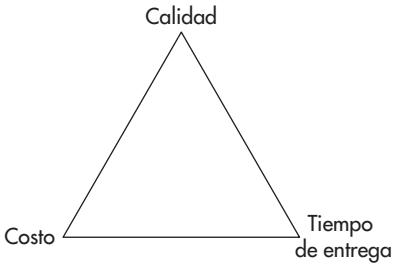


Figura 5-3 El triángulo del proyecto.

entrega y costo (también conocido como bueno, rápido y barato). Las líneas entre los puntos nos recuerdan que los objetivos se interrelacionan. En realidad, casi todos los expertos aceptan que sólo se pueden optimizar dos de los objetivos, y cuando ocurre esto, el tercer objetivo siempre se ve afectado. La regla que suele entenderse es que debe elegir dos condiciones y optimizar su proyecto de acuerdo con ellas. Asimismo, generalmente se ha demostrado que la regla se aplica a las actividades humanas y no a tecnología pura. Por ejemplo, puede crear un nuevo formato de video que genere imágenes de mayor calidad más rápido y a menor costo. Sin embargo, si inicia un proyecto para diseñar ese formato nuevo, no es posible optimizar las tareas del proyecto para los tres objetivos.

Esta regla no comenzó con la industria del software. En realidad, algunos afirman que es una antigua máxima de Hollywood acerca de la creación de películas. Aunque cada productor quiere una película de alta calidad, hecha con rapidez, y concluida dentro de un presupuesto, esto simplemente no puede lograrse. Una buena película hecha rápido no es barata. Una película concluida con rapidez y a un costo bajo no es buena. Y una película que es buena y económica no puede hacerse rápidamente. Al aplicar la analogía a los proyectos de desarrollo de aplicaciones, surgen tres opciones:

- Diseñe y desarrolle el sistema rápido y a un nivel alto, pero espere costos más elevados.
- Diseñe y desarrolle el sistema rápido y con costos reducidos, pero espere que el resultado alcance un nivel de calidad bajo.
- Diseñe y desarrolle el sistema con una calidad elevada y costos reducidos, pero espere que el proyecto tarde más.

Pruebe esto 5-1 Proyecto de tareas de administración de una base de datos

En este ejercicio Pruebe esto, asignará las tareas comunes de la administración de proyectos a las fases de un proyecto de CVDS. Es posible que deba investigar un poco para comprender los detalles de una o más tareas, pero esto mejorará su experiencia de aprendizaje.

Paso a paso

1. Haga una lista de las fases de un proyecto de CVDS.

- **a.** Planeación.
- **b.** Recopilación de requisitos.
- **c.** Diseño conceptual.
- **d.** Diseño lógico.
- **e.** Diseño físico.
- **f.** Construcción.
- **g.** Implementación y asignación.
- **h.** Soporte continuo.

2. Con el uso de lo aprendido en este capítulo, y de lo que sea capaz de hallar en otras fuentes, asigne cada una de las tareas siguientes a una de las fases del proyecto. Observe que algunas se aplican a más de una fase. Asimismo, las metodologías se suelen ajustar para que se adapten a la organización donde se aplican, de modo que no hay respuestas correctas o incorrectas absolutas para algunas de las tareas.

- **a.** Normalización.
- **b.** Adición de claves externas a la base de datos.
- **c.** Especificación de la ubicación física de los objetos de la base de datos en los medios de almacenamiento.
- **d.** Especificación del identificador único para cada relación.
- **e.** Establecimiento de la clave principal a cada tabla.
- **f.** Determinación de las vistas requeridas por los usuarios de negocios.
- **g.** Eliminación de los datos que se derivan con facilidad.
- **h.** Resolución de las relaciones varios a varios.
- **i.** Definición de las vistas de la base de datos.
- **j.** Modificación de la base de datos para cumplir los requisitos de negocios.
- **k.** Desnormalización la base de datos para mejorar el rendimiento.
- **l.** Especificación de un nombre lógico para cada entidad y atributo.
- **m.** Asignación de un nombre físico a cada tabla y columna.
- **n.** Adición de datos que se puedan derivar para mejorar el rendimiento.

(*continúa*)

o. Especificación de índices para la base de datos.

p. Traducción del modelo conceptual de datos a un modelo lógico.

q. Documentación de las reglas de negocios que no se pueden representar en el modelo de datos.

r. Identificación de los atributos requeridos por los usuarios de negocios.

s. Identificación de las relaciones entre las entidades.

t. Identificación y documentación de los requisitos de datos empresariales.

u. Aseguramiento de que se cumplan los requisitos de datos de usuarios.

v. Afinación de la base de datos para mejorar el desempeño.

w. Evaluación de las opciones disponibles de DBMS.

Resumen de Pruebe esto

En este ejercicio Pruebe esto, asignó tareas de un proyecto a las fases del CVDS por medio de la información de este capítulo, y también de una investigación independiente. Encontrará la solución del autor en el apéndice B pero, como ya se mencionó, no existe una sola solución correcta para este ejercicio.

Autoexamen Capítulo 5

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. ¿Cuáles de los siguientes elementos son parte de las fases de una metodología de ciclo de vida de desarrollo de sistemas (CVDS)?

A Diseño físico.

B Diseño lógico.

C Creación de prototipos.

D Recopilación de requisitos.

E Soporte continuo.

2. Durante la fase de requisitos de un proyecto de CVDS,

A Se descubren las vistas de usuarios.

B Se emplea un entorno de aseguramiento de la calidad.

C Es posible efectuar encuestas.

D Se suele hacer entrevistas.

E Es posible emplear observaciones.

3. Las ventajas de realizar entrevistas son

A Las entrevistas requieren menos tiempo que otros métodos.

B Se obtienen respuestas a preguntas no formuladas.

C Es posible aprender mucho de las respuestas no verbales.

D Las preguntas se presentan de manera más objetiva, en comparación con las técnicas de la encuesta.

E Las entidades se descubren con mayor facilidad.

4. Algunas ventajas de efectuar encuestas son

A Se cubre mucho terreno con rapidez.

B No se incluyen respuestas no verbales.

C Las responden casi todos los participantes.

D Es sencillo redactarlas.

E No es necesario crear prototipos de los requisitos.

5. Las ventajas de la observación son

A Siempre se observa a las personas actuando normalmente.

B Es probable que se vean muchas situaciones en que se manejan excepciones.

C Realmente se ve cómo son las cosas, y no cómo las presenta la administración o la documentación.

D El efecto Hawthorne mejora sus resultados.

E Pueden observarse eventos que nadie más describiría.

6. Las ventajas de la revisión de documentos son

A Las imágenes y los diagramas son recursos valiosos para comprender los sistemas.

B Las revisiones de documentos se hacen con relativa rapidez.

C Los documentos siempre se mantendrán actualizados.

D Los documentos siempre reflejarán las prácticas actuales.

E Los documentos suelen presentar compendios que son mejores que otras técnicas.

7. Los módulos de un programa de aplicaciones se especifican durante la fase _____ del CVDS.

8. Se suele efectuar un estudio de factibilidad durante la fase _____ de un proyecto de CVDS.

9. Ocurre una normalización durante la fase _____ de un proyecto de CVDS.

10. Se escribe DDL para definir los objetos de una base de datos durante la fase _____ de un proyecto de CVDS.

11. Las especificaciones de un programa se escriben durante la fase _____ de un proyecto de CVDS.

12. Durante la implementación y el lanzamiento,

- **A** Se ubican los usuarios en el sistema real.
- **B** Se diseñan mejoras.
- **C** Las aplicaciones antigua y nueva deben ejecutarse en paralelo.
- **D** Se realiza una prueba de aseguramiento de la calidad.
- **E** Sucede la capacitación de usuarios.

13. Durante el soporte continuo,

- **A** Se implementan de inmediato las mejoras.
- **B** El depósito para la base de datos puede requerir una expansión.
- **C** Ya no se requiere el entorno de presentación.
- **D** Ocurre la reparación de defectos.
- **E** Se aplican parches, si es necesario.

14. Cuando se crea un borrador de los requisitos, _____ puede funcionar bien.

15. El desarrollo rápido de aplicaciones crea sistemas con rapidez al omitir _____.

16. Los tres objetivos representados en el triángulo de una aplicación son _____, _____ y _____.

17. Al principio la base de datos se construye en el entorno _____.

18. La conversión de una base de datos se prueba durante la fase _____ de un proyecto de CVDS.

19. Las vistas de usuarios se analizan durante la fase _____ de un proyecto de CVDS.

20. La base de datos relacional fue inventada por _____.

Capítulo 6

Diseño de una base de datos mediante normalización

Habilidades y conceptos clave

- La necesidad de normalización.
- Aplicación del proceso de normalización.
- Desnormalización.

En este capítulo aprenderá a efectuar el diseño lógico de una base de datos, proceso denominado *normalización*. En cuanto a la comprensión de la tecnología de una base de datos relacional, éste es el tema más importante de este libro, porque la normalización le enseña el mejor modo para organizar sus datos en tablas.

La normalización es una técnica para producir un conjunto de *correlaciones* (datos representados de manera lógica en un formato bidimensional que emplee filas y columnas) que posea cierto grupo de propiedades. De capítulos anteriores, recordará que E. F. (Ted) Codd es el padre de la base de datos relacional, y que desarrolló el proceso en 1972, cuando propuso tres formas normales. El nombre fue una especie de broma política para esa época. El presidente Nixon "normalizaba" las relaciones con China, de modo que Codd pensó que si era posible normalizar las relaciones con un país, también era posible "normalizar" las relaciones entre los datos. Más tarde se añadieron formas normales adicionales, que se analizan al final del capítulo.

El proceso de normalización se presenta en la figura 6-1. En la superficie, es muy sencillo y directo, pero se requiere mucha práctica para ejecutar el proceso de manera uniforme y correcta. En resumen, se toma cualquier correlación y se elige un identificador único para la entidad que representa. Luego, mediante una serie de pasos que aplican diversas reglas, se reorganiza la correlación en formas normales cada vez más progresivas. Las definiciones de cada una de estas formas y el proceso requerido para llegar a cada una se cubren en las secciones siguientes.

En todo el proceso de normalización, cuando es posible se emplean con regularidad *los términos lógicos.* La única excepción es el término *clave principal*, que se usa en lugar de *identificador único* para uniformar con las prácticas de la industria. A los principiantes les resulta más fácil pensar en objetos físicos que se crearán en algún momento a partir del diseño lógico. Esto se debe a que aprender a considerar las bases de datos en los niveles de abstracción conceptual y lógica en lugar del nivel físico es, en realidad, una disciplina muy difícil de dominar mentalmente. Si le resulta más fácil pensar en tablas que en correlaciones, y en columnas que en atributos, necesita romper ese hábito cuanto antes. Quienes sólo piensan físicamente y al mismo tiempo intentan normalizar las tablas, más tarde encuentran dificultades, porque no existe necesariamente una correspondencia uno a uno entre las relaciones formali-

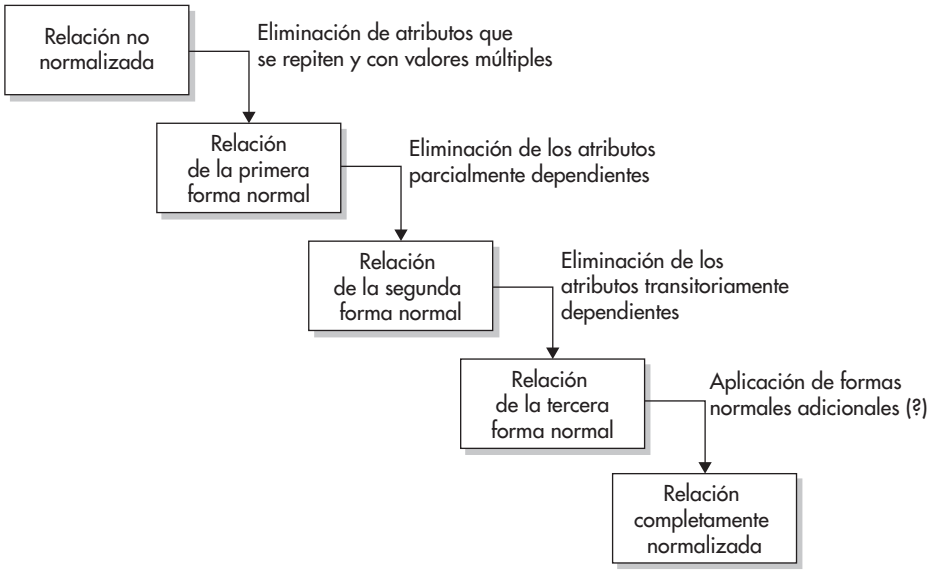


Figura 6-1 El proceso de normalización.

zadas y las tablas. En realidad, es el diseño físico de una base de datos el que transforma las correlaciones normalizadas en tablas relacionales, y existe cierta libertad para ubicar las correlaciones normalizadas en tablas físicas. La tabla siguiente le ayudará a recordar la correspondencia entre los términos lógicos y físicos:

Término lógico	Término físico
Relación o entidad	Tabla
Identificador único	Clave principal
Atributo	Columna
Tupla	Fila

NOTA

Relación fue el término original de Codd para una estructura de datos hecha de filas y columnas, y es la base del nombre *base de datos relacional*. Sin embargo, con el tiempo el término *entidad* se volvió más popular, aunque las definiciones de los dos no son exactamente iguales. Tenga cuidado de no confundir este tipo de *relación* (que corresponde a una estructura de datos) con una relación que indica la manera en que se relaciona una estructura con otra. En realidad, puede ser esta confusión la que ha hecho que las personas dejen de utilizar el término.

La necesidad de normalización

En su obra inicial, con la teoría para una base de datos relacional, Codd descubrió que las relaciones no normalizadas presentaban ciertos problemas cuando se hacía un intento por actualizar los datos en ellas. Aplicó el término *anomalías* a estos problemas. La razón por la que se normalizan las relaciones es para *eliminar* estas anomalías de los datos. Resulta esencial que se comprendan estas anomalías, porque también indican cuándo es aceptable forzar las reglas durante el diseño físico por medio de la "desnormalización" de las relaciones (lo que se cubre más adelante en el capítulo). Resulta evidente que para forzar las reglas, primero necesita comprender por qué existen.

En la figura 6-2 se presenta una factura de Industrias Acme, una empresa ficticia. La factura contiene los atributos comunes de una factura impresa de un proveedor. De manera conceptual, la factura es una vista de usuario. Se empleará el ejemplo de esta factura durante el análisis del proceso de normalización.

Anomalía de inserción

La *anomalía de inserción* se refiere a una situación en que no es posible insertar una tupla nueva en una relación debido a una dependencia artificial de otra. (Una *tupla* es un conjunto de valores de datos que forman una ocurrencia de una entidad. En una base de datos física, una tupla es una fila de datos.) El error que ha provocado la anomalía es que los atributos de dos entidades diferentes están mezclados en la misma relación. En cuanto a la figura 6-2, se observa que la identificación, el nombre y la dirección del cliente se incluyen en la vista de la factura. Si se hiciera una relación a partir de esta vista tal como está y, en algún momento, una

Industrias Acme
FACTURA

Número de cliente: 1454837
Cliente: W. Coyote
Entregas
Rocas, Son 84211
(599) 555-9345

Términos: 30 días
Método de envío: USPS

Fecha de pedido: 01/12/2008

Núm. de producto	Descripción	Cantidad	Precio unitario	Monto extendido
SPR-2290	Resortes superextensibles	2	24.00	$48.00
STR-67	Sujetadores de pies, de piel	2	2.50	$5.00
HLM-45	Casco de lujo	1	67.88	$67.88
SFR-1	Combustible sólido para cohetes	1	128,200.40	$128 200.40
ELT-1	Transmisor de ubicación de emergencia	1	79.88	**Regalo gratis**

MONTO TOTAL DEL PEDIDO: **$128 321.28**

Figura 6-2 Factura de Industrias Acme.

tabla a partir de la relación, pronto descubriría que no puede insertar un cliente nuevo en la base de datos a menos que el cliente haya comprado algo. Esto se debe a que todos los datos del cliente están incrustados en la factura.

Anomalía de eliminación

La *anomalía de eliminación* es lo opuesto de la anomalía de inserción. Se refiere a una situación en que una eliminación de datos acerca de una entidad específica provoca una pérdida no intencional de los datos que caracterizan a otra entidad. En el caso de la factura de Industrias Acme, si se elimina la última factura que pertenece a un cliente en particular, se pierden todos los datos relacionados con ese cliente. Una vez más, esto se debe a que están mezclados de manera incorrecta datos de dos entidades (clientes y facturas) en una sola relación, si la factura se ha implementado sólo como una tabla, sin aplicar el proceso de normalización a la relación.

Anomalía de actualización

La *anomalía de actualización* se refiere a una situación en que la actualización de un solo valor de datos requiere que se actualicen varias tuplas (filas) de datos. En el ejemplo de la factura, si se quiere cambiar la dirección del cliente, sería necesario modificarla en cada factura para el cliente. Esto se debe a que la dirección se guardaría de manera redundante en cada factura de ese cliente. Para empeorar las cosas, los datos redundantes ofrecen una oportunidad inapreciable de actualizar muchas copias de los datos y al mismo tiempo pasar por alto algunas, lo que genera datos no uniformes. El mantra del diseñador de base de datos experimentado es: para cada atributo, capturar una vez, guardar una vez y usar esa copia en todas partes.

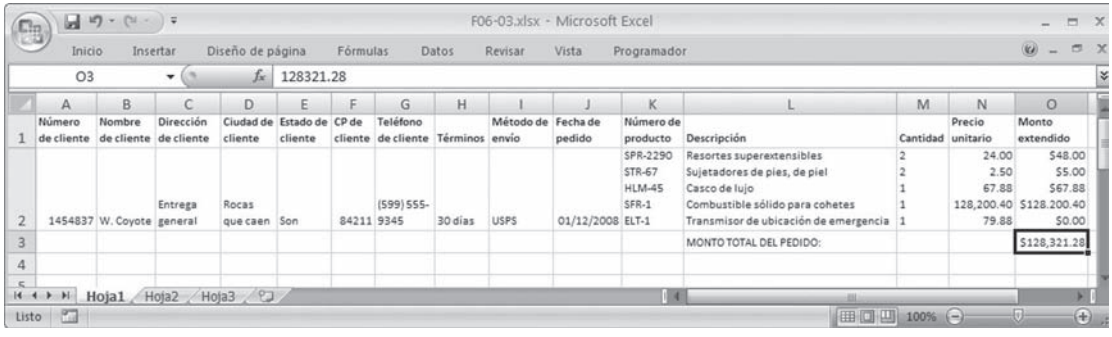
Aplicación del proceso de normalización

El proceso de normalización se aplica a cada vista de usuario recopilada durante las etapas iniciales del diseño. A algunas personas les resulta más fácil aplicar el primer paso (seleccionar una clave principal) para cada vista de usuario, y luego aplicar el paso siguiente (convertirla a la primera forma normal), etc. Otras personas prefieren tomar la primera vista de usuario y aplicarle todos los pasos de la normalización, luego otra vista de usuario, etc. Con la práctica, comprenderá cuál proceso funciona mejor para usted, pero cualquiera que elija, su método debe ser *muy* sistemático. El ejemplo que se presenta sólo tiene una vista de usuario (la factura de Industrias Acme), de modo que esto puede parecer un punto debatible, pero los dos problemas prácticos del final del capítulo contienen varias vistas de usuario cada uno, de modo que podrá intentar esto pronto. Utilizar marcadores en un pizarrón montado en la pared es útil porque puede borrar fácilmente y volver a escribir las relaciones conforme avanza.

Comience por considerar que cada vista de usuario es una relación, lo que significa que debe representarla como si fuera una tabla bidimensional. Conforme avance por el proceso de normalización, volverá a escribir las relaciones existentes y a crear algunas nuevas. Para algunas personas es útil dibujar las relaciones con tuplas (filas) de datos de ejemplo para visualizar el trabajo. Si adopta este método, no olvide que sus datos deben representar situaciones reales. Por ejemplo, es posible que pase por alto que dos clientes pueden tener exactamente el mismo nombre en el ejemplo de la factura; por esta razón, los resultados de su normalización pueden ser incorrectos. Por lo tanto, cuando utilice este método *siempre* debe pensar en la mayor cantidad posible de posibilidades. En la figura 6-3 se muestra la información del ejemplo de la factura (figura 6-2) representada en forma tabular. Sólo se presenta una factura aquí, pero se pueden llenar muchas más para mostrar ejemplos de varias facturas por cliente, varios clientes, el mismo producto en varias facturas, y demás.

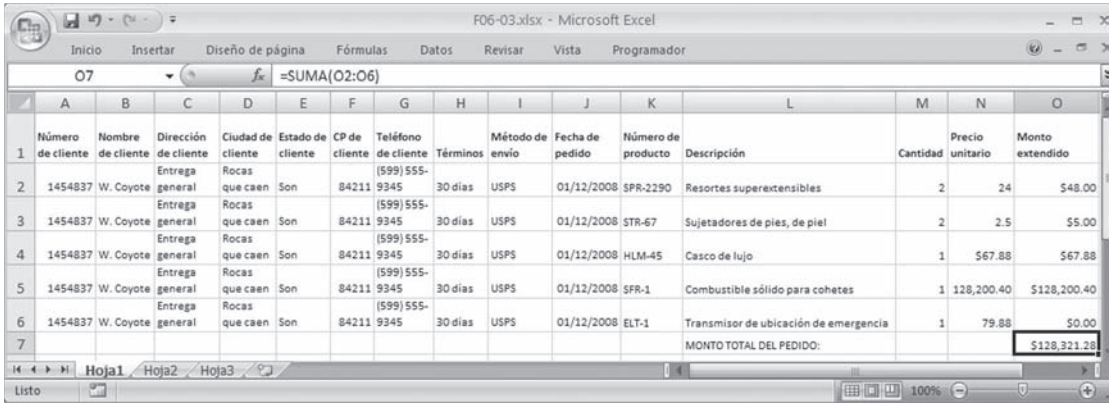
Es probable que haya observado que cada factura tiene muchos artículos de línea. Esto será información esencial al llegar a la primera forma normal. En la figura 6-3, se colocaron varios valores en las celdas para las columnas que contienen datos de los artículos de línea. A éstos se les denomina *atributos de valores múltiples* porque poseen varios valores para cuando menos algunas tuplas (filas) en la relación. Si construyera una tabla de base de datos real de esta manera, estaría limitada su capacidad para usar un lenguaje como SQL para consultar esas columnas. Por ejemplo, para hallar todos los pedidos que contienen un producto específico sería necesario que examine la sintaxis de los datos de la columna con un operador **LIKE**. Las actualizaciones también serían difíciles de manejar porque SQL no fue diseñado para manejar columnas con varios valores. Y, lo peor de todo, la eliminación de un producto de una factura requeriría una instrucción UPDATE de SQL en lugar de una DELETE, porque no sería conveniente eliminar toda la factura. Cuando reflexione sobre la primera forma normal, más adelante este capítulo, verá cómo mitigar este problema.

En la figura 6-4 se supone otro modo para organizar una relación mediante la factura presentada en la figura 6-2. Aquí los datos de la columna con valores múltiples se han colocado



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Número de cliente	Nombre de cliente	Dirección de cliente	Ciudad de cliente	Estado de cliente	CP de cliente	Teléfono de cliente	Términos	Método de envío	Fecha de pedido	Número de producto	Descripción	Cantidad	Precio unitario	Monto extendido
											SPR-2290	Resortes superextensibles	2	24.00	$48.00
											STR-67	Sujetadores de pies, de piel	2	2.50	$5.00
											HLM-45	Casco de lujo	1	67.88	$67.88
			Entrega	Rocas			(599) 555-				SFR-1	Combustible sólido para cohetes	1	128,200.40	$128,200.40
2	1454837	W. Coyote	general	que caen	Son	84211	9345	30 días	USPS	01/12/2008	ELT-1	Transmisor de ubicación de emergencia	1	79.88	$0.00
3												MONTO TOTAL DEL PEDIDO:			$128,321.28
4															

Figura 6-3 Factura de Industrias ACME representada en forma tabular.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Número de cliente	Nombre de cliente	Dirección de cliente	Ciudad de cliente	Estado de cliente	CP de cliente	Teléfono de cliente	Términos	Método de envío	Fecha de pedido	Número de producto	Descripción	Cantidad	Precio unitario	Monto extendido
2	1454837	W. Coyote	Entrega general	Rocas que caen	Son	84211	(599) 555-9345	30 días	USPS	01/12/2008	SPR-2290	Resortes superextensibles	2	24	$48.00
3	1454837	W. Coyote	Entrega general	Rocas que caen	Son	84211	(599) 555-9345	30 días	USPS	01/12/2008	STR-67	Sujetadores de pies, de piel	2	2.5	$5.00
4	1454837	W. Coyote	Entrega general	Rocas que caen	Son	84211	(599) 555-9345	30 días	USPS	01/12/2008	HLM-45	Casco de lujo	1	$67.88	$67.88
5	1454837	W. Coyote	Entrega general	Rocas que caen	Son	84211	(599) 555-9345	30 días	USPS	01/12/2008	SFR-1	Combustible sólido para cohetes	1	128,200.40	$128,200.40
6	1454837	W. Coyote	Entrega general	Rocas que caen	Son	84211	(599) 555-9345	30 días	USPS	01/12/2008	ELT-1	Transmisor de ubicación de emergencia	1	79.88	$0.00
7												MONTO TOTAL DEL PEDIDO:			$128,321.28

Figura 6-4 Factura de Industrias ACME representada sin atributos con valores múltiples.

en filas separadas y los datos de las otras columnas se han repetido para que coincidan. El problema obvio aquí son los datos repetidos. Por ejemplo, el nombre y la dirección del cliente se repiten en cada artículo de línea de la factura, lo que no sólo desperdicia recursos, sino también lo expone a inconsistencias cuando los datos no se conservan de manera uniforme (por ejemplo, si actualiza la ciudad para un artículo de línea, pero no para los demás).

Volver a escribir las vistas de usuarios como tablas con datos representativos es un proceso tedioso que consume tiempo. Por esta razón, usted simplemente escribe los atributos como una lista y los visualiza en su mente como las tablas bidimensionales en que se convertirán en algún momento. Esto requiere práctica y cierta capacitación mental, pero una vez que lo domina, le parecerá mucho más rápido visualizar los datos en lugar de escribir ejemplos pormenorizados. La siguiente es la lista para el ejemplo de la factura de la figura 6-2:

```
FACTURA: Número de cliente, Nombre de cliente, Dirección de cliente,
         Ciudad de cliente, Estado de cliente, CP de cliente,
         Teléfono de cliente, Términos, Método de envío,
         Fecha de pedido, Número de producto, Descripción, Cantidad,
         Precio unitario, Monto extendido, Monto total del pedido
```

Para mayor claridad, se ha agregado un nombre a la relación, y se ha puesto en mayúsculas y separado de los atributos con un signo de dos puntos. Ésta es la convención que se utilizará para el resto del capítulo. Sin embargo, si le funciona mejor otra técnica, no dude en utilizarla. Lo mejor de todo es que no importa cuál representación utilice (figura 6-3, figura 6-4, o la lista anterior); si aplica adecuadamente el proceso de normalización y sus reglas, generará un diseño de base de datos equivalente.

Elección de una clave principal

Cuando aplica normalización, considera que cada vista de usuario es como una relación. En otras palabras, conceptualiza cada vista de usuario como si ya estuviera implementada en una tabla bidimensional. El primer paso de la normalización consiste en elegir una clave principal entre los identificadores únicos que encuentra en la relación.

Recuerde que un *identificador único* es un conjunto de uno o más atributos que identifican de manera inequívoca cada ocurrencia de una relación. En muchos casos, puede encontrar un solo atributo. En nuestro ejemplo, el número de cliente de la factura identifica de manera inequívoca los datos del cliente dentro de ésta, pero debido a que un cliente puede tener varias facturas, no es adecuado como identificador de la factura completa.

Cuando no encuentre un atributo que sirva como identificador único, puede unir varios atributos para formarlo. Verá cómo sucede esto con la factura de ejemplo cuando se dividen los artículos de línea de la factura mientras la normaliza. Es muy importante comprender que cuando un identificador único está formado por varios atributos, éstos no se pueden combinar; todavía existen como atributos independientes y se convertirán en columnas individuales en las tablas creadas a partir de nuestras relaciones normalizadas.

En unos cuantos casos, ningún grupo de atributos en una relación se puede emplear razonablemente como identificador único. (Encontrará que muchas personas emplean indistintamente los términos *identificador* y *clave*.) Cuando ocurre esto, debe inventar un identificador único, a menudo con valores asignados en forma secuencial o aleatoria conforme agrega ocurrencias de la entidad a la base de datos. Esta técnica (algunos la podrían llamar "acto de desesperación") es el origen de identificadores únicos como números de seguridad social, Id de empleado y números de licencia de conducir. Los identificadores únicos que tienen un significado real son identificadores *naturales*, y los que no lo poseen (entre los que se incluyen los que debemos inventar) son identificadores *sustitutos* o *artificiales*. Al parecer, en el ejemplo de la factura no existe un identificador único natural para la relación. Puede tratar de usar el número de cliente combinado con la fecha de pedido, pero si un cliente tiene dos facturas en la misma fecha, esto no sería único. Por lo tanto, sería mucho mejor inventar un identificador, como un número de factura.

Cada vez que selecciona un identificador único para una relación, debe tener la *certeza* de que el identificador *siempre* será único. Si llega a existir cuando menos *un* caso en que no sea único, no puede utilizarlo. Por ejemplo, los nombres de personas son inconvenientes como identificador único. Tal vez nunca conozca a alguien exactamente con su nombre, pero en el mundo hay personas con un nombre idéntico al suyo. Como ejemplo del daño que provoca un identificador único mal seleccionado, considere el caso de cuando el gobierno brasileño comenzó a registrar a los votantes en 1994, para reducir los fraudes electorales. Se eligieron como identificador único el nombre del padre, el nombre de la madre y la fecha de nacimiento. Por desgracia, esta combinación sólo es única para hermanos nacidos en fechas *diferentes*; como resultado, cuando hermanos nacidos en la misma fecha (mellizos, triates y demás)

trataron de registrarse para votar, se permitió el registro al primero que apareció y los demás fueron descartados. ¿Parece imposible? No, esto sucedió en la realidad. Y para empeorar las cosas, en Brasil es *obligatorio* que los ciudadanos voten; para conseguir un empleo deben comprobar que lo hicieron. Alguien debió dedicar más tiempo a analizar la singularidad del identificador "único" elegido.

En ocasiones una relación tiene más de un identificador único (clave) posible. Cuando ocurre esto, a cada posibilidad se le denomina *candidata*. Una vez que ha identificado todas las candidatas posibles para una relación, debe elegir una de ellas para que sea la clave principal de la relación. La elección de una clave principal es *esencial* para el proceso de normalización, porque todas las reglas de normalización hacen referencia a la clave principal. Éstos son los criterios para elegir la clave principal entre las candidatas (en orden de precedencia, la más importante primero):

- *Si sólo existe una candidata, selecciónela.*
- *Elija la candidata con menor probabilidad de que cambie su valor*. La modificación de los valores de clave principal una vez que se guardan datos en las tablas es un asunto complicado porque la clave principal puede aparecer como clave externa en muchas otras tablas. Por cierto, casi siempre es menos probable que cambien las claves sustitutas en comparación con las naturales.
- *Seleccione la candidata más sencilla*. La candidata con la menor cantidad de atributos se considera la más sencilla.
- *Escoja la candidata más breve*. Esto es sólo por razones de eficiencia. No obstante, cuando una clave principal puede aparecer en muchas tablas como clave externa, por lo general vale la pena ahorrar un poco de espacio con cada una.

En el ejemplo de la factura, se ha optado por agregar una clave principal sustituta llamada Número de factura. Esto proporciona una clave principal sencilla para las facturas de Industrias ACME cuya unicidad está garantizada, porque es posible hacer que el DBMS asigne automáticamente números secuenciales a las facturas nuevas conforme son generadas. Al mismo tiempo, es probable que esto satisfaga a los contadores de Acme, porque les ofrece un número sencillo para rastrear las facturas.

Se pueden emplear muchas convenciones para identificar la clave principal cuando escribe el contenido de las relaciones. Emplear sólo letras mayúsculas provoca confusión porque en la actualidad se usan muchas siglas que no siempre pueden ser una clave principal. Además, subrayar o poner en negritas los nombres de los atributos puede ser complicado porque no siempre se muestran de la misma manera. Por lo tanto, en este libro se utilizan las letras *CP* entre paréntesis después de los nombres de atributos de la clave principal. La reelaboración de la relación de la factura en forma de lista con la clave principal añadida genera lo siguiente:

```
FACTURA: Número de factura (CP), Número de cliente, Nombre de cliente,
         Dirección de cliente, Ciudad de cliente, Estado de cliente,
         CP de cliente, Teléfono de cliente, Términos,
         Método de envío, Fecha de pedido, Número de producto,
         Descripción, Cantidad, Precio unitario,
         Monto extendido, Monto total del pedido
```

Primera forma normal: eliminación de los datos repetidos

Se dice que una relación está en la *primera forma normal* cuando no contiene atributos con valores múltiples; es decir, cada intersección de una fila y una columna en la relación debe contener *cuando mucho* un valor de datos (decir "cuando mucho" permite valores omitidos o nulos). En ocasiones, encontrará un grupo de atributos que se repiten juntos, como los artículos de línea de la factura. Cada atributo del grupo tiene valores múltiples, pero varios atributos están tan estrechamente relacionados que sus valores se repiten juntos. A esto se le denomina *grupo repetido*, pero en realidad es sólo un caso especial del problema de un atributo con varios valores.

Por convención, conviene poner entre paréntesis los grupos repetidos y los atributos con valores múltiples. Al volver a redactar la factura de esta manera para mostrar los datos de los artículos de línea como un grupo repetido, se obtiene esto:

```
FACTURA: Número de factura (CP), Número de cliente, Nombre de cliente,
         Dirección de cliente, Ciudad de cliente, Estado de cliente,
         CP de cliente, Teléfono de cliente, Términos,
         Método de envío, Fecha de pedido, (Número de producto,
         Descripción, Cantidad, Precio unitario,
         Monto extendido), Monto total del pedido
```

Es esencial que comprenda que, aunque sabe que Industrias ACME tiene muchos clientes, sólo existe uno para una factura específica, de modo que los datos del cliente en la factura *no* sean un grupo repetido. Es posible que haya observado que los datos de un cliente específico se repiten en cada factura para ese cliente, pero este problema se abordará con la tercera forma normal. Debido a que hay un solo cliente por factura, el problema no se resuelve cuando se transforma la relación a la primera forma normal.

Para transformar las relaciones no normalizadas a la primera forma normal, debe mover a relaciones nuevas los atributos con valores múltiples y los grupos repetidos. Debido a que un grupo repetido es un conjunto de atributos que se repiten *juntos*, todos los atributos de un grupo repetido deben moverse a la misma relación nueva. Sin embargo, un atributo de varios valores (atributos individuales que tienen varios valores) debe trasladarse a su propia relación nueva en lugar de combinarse con otros atributos de valores múltiples en la relación nueva. Como comprobará después, esta técnica evita problemas con la cuarta forma normal.

He aquí el procedimiento para trasladar un atributo con varios valores o un grupo repetido a una relación nueva:

1. Cree una relación nueva con un nombre significativo. Suele ser sensato incluir todo o una parte del nombre de la relación original en el nombre de la nueva relación.
2. Copie la clave principal de la relación original a la nueva. Los datos dependen de esta clave principal en la relación original, de modo que todavía deben depender de esta clave en la relación nueva. Esta clave principal copiada se convierte ahora en una *clave externa* para la relación original. Conforme aplica una normalización al diseño de una base de datos, recuerde siempre que en algún momento tendrá que escribir SQL para reproducir la vista de usuario original a partir de la cual comenzó. De modo que las claves externas utilizadas para volver a combinar las cosas son punto menos que esenciales.
3. Mueva el grupo repetido o el atributo con varios valores a la relación nueva. (Se utiliza la palabra *mueva* porque estos atributos son *removidos* o eliminados de la relación original.)
4. Vuelva única la clave principal (tal como la copió de la relación original) al agregarle atributos del grupo repetido. Si mueve un atributo con varios valores, que es básicamente un grupo repetido de un solo atributo, éste se añade a la clave principal. Esto parecerá extraño al principio, pero los atributos de clave principal que copió de la tabla original son una *clave externa* en la relación nueva. Es muy normal que una parte de una clave principal también sea una clave externa. Un detalle adicional: es perfectamente aceptable tener una relación en que todos los atributos son parte de la clave principal (es decir, en que no hay atributos "que no son de clave"). Esto es relativamente común en las tablas de intersección.
5. Como opción, puede optar por reemplazar la clave principal con un solo atributo de clave sustituta. Si hace eso, debe conservar los atributos que integran la clave principal natural formada en los pasos 2 y 4.

Para el ejemplo de la factura de Industrias ACME, éste es el resultado de convertir la relación original a la primera forma normal:

```
FACTURA: Número de factura (CP), Número de cliente, Nombre de cliente,
         Dirección de cliente, Ciudad de cliente, Estado de cliente,
         CP de cliente, Teléfono de cliente, Términos,
         Método de envío, Fecha de pedido, Monto total del pedido

ELEMENTO DE LÍNEA DE FACTURA: Número de factura (CP), Número de
         producto (CP), Descripción, Cantidad, Precio unitario, Monto
         extendido
```

Observe lo siguiente:

- El atributo Número de factura fue copiado de FACTURA para ELEMENTO DE LÍNEA DE FACTURA y se agregó Número de producto para formar la clave principal de la relación ELEMENTO DE LÍNEA DE FACTURA.
- El grupo repetido completo (Número de producto, Descripción, Cantidad, Precio unitario y Monto extendido) fue eliminado de la relación FACTURA.
- Número de factura todavía es la clave principal de FACTURA, y ahora también funciona como una clave externa en ELEMENTO DE LÍNEA DE FACTURA, además de ser *parte* de la clave principal de ELEMENTO DE LÍNEA DE FACTURA.
- No existen grupos repetidos ni atributos de valores múltiples en las relaciones; por lo tanto, están en la primera forma normal.

Tome en cuenta una consecuencia interesante de preparar una clave principal natural para la relación ELEMENTO DE LÍNEA DE FACTURA: no puede poner el mismo producto en una factura específica más de una vez. Esto puede ser conveniente, pero también puede limitar a Industrias ACME. Debe comprender sus reglas de negocios para entenderlo. Si Industrias ACME quiere la opción de colocar varios artículos de línea en la misma factura para el mismo producto (tal vez con diferentes precios), en lugar de eso debe crear una clave sustituta. Asimismo, hay quienes creen que las claves principales formadas por varios atributos no son convenientes, además de que algunos productos de software simplemente no las permiten. La opción consiste en formar una clave principal sustituta para la relación ELEMENTO DE LÍNEA DE FACTURA. Si opta por esto, la relación se vuelve a escribir de este modo:

```
ELEMENTO DE LÍNEA DE FACTURA: ID de elemento de línea de factura (CP),
                              Número de factura, Número de producto,
                              Descripción, Cantidad,
                              Precio unitario, Monto extendido
```

Se va a utilizar la forma anterior (la que tiene la clave principal compuesta formada por Número de factura y Número de producto, también llamada *clave natural*) mientras continúa la normalización.

Segunda forma normal: eliminación de las dependencias parciales

Antes de que explore la segunda forma normal, debe comprender el concepto de *dependencia funcional*. Para esta definición, se utilizarán dos atributos arbitrarios, denominados “A” y “B”. El atributo B es *funcionalmente dependiente* del A si en cualquier momento no más de un valor del B se asocia con un valor específico del atributo A. Si se pregunta en cuál planeta viví

antes de éste, intentaré ofrecer una definición más comprensible. En primer lugar, suponga que el atributo B es funcionalmente dependiente del A; es como decir que el atributo A *determina* el B, o que el A es un *determinante* (identificador único) del atributo B. En segundo lugar, es momento de regresar a las relaciones de la primera forma normal del ejemplo de Industrias ACME:

```
FACTURA: Número de factura (CP), Número de cliente, Nombre de cliente,
         Dirección de cliente, Ciudad de cliente, Estado de cliente,
         CP de cliente, Teléfono de cliente, Términos,
         Método de envío, Fecha de pedido, Monto total del pedido

ELEMENTO DE LÍNEA DE FACTURA: Número de factura (CP), Número de producto
         (CP), Descripción, Cantidad, Precio unitario, Monto extendido
```

En la relación FACTURA, es fácil apreciar que Número de cliente es funcionalmente dependiente de Número de factura, porque en cualquier momento sólo puede haber un valor de Número de cliente asociado con un valor determinado de Número de factura. El simple hecho de que Número de factura identifique de manera única a Número de cliente en esta relación significa que, a su vez, Número de cliente es *funcionalmente dependiente* de Número de factura.

En la relación ELEMENTO DE LÍNEA DE FACTURA, también se puede decir que Descripción es funcionalmente dependiente de Número de producto porque, en cualquier momento, sólo existe un valor de Descripción asociado con Número de producto. Sin embargo, el hecho de que Número de producto sea sólo parte de la clave de ELEMENTO DE LÍNEA DE FACTURA es el verdadero problema atendido por la segunda forma normal.

Se dice que una relación está en la *segunda forma normal* si cumple con los dos criterios siguientes:

- La relación está en la primera forma normal.
- Todos los atributos que no son una clave son funcionalmente dependientes de la clave principal *completa*.

Al analizar otra vez Descripción debe resultar fácil ver que Número de producto *por sí solo* determina el valor. Dicho de otro modo, si el mismo producto aparece como un artículo de línea en muchas facturas diferentes, Descripción es igual *sin tomar en cuenta* Número de factura. O se puede decir que Descripción es funcionalmente dependiente sólo de *una parte* de la clave principal, lo que significa que sólo depende de Número de producto y no de la *combinación* de Número de factura y Número de producto.

Ahora también debe ser evidente que la segunda forma normal sólo se aplica a las relaciones donde se han unido claves principales (es decir, las formadas por varios atributos). Si

una clave principal está formada sólo por un atributo único, como ocurre con la versión de la primera forma normal de la relación FACTURA, y la clave principal es indivisible (es decir, que no tiene partes secundarias que tengan sentido por sí mismas), como deben serlo todos los atributos, no es posible que nada dependa de *una parte* de la clave principal. De esto se infiere que cualquier relación de la primera forma normal que tiene sólo un atributo simple para su clave principal está *automáticamente* en la segunda forma normal.

No obstante, al analizar la relación ELEMENTO DE LÍNEA DE FACTURA, las violaciones a la segunda forma normal se aprecian fácilmente: Descripción y Precio unitario dependen sólo de Número de producto, en lugar de hacerlo de la *combinación* de Número de factura y Número de producto. Pero debe detenerse a reflexionar: ¿qué pasa con los cambios en el precio? Si Acme decide modificar sus precios, ¿querría que ese cambio fuera retroactivo para cada factura que haya creado? Después de todo, una factura es un registro oficial que debe conservar durante varios años, de acuerdo con la legislación fiscal de varios países. Se trata de un dilema común con los atributos que cambian rápido, como los precios. Debe tener la capacidad de recordar el precio en cualquier momento o debe guardar el precio con la factura para reproducirla cuando se requiera (es decir, cuando lo visiten los auditores fiscales).

Para mayor sencillez, almacenemos el precio en dos lugares: uno sería el precio de venta actual y, el otro, el precio cuando se hizo una venta. Como la última opción es una instantánea de un momento específico que no va a cambiar, no hay anormalidades en este almacenamiento aparentemente redundante. Una opción hubiera sido guardar un historial de precios determinado por fecha en algún lugar donde se puede emplear para reconstruir el precio correcto para cualquier factura. Ésa es una alternativa práctica aquí, pero nunca se podría hacer con transacciones bursátiles o de la bolsa de valores, por ejemplo. Lo importante es que aunque el precio de venta *parece* redundante, no surgen *anomalías* para el atributo adicional, de modo que no hay ningún daño. Observe que los nombres de atributos se ajustaron para que su significado sea absolutamente claro.

Una vez que encuentra una violación a la segunda forma normal, la solución consiste en mover cualquier atributo que sea parcialmente dependiente de una relación nueva donde dependa de la clave *completa*, y no de *una parte* de la clave. Éste es el ejemplo de la factura reescrito en la segunda forma normal:

```
FACTURA: Número de factura (CP), Número de cliente, Nombre de cliente,
         Dirección de cliente, Ciudad de cliente, Estado de cliente,
         CP de cliente, Teléfono de cliente, Términos,
         Método de envío, Fecha de pedido, Monto total del pedido

ELEMENTO DE LÍNEA DE FACTURA: Número de factura (CP), Número de producto
         (CP), Cantidad, Precio unitario de venta, Monto extendido

PRODUCTO: Número de producto (CP), Descripción,
          Precio unitario de lista
```

La mejora a partir de la solución de la primera forma normal es que el mantenimiento de Descripción ahora no tiene anomalías. Puede incorporar un producto nuevo independientemente de la existencia de una factura para el producto. Si quiere cambiar Descripción, puede hacerlo con sólo modificar un valor en una fila de datos. Asimismo, si por alguna razón se eliminara de la base de datos la última factura de un producto específico, no perderá su descripción (todavía estará en una fila en la relación PRODUCTO). Recuerde *siempre* que la razón para normalizar es eliminar estas anomalías.

Tercera forma normal: eliminación de las dependencias transitorias

Para comprender la tercera forma normal, primero debe entender una dependencia transitoria. Se dice que un atributo que depende de otro que no es la clave principal de la relación es *transitoriamente dependiente*. Al analizar la relación FACTURA en la segunda forma normal, se observa claramente que Nombre de cliente depende de Número de factura (cada Número de factura sólo tiene un valor de Nombre de cliente asociado); al mismo tiempo, Nombre de cliente también depende de Número de cliente. De igual manera, se puede decir lo mismo del resto de los atributos del cliente. El problema aquí es que los atributos de otra entidad (Cliente) se han incluido en la relación FACTURA.

Se dice que una relación está en la *tercera forma normal* si cumple los dos criterios siguientes:

- La relación está en la segunda forma normal.
- No existe una dependencia transitoria (es decir, todos los atributos que no son claves dependen *sólo* de la clave principal).

Para transformar una relación de la segunda forma normal a la tercera forma, simplemente traslade los atributos dependientes a relaciones donde dependan sólo de la clave principal. Tenga cuidado de dejar como clave externa el atributo del que dependen en la relación original. La necesitará para reconstruir la vista de usuario original mediante una combinación.

Si se ha preguntado sobre los atributos que se calculan fácilmente como Monto extendido (importe) en la relación ELEMENTO DE LÍNEA DE FACTURA, en realidad la tercera forma normal es la que los prohíbe, pero se requiere una interpretación sutil de la regla. Debido a que Monto extendido se calcula al multiplicar Precio unitario de venta × Cantidad, se deduce que Monto extendido está *determinado* por la combinación de Precio unitario de venta y Cantidad y, por lo tanto, *depende transitoriamente* de estos dos atributos. Así, la tercera forma normal le indica que elimine los atributos que se calculan con facilidad. Y en este caso, simplemente se retiran. Mediante una lógica similar, también puede eliminar Monto total del pedido de la relación FACTURA porque puede sumar la relación ELEMENTO DE LÍNEA DE

FACTURA para reproducir el valor. Un buen diseñador incluirá una nota en la documentación que especifique la fórmula del atributo calculado, de modo que se pueda reproducir su valor cuando sea necesario. Otra opción muy eficaz consiste en escribir el SQL que reproduce las vistas originales una vez concluido un proceso de normalización. Es un modo excelente para *probar* su normalización porque puede emplear el SQL para *demostrar* que las vistas de usuario originales se reproducen con facilidad.

He aquí los datos de la factura de Industrias ACME reescritos en la tercera forma normal:

```
FACTURA: Número de factura (CP), Número de cliente, Términos,
         Método de envío, Fecha de pedido

ELEMENTO DE LÍNEA DE FACTURA: Número de factura (CP), Número de producto
         (CP), Cantidad, Precio unitario de venta

PRODUCTO: Número de producto (CP), Descripción,
          Precio unitario de lista

CLIENTE: Número de cliente (CP), Nombre de cliente,
         Dirección de cliente, Ciudad de cliente, Estado de cliente,
         CP de cliente, Teléfono de cliente
```

Pregunta al experto

P: En la entidad CLIENTE que acaba de ejemplificar, ¿no son Ciudad de cliente y Estado de cliente transitoriamente dependientes de CP de cliente?

R: En realidad no. Aun en los casos en que siempre tiene el código postal *completo* de nueve dígitos del Servicio Postal de Estados Unidos, nada en absoluto garantiza que CP de cliente siempre contendrá sólo una ciudad, un condado y un estado. Sí, el Servicio Postal publica una lista de códigos postales que ofrece ciudad, condado y estado para cada código postal, pero sólo señala la ubicación de la oficina postal que atiende ese código; no indica que todas las direcciones dentro de ese código postal estén en la ciudad, el condado y el estado listados. En el pasado, algunos códigos postales en Estados Unidos han traspasado las fronteras estatales. Además, existen miles de ejemplos de ciudades y pueblos diferentes que comparten códigos postales. Tampoco es posible emplear los códigos postales para determinar el condado dentro del estado: casi 20% de los códigos postales de cinco dígitos contienen parte de más de un condado. Cuando deduzca cosas debe tener cuidado. El Servicio Postal será el primero en decirle que no es su responsabilidad alinear su sistema de zonificación con las fronteras políticas. El único modo 100% confiable de asignar ciudad, condado y estado a una dirección en Estados Unidos es utilizar la dirección completa con la calle en una tabla de códigos postales que incluya los nombres de las calles y los rangos de la numeración de edificios que se aplican a ese código postal.

Por lo tanto, ¿debe formar una relación CP de cliente y normalizar Ciudad de cliente y Estado de cliente de todas sus direcciones? ¿O eso se consideraría un exceso de diseño? La pregunta obtiene su respuesta al regresar a las anomalías, porque por principio de cuentas, la razón específica para normalizar sus datos es impedir las anomalías de inserción, actualización y eliminación:

- Si se forma una ciudad nueva, ¿necesita agregarla a la base de datos, aunque no tenga clientes que vivan en ella? (Ésta es una anomalía de inserción.)
- Si se disuelve una ciudad, ¿tiene la necesidad de eliminar su información sin perder otros datos? (Ésta es una anomalía de eliminación.)
- Si una ciudad cambia de nombre (no es común, pero ha ocurrido), ¿es complicado que encuentre todos los clientes de esa ciudad y modificar sus direcciones de acuerdo con eso?

Si respondió afirmativamente a alguna de las preguntas, debe normalizar los atributos Ciudad de cliente y Estado de cliente en una tabla con una clave principal CP de cliente. (Observe que los nombres de ciudad y estado asignados serán los de la oficina postal que atiende ese código postal, que son los que prefiere esa oficina, pero tal vez no sean los que eligen quienes reciben el correo.) En realidad, puede adquirir los datos de códigos postales con regularidad del Servicio Postal de Estados Unidos u otras fuentes, o puede suscribirse a un servicio de aclaración de direcciones que estandariza las direcciones y ofrece códigos postales exactos de cada una. Además, si conserva otros datos por código postal, como tarifas de envío, tiene mayor razón para normalizarlos. Pero en caso contrario, el ejemplo del código postal es una valiosa lección que explica las razones por las que normalizamos (o no) y cuándo tal vez no sea importante hacerlo.

Otro argumento para no normalizar los datos de CP de cliente es que los datos no son estables. Una oficina postal suele agregar y dividir constantemente códigos postales, y cuando las ciudades adquieren territorio nuevo, puede cambiar la lista de códigos postales para la ciudad. En todo momento, debe prevalecer el sentido común.

NOTA

He aquí un modo fácil de recordar las regla de la primera, segunda y tercera formas normales: en una relación en la tercera forma normal, cada atributo que no es una clave debe depender de la clave, sólo de la clave, y nada más que de la clave, de modo que ¡sálvame, Codd!

Más allá de la tercera forma normal

Desde la introducción original de la normalización, diversos autores han ofrecido versiones avanzadas. La tercera forma normal cubrirá mucho más de 90% de los casos que verá en los sistemas de información de negocios, y es considerada la “regla de oro” en los sistemas empresariales. Una vez que haya dominado la tercera forma normal, vale la pena que conozca formas normales adicionales.

Forma normal de Boyce-Codd

La forma normal de Boyce-Codd (Boyce-Codd Normal Form, BCNF) es una versión más sólida de la tercera forma normal. Resuelve anomalías que ocurren cuando un atributo que no es una clave es un *determinante* de un atributo que resulta parte de la clave principal (por ejemplo, cuando un atributo que es parte de la clave principal resulta funcionalmente dependiente de un atributo que no es una clave).

Por ejemplo, suponga que Industrias ACME asigna varios especialistas en soporte de producto a cada cliente, y cada especialista maneja sólo una línea de productos específica. A continuación aparece una relación que asignan los especialistas a los clientes. En realidad pueden utilizarse ID de cliente e ID de especialista de soporte (Empleado) en lugar de los nombres del cliente y de especialista en soporte, pero sus nombres se emplean aquí para ilustrar mejor el problema.

Cliente	Línea de producto	Especialista de soporte
W. Coyote	Resortes	R.E. Cortés
W. Coyote	Sujetadores	B. Barros
W. Coyote	Cascos	C. Buendía
W. Coyote	Cohetes	R. Gutiérrez
Ejército	Cohetes	R. Gutiérrez
S. González	Resortes	R.E. Cortés
S. González	Sujetadores	B. Barros
S. González	Cohetes	E. Juárez
L. Abundiz	Cascos	S.D. Olmos

En este ejemplo, debe unir los atributos Cliente y Línea de productos para formar una clave principal. Sin embargo, debido a que un especialista en soporte específico sólo apoya una línea de productos, también es cierto que el atributo Especialista de soporte determina el atributo Línea de producto. Si ha elegido una clave principal sustituta en lugar de combinar Cliente y Línea de producto para la clave principal, sería obvio que se presenta una violación de la tercera forma normal; un atributo que no es una clave determina otro atributo que no es una clave (en este caso, Especialista de soporte determina Línea de producto). Sin embargo, encubrió el error de normalización al hacer a Línea de producto parte de la clave principal. Por esta razón se considera que la BCNF es una versión más sólida de la tercera forma normal.

La BCNF tiene dos requisitos:

- La relación debe estar en la tercera forma normal.
- No existen determinantes que no sean la clave principal o una clave candidata para la tabla. Es decir, un atributo que no es una clave no identifica (determina) de manera única cualquier otro atributo, incluido uno que participa en la clave principal.

La solución consiste en dividir el determinante no deseado hacia una tabla distinta, igual que se hace con una violación de la tercera forma normal. La versión BCNF de esta relación se presenta aquí:

```
ASIGNACIÓN DE ESPECIALISTA DE SOPORTE: ID DE CLIENTE (CP),
                                       ID DE ESPECIALISTA DE SOPORTE

ESPECIALIDAD DE ESPECIALISTA DE SOPORTE: ID DE ESPECIALISTA DE SOPORTE
                                         (CP), LÍNEA DE PRODUCTO
```

En forma tabular, las relaciones y los datos se ven así (de nuevo, los nombres han sido sustituidos por las identificaciones para facilitar la visualización de los datos):

Cliente	Especialista de soporte
W. Coyote	R.E. Cortés
W. Coyote	B. Barros
W. Coyote	C. Buendía
W. Coyote	R. Gutiérrez
Ejército	R. Gutiérrez
S. González	R.E. Cortés
S. González	B. Barros
S. González	E. Juárez
L. Abundiz	S.D. Olmos

Especialista de soporte	Línea de producto
B. Barros	Sujetadores
C. Buendía	Cascos
E. Juárez	Cohetes
R.E. Cortés	Resortes
R. Gutiérrez	Cohetes
S.D. Olmos	Cascos

Cuarta forma normal

Una vez en la BCNF, los problemas de normalización restantes se refieren casi exclusivamente a relaciones en donde cada atributo es parte de la clave principal. Surge una anomalía de ese tipo cuando se incluyen en la misma relación dos o más atributos con varios valores. Por ejemplo, suponga que quiere rastrear las habilidades de oficina y de idiomas de los empleados. Prepararía una relación como ésta:

ID de empleado	Habilidad de oficina	Habilidad de idiomas
1001	Escribe 40 ppm	Español
1001	Usa todos los dedos	Francés
1002	Hojas de cálculo	Español
1002	Usa todos los dedos	Alemán

Puede formar una clave principal para esta relación al seleccionar la combinación de ID de empleado y Habilidad de oficina, o ID de empleado y Habilidad de idiomas. De este modo tiene dos opciones para relaciones en la tercera forma normal:

```
HABILIDAD DE EMPLEADO: ID DE EMPLEADO (CP), HABILIDAD DE OFICINA (CP),
                       HABILIDAD DE IDIOMAS

HABILIDAD DE EMPLEADO: ID DE EMPLEADO (CP), HABILIDAD DE IDIOMAS (CP),
                       HABILIDAD DE OFICINA
```

Las dos opciones presentadas están en la tercera forma normal, y también las dos aprueban la BCNF. Por supuesto, el problema es que existe una relación implícita entre las habilidades de oficina y las de idiomas. ¿La primera tupla para el empleado 1001 implica que sólo puede escribir en español? ¿Y la segunda implica que sólo puede mecanografiar con los 10 dedos en francés?

Este tipo de relaciones son raras en la vida real porque cuando los diseñadores experimentados resuelven problemas de un atributo con varios valores para satisfacer la primera forma normal, mueven cada atributo con varios valores a su propia relación, en lugar de combinarlos como se aprecia aquí. De modo que, con cierta interpretación estricta de los procedimientos de la primera forma normal, esto se evita del todo. Asimismo, si va a aplicar las reglas de la quinta forma normal, cubre las anomalías resueltas por la cuarta en términos mucho más fáciles de comprender, de modo que puede saltar este paso por completo. Sin embargo, si encuentra una violación a la cuarta forma normal, el remedio es poner cada atributo con valores múltiples en una relación separada, igual que aquí:

```
HABILIDAD DE OFICINA DEL EMPLEADO: ID DE EMPLEADO (CP), HABILIDAD DE
                                   OFICINA (CP)

HABILIDAD DE IDIOMAS DEL EMPLEADO: ID DE EMPLEADO (CP), HABILIDAD DE
                                   IDIOMAS (CP)
```

Quinta forma normal

La quinta forma normal es muy fácil de comprender. Simplemente sigue dividiendo las relaciones, y sólo deténgase cuando sea verdadera una de las condiciones siguientes:

- Cualquier división adicional conduciría a relaciones en donde la vista original no se puede reconstruir con combinaciones.

- Las únicas divisiones que quedan son triviales. Ocurren *divisiones triviales* cuando las relaciones resultantes tienen una clave principal formada sólo por la clave principal o la clave candidata de la otra relación.

Aunque la quinta forma normal parece prohibir todas las relaciones en tres sentidos, algunas de éstas son legítimas. Sólo surgen problemas cuando las entidades se pueden dividir en relaciones más sencillas y fundamentales.

Para casi todos los practicantes, la quinta forma normal es sinónimo de *completamente normalizada*. No obstante, en años recientes, el experto en administración de bases de datos C. J. (Chris) Date ha propuesto una sexta forma normal que atiende los datos temporales y de intervalos. Falta ver si será ampliamente adoptada o no.

Forma normal dominio-clave (DKNF)

Ron Fagin presentó una forma normal dominio-clave (Domain-Key Normal Form, DKNF) en un documento de investigación publicado en 1981. La teoría es que una relación está en la DKNF si y sólo si cada restricción sobre la relación es resultado de las definiciones de los dominios y las claves. Aunque Fagin pudo demostrar que las relaciones en la DKNF no tienen anomalías de modificación, no aportó un procedimiento o regla paso a paso para alcanzarla. Por lo tanto, el dilema es que los diseñadores no tienen una indicación firme del momento en que se ha alcanzado la DKNF para una relación. Ni existe la noción de que las restricciones son una consecuencia de claves obvias. Es probable que ésta sea la razón por la que el uso de la DKNF no se haya extendido ni se suele esperar en el diseño de bases de datos para aplicaciones empresariales. El interés académico también se ha desvanecido.

Desnormalización

Como ha visto, la normalización conduce a más relaciones, lo que se traduce en más tablas y combinaciones. Cuando los usuarios de una base de datos padecen problemas de desempeño que no se pueden resolver por otros medios, como afinar la base de datos o actualizar el hardware en que funciona el RDBMS, es posible que se requiera una desnormalización. Casi todos los expertos en bases de datos consideran la desnormalización como un último recurso, o incluso un acto de desesperación. Con los mejoramientos continuos en el hardware y las eficiencias en el RDBMS, una desnormalización se ha vuelto mucho menos necesaria que en los días iniciales de las bases de datos relacionales. El punto más esencial es que desnormalización no significa dejar de molestarse en normalizar desde el principio. Una vez que se ha logrado un diseño de base de datos normalizada, es posible hacer ajustes con las consecuencias potenciales (anomalías) en mente.

Entre los pasos posibles para una desnormalización están los siguientes:

- Volver a combinar las relaciones que fueron divididas para satisfacer las reglas de normalización.

- Guardar los datos redundantes en tablas.
- Guardar los datos resumidos en tablas.

Observe también que la normalización está diseñada para eliminar anomalías de bases de datos que se utilizan para sistemas de procesamiento de transacciones en línea. Las bases de datos que guardan datos históricos utilizados sólo para propósitos analíticos no están sujetas a anomalías de inserción, actualización y eliminación. En el capítulo 12 se ofrece más información acerca de las bases de datos que contienen información histórica.

Problemas prácticos

En esta sección se incluyen dos problemas prácticos (en forma de ejercicios Pruebe esto) con soluciones para que usted mismo pruebe la normalización. Éstos son problemas prácticos muy delimitados y a escala reducida, de modo que casi todos los lectores deben resolverlos en casi una hora cada uno. Conforme trabaje en ellos, tendrá más éxito si se concentra sólo en las listas presentadas y no se preocupa por otros procesos de negocios y datos que pudieran ser necesarios. Para cada problema práctico, la intención es que produzca relaciones en la tercera forma normal que permitan las listas presentadas y después dibuje un diagrama entidad-relación (ERD) de las relaciones formalizadas. Cuando dibuje los ERD, recuerde que son muy fáciles de crear una vez que una normalización esté completa: usted simplemente prepara un rectángulo para cada relación normalizada y después dibuja relaciones en todos los lugares de una relación en que se utiliza una clave principal como una clave externa en otra (o la misma) relación. Todas éstas deben ser relaciones uno a varios, y la clave externa siempre debe estar en el lado *varios* de la relación. Mi solución para cada problema aparece en el apéndice B.

Pruebe esto 6-1 Registros académicos en UTLA

La Universidad de Acrónimos de Tres Letras (UATL) es una pequeña institución académica que ofrece posgrados y educación continua para adultos. Gran parte del trabajo de conservación de registros académicos se hace en forma manual o con herramientas personales como hojas de cálculo. Está en proceso un esfuerzo de modernización, que incluye la creación de sistemas integrados de aplicaciones y base de datos para efectuar las funciones empresariales básicas.

Las vistas de usuarios

La UATL pretende construir un sistema para rastrear sus actividades académicas, entre ellas ofertas de cursos, aptitudes de los instructores para los cursos, inscripciones para cursos y calificaciones de los estudiantes. Las ilustraciones siguientes presentan los informes deseados con datos de muestra (éstas son las vistas de usuarios que deben normalizarse).

Informe de estudiante:

Informe de estudiante:

ID	Nombre	Dirección de correo				Teléfono de casa
4567	Juan Pérez Av.	Laurel 38	México	DF	07780	5348-1581
4973	Arturo López	Estelar 43	México	DF	07340	5643-1790
6758	Ricardo Juárez	Calle 3 51	México	DF	07893	6789-0312

Informe de curso:

Informe de curso:

ID	Título	Núm. de créditos	Cursos de prerrequisito	Descripción
X100	Conceptos de proc. de datos	4	Ninguno	El curso…
X301	Programación en C 1	4	X100	Los estudiantes…
X302	Programación en C 2	6	X301	Continuación de…
X408	Conceptos de DBMS	6	X301	El objetivo…
X422	Análisis de sistemas	6	X301, X422	Introducción a…

Informe de instructor:

Informe de instructor:

ID	Nombre	Dirección de casa	Tel. de casa	Tel. de oficina	Cursos
756	Jorge Pineda	Claveles 53 México DF 07780	5698-1098	x-7463 X408,	X422
795	Arturo Ramos	Estudiantes 10 México DF 07340	1234-5678	x-5328 X301,	X302
801	Gabriel García	Universidad 11 Toluca Estado de México 97346	345-7890 543-8915	X100,	X422

Informe de sección:

Informe de sección:

Año: 2010 **Semestre:** Spr **Edificio:** Facultad **Aula:** 70 **Días:** Mar **Hora:** 7-10

Instructor: 756, Jorge Pineda **Curso:** X408 **Créditos:** 6

Id de estudiante	Nombre de estudiante	Calificación
4567	Juan Pérez	10
6758	Ricardo Juárez	8

Año: 2010 **Semestre:** Spr **Edificio:** Anexo **Aula:** 7 **Días:** Mie **Hora:** 7-10

Instructor: 756, Jorge Pineda **Curso:** X408 **Créditos:** 6

Id de estudiante	Nombre de estudiante	Calificación
4973	Arturo López	8
6758	Ricardo Juárez	9

Año: 2010 **Semestre:** Spr **Edificio:** Facultad **Aula:** 70 **Días:** Mar, Vie **Hora:** 7-9

Instructor: 801, Gabriel García **Curso:** X100 **Créditos:** 4

Id de estudiante	Nombre de estudiante	Calificación

(*continúa*)

No es posible diseñar una base de datos sin cierto conocimiento de las reglas de negocios y procesos de una organización. Éstos son algunos elementos que debe recordar:

- Para cada estudiante, sólo se conserva una dirección para correspondencia y un número telefónico.
- Cada curso tiene una cantidad fija de créditos (es decir, no se ofrecen cursos con créditos variables).
- Cada curso puede tener uno o más cursos que son requisitos previos. La lista de todos los requisitos previos para cada curso se presenta en Informe de curso.
- De cada instructor, sólo se conservan una dirección para correspondencia, un número telefónico de casa y uno de oficina.
- Un comité de calificación debe aprobar a los instructores antes de que se les permita impartir un curso específico. Después se agregan las aptitudes (es decir, los cursos que el comité ha determinado que el instructor está calificado para enseñar) a los registros de instructor, como se aprecia en Informe de instructor. La lista de cursos para los que es apto no implica que el instructor en realidad ha impartido el curso, sino sólo que está calificado para hacerlo.
- Con base en la demanda, cualquier curso puede ofrecerse varias veces, aun en el mismo año y semestre. Cada oferta se denomina "sección" y se aprecia en Informe de sección.
- Los estudiantes se inscriben en una sección específica de un curso y reciben una calificación por su participación en esa oferta de curso. Si vuelven a tomar el curso en un momento posterior, reciben otra calificación, y ambas son parte de su registro académico permanente.
- Aunque se anotan el día, la hora, el edificio y el aula en Informe de sección, esto se hace sólo para facilitar el registro de estudiantes. La programación de aulas está fuera del alcance de este proyecto.
- Los atributos de días y horas en Informe de sección son sólo descripciones del programa de reuniones. La creación de un calendario de reuniones para las secciones está fuera del alcance de este proyecto.

Como resulta útil verlos, éstos son los atributos reescritos mediante el método de lista de relaciones, con los grupos repetidos y los atributos con valores múltiples entre paréntesis:

```
INFORME DE ESTUDIANTE: ID, NOMBRE DE ESTUDIANTE, DIRECCIÓN DE ESTUDIANTE,
                       CIUDAD, ESTADO, CÓDIGO POSTAL, TELÉFONO DE CASA

INFORME DE CURSO: ID, TÍTULO, NÚMERO DE CRÉDITOS,
                  (CURSO DE PRERREQUISITO), DESCRIPCIÓN
```

```
INFORME DE INSTRUCTOR: ID, NOMBRE DE INSTRUCTOR, DIRECCIÓN,
                       CIUDAD, ESTADO, CÓDIGO POSTAL, TELÉFONO DE CASA,
                       TELÉFONO DE OFICINA, (CURSOS CALIFICADOS)

INFORME DE SECCIÓN: AÑO, SEMESTRE, EDIFICIO, AULA, DÍAS,
                    HORAS, ID DE INSTRUCTOR, NOMBRE DE INSTRUCTOR,
                    ID DE CURSO, NÚMERO DE CRÉDITOS,
                    (ID DE ESTUDIANTE, NOMBRE DE ESTUDIANTE,
                    CALIFICACIÓN)
```

Paso a paso

1. Estudie cada vista de usuario de la descripción anterior, junto con las reglas de negocios. Tal vez tenga que hacer algunas suposiciones, si tiene preguntas que la descripción no responde.
2. Aplique el proceso de normalización descrito en el capítulo para normalizar cada vista como relaciones que estén en cuando menos la tercera forma normal. Tenga cuidado de consolidar las relaciones normalizadas que desarrolle conforme avanza. Para los propósitos de este ejercicio, dos relaciones no deben compartir la misma clave principal. (Las excepciones a esta regla se cubren en los capítulos siguientes.)
3. Indique con claridad la clave principal de cada relación. Recuerde que una clave principal puede ser uno o más atributos dentro de la relación.
4. Dibuje un ERD con una entidad (rectángulo) para cada relación normalizada y líneas de relación adecuadas con la cardinalidad indicada en forma clara. Esto debe ser muy fácil de hacer una vez concluida la normalización: simplemente dibuje una línea desde cada clave externa hasta la clave principal correspondiente y marque el extremo de la línea de la clave externa como “varios” y el extremo de la clave principal como “uno”.

Resumen de Pruebe esto

En este ejercicio Pruebe esto, normalizó cuatro vistas de usuario y dibujó un ERD de su diseño. Mi solución aparece en el apéndice B.

Pruebe esto 6-2 Compañía de Libros de Computación

La Compañía de Libros de Computación (CLC) compra libros a editoriales y los vende mediante pedidos por correo y telefónicos. Pretende expandir sus servicios al ofrecer pedidos en línea a través de Internet, y para hacer eso, CLC tiene una urgente necesidad de crear una base de datos que contenga su información de negocios.

Las vistas de usuario

A través de todas estas vistas de usuarios, "venta" y "precio" son referencias a la venta minorista de un libro a un cliente de CLC, mientras que "compra" y "costo" son referencias a la compra de libros a una editorial (proveedora de CLC). Cada vista de usuario se describe brevemente con una lista de sus atributos después de cada descripción. Por convención, los atributos con valores múltiples y los grupos repetidos están entre paréntesis.

El Catálogo de libros presenta todos los libros que CBC tiene para venta. Cada libro se identifica de manera única mediante el número de libro estándar internacional (ISBN, International Standard Book Number). Aunque un ISBN identifica un libro de manera inequívoca, en esencia es una clave sustituta, por lo que no hay modo de saber la edición de un libro específico con sólo mirar el ISBN. Cuando salen ediciones nuevas, CLC suele tener existencias sobrantes de ediciones anteriores y las ofrece a un precio reducido. El ISBN de una edición previa en el catálogo de libros está diseñado para ayudar al comprador a localizar una edición anterior, si existe. Los libros están organizados por temas, por lo que cada libro sólo tiene un tema. Cualquier libro puede tener varios autores. (Aunque el catálogo presenta sólo nombres de autores, recuerde que los nombres de las personas rara vez son únicos, y nada evita que dos personas con el mismo nombre escriban dos libros distintos.)

Ésta es la información de Catálogo de libros:

```
CATÁLOGO DE LIBRO: CÓDIGO DE TEMA, DESCRIPCIÓN, TÍTULO DE LIBRO,
                   ISBN DE LIBRO, PRECIO DE LIBRO, ISBN DE EDICIÓN ANTERIOR,
                   PRECIO DE EDICIÓN ANTERIOR, (AUTORES DE LIBRO),
                   NOMBRE DE EDITOR
```

El informe Inventario de libros ayuda al gerente de almacén a controlar el inventario que contiene. La Cantidad recomendada es el punto para volver a hacer un pedido, lo que significa que cuando el inventario disponible cae por debajo de la cantidad recomendada, es el momento de pedir más libros de ese título.

```
REPORTE DE INVENTARIO: ISBN DE LIBRO, CÓDIGO DE EDICIÓN DE LIBRO,
                       COSTO, PRECIO DE VENTA, EJEMPLARES DISPONIBLES,
                       EJEMPLARES SOLICITADOS, CANTIDAD RECOMENDADA
```

La vista Pedidos de libros de clientes muestra los pedidos hechos por los clientes de CLC para compras de libros:

```
PEDIDOS DE LIBROS DE CLIENTES: ID DE CLIENTE, NOMBRE DE CLIENTE,
                               DIRECCIÓN DE CALLE, CIUDAD, ESTADO,
                               CÓDIGO POSTAL (ISBN, CÓDIGO DE EDICIÓN,
                               CANTIDAD, PRECIO), FECHA DE PEDIDO,
                               PRECIO TOTAL
```

CLC factura a los clientes cuando los libros son enviados, de modo que los pedidos no enviados no tienen una factura. Se crea una factura para cada envío. (Un pedido puede tener cero, una o más facturas, pero cada factura pertenece a sólo un pedido.) La Factura de venta de libros se ve así:

```
FACTURA DE VENTA DE LIBROS: NÚMERO DE FACTURA DE VENTAS, ID DE CLIENTE,
                            NOMBRE DE CLIENTE, DIRECCIÓN DE CALLE DE
                            CLIENTE, CIUDAD DE CLIENTE, ESTADO DE
                            CLIENTE,CÓDIGO POSTAL DE CLIENTE, (ISBN DE
                            LIBRO, TÍTULO, CÓDIGO DE EDICIÓN, (AUTORES
                            DE LIBRO), CANTIDAD, PRECIO, NOMBRE DE
                            EDITOR), GASTOS DE ENVÍO, IMPUESTO DE VENTA
```

El informe Facturación maestra ayuda a los departamentos de cobranza y servicios al cliente a administrar las cuentas de los clientes. Un sistema para registrar los pagos de los clientes en contra de las facturas está fuera del alcance del proyecto actual, pero los promotores del proyecto de CLC quieren mantener un saldo actual que muestre cuánto debe cada cliente a la empresa. Cuando se generen facturas, se utilizará un activador de base de datos para agregar los totales de una factura Saldo vencido. Cuando se reciban los pagos, el personal de CLC ajustará manualmente el saldo por cobrar. Éstos son los atributos del informe Facturación maestra:

```
FACTURACIÓN MAESTRA: ID DE CLIENTE, NOMBRE DE CLIENTE, DIRECCIÓN DE
                     CALLE, CIUDAD, ESTADO, CÓDIGO POSTAL, TELÉFONO,
                     SALDO VENCIDO
```

Cada vez que CLC compra libros a una editorial, ésta envía una factura a CLC. Para ayudar a administrar el costo del inventario, CLC pretende guardar la información de Facturas de compra y comunicarla mediante esta vista:

```
FACTURAS DE COMPRA: ID DE EDITOR, NOMBRE DE EDITOR,
                    DIRECCIÓN DE CALLE, CIUDAD, ESTADO, CÓDIGO POSTAL,
                    NÚMERO DE FACTURA DE COMPRA, FECHA DE FACTURA,
                    (ISBN DE LIBRO, CÓDIGO DE EDICIÓN, TÍTULO,
                    CANTIDAD, COSTO UNITARIO, COSTO EXTENDIDO),
                    COSTO TOTAL
```

Observe que Costo extendido se calcula como Costo unitario por Cantidad.

(*continúa*)

Paso a paso

1. Estudie cada una de las vistas de usuarios en la descripción anterior, junto con las reglas de negocios. Es posible que tenga que plantear suposiciones, si tiene preguntas que la descripción no responde.
2. Aplique el proceso de normalización descrito en este capítulo, y normalice cada vista a relaciones que estén en cuando menos la tercera forma normal. Tenga cuidado de consolidar las relaciones normalizadas que prepare conforme avance. Para este ejercicio, dos relaciones no deben compartir la misma clave principal. (Las excepciones a esta regla se cubren en los capítulos siguientes.)
3. Indique con claridad la clave principal de cada relación. Recuerde que una clave principal puede ser uno o más atributos dentro de la relación.
4. Dibuje un ERD con una entidad (rectángulo) para cada relación normalizada y líneas de relación adecuadas con la cardinalidad indicada de forma clara. Esto debe ser muy fácil de hacer una vez concluida la normalización: simplemente dibuje una línea desde cada clave externa hasta la clave principal correspondiente y marque el extremo de la línea de la clave externa como "varios" y el extremo de la clave principal como "uno".

Resumen de Pruebe esto

En este ejercicio Pruebe esto, normalizó seis vistas de usuarios que eran más complicadas que las del ejercicio Pruebe esto anterior y dibujó un ERD de su diseño. Mi solución aparece en el apéndice B.

✓ Autoexamen Capítulo 6

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. La normalización
   - **A** Fue desarrollada por E. F. Codd.
   - **B** Fue introducida primero con cinco formas normales.
   - **C** Apareció en 1972.
   - **D** Proporciona un grupo de reglas para cada forma normal.
   - **E** Ofrece un procedimiento para convertir las relaciones en cada forma normal.

2. El propósito de la normalización es

- **A** Eliminar los datos redundantes.
- **B** Eliminar ciertas anomalías de las relaciones.
- **C** Aportar una razón para desnormalizar la base de datos.
- **D** Optimizar el desempeño de la recuperación de datos.
- **E** Optimizar los datos para inserciones, actualizaciones y eliminaciones.

3. Cuando se implementa, una relación en la tercera forma normal se convierte en _____.

4. Anomalía de inserción alude a una situación en que

- **A** Los datos deben insertarse antes de que puedan eliminarse.
- **B** Demasiadas inserciones hacen que la tabla se llene.
- **C** Los datos deben eliminarse antes de que puedan insertarse.
- **D** Una inserción requerida no puede hacerse debido a una dependencia artificial.
- **E** Una inserción requerida no puede hacerse a causa de datos duplicados.

5. Anomalía de eliminación alude a una situación en que

- **A** Los datos deben eliminarse antes de que puedan insertarse.
- **B** Los datos deben insertarse antes de que puedan eliminarse.
- **C** Una eliminación de datos provoca una pérdida no intencional de datos de otra entidad.
- **D** Una eliminación requerida no puede hacerse debido a restricciones referenciales.
- **E** Una eliminación requerida no puede hacerse por falta de privilegios.

6. Anomalía de actualización alude a una situación en que

- **A** Una actualización sencilla requiere actualizaciones a varias filas de datos.
- **B** Los datos no pueden actualizarse porque no existen en la base de datos.
- **C** Los datos no pueden actualizarse a causa de una falta de privilegios.
- **D** Los datos no pueden actualizarse debido a una restricción de unicidad.
- **E** Los datos no pueden actualizarse a causa de una restricción referencial existente.

7. Las reglas de los identificadores únicos en la normalización

- **A** Son innecesarias.
- **B** Son necesarias una vez que alcanza la tercera forma normal.
- **C** Todas las formas normalizadas requieren la designación de una clave principal.
- **D** No es posible normalizar relaciones sin primero elegir una clave principal.
- **E** No es posible elegir una clave principal hasta que se normalizan las relaciones.

8. Escribir vistas de usuarios de ejemplo con datos representativos en ellas es

- **A** El único modo de normalizar las vistas de usuarios con éxito.
- **B** Un proceso tedioso y que consume tiempo.
- **C** Un modo eficaz para comprender los datos que se normalizan.
- **D** Tan bueno como los ejemplos presentados en los datos de ejemplo.
- **E** Una técnica de normalización ampliamente utilizada.

9. Algunos criterios útiles para elegir una clave principal entre varias claves candidatas son

- **A** Seleccionar la candidata más sencilla.
- **B** Escoger la candidata más breve.
- **C** Elegir la candidata con más probabilidad de que cambie su valor.
- **D** Preferir las claves unidas a las claves con un solo atributo.
- **E** Inventar una clave sustituta si es la mejor clave posible.

10. La primera forma normal resuelve las anomalías provocadas por _____.

11. La segunda forma normal soluciona las anomalías inducidas por _____.

12. La tercera forma normal elimina las anomalías generadas por _____.

13. En general, las violaciones a una regla de normalización son resueltas por medio de

- **A** Combinación de relaciones.
- **B** Desplazamiento de atributos o grupos de atributos a una relación nueva.
- **C** Combinación de atributos.
- **D** Creación de tablas de resumen.
- **E** Una desnormalización.

14. Una clave externa en una relación normalizada puede ser

- **A** La clave principal completa de la relación.
- **B** Una parte de la clave principal completa de la relación.
- **C** Un grupo repetido.
- **D** Un atributo que no es una clave en la relación.
- **E** Un atributo con varios valores.

15. La forma normal de Boyce-Codd atiende las anomalías causadas por _____.

16. La cuarta forma normal resuelve las anomalías provocadas por _____.

17. La quinta forma normal enfrenta las anomalías generadas por _____.

18. La forma normal dominio-clave soluciona las anomalías derivadas de _____.

19. Casi todos los sistemas de negocios requieren ser normalizados sólo hasta _____.

20. El manejo adecuado de los atributos con varios valores, al convertir relaciones a la primera forma normal, suele evitar problemas subsecuentes con_____.

Capítulo 7

Modelado de datos y de procesos

Habilidades y conceptos clave

- Modelado de la relación entre entidades.
- Modelos de procesos.
- Determinación de relaciones entre entidades y procesos.

Tal como observó en el capítulo 5, los modelados de datos y procesos son pasos importantes que forman parte de la etapa del diseño lógico de un proyecto de desarrollo de un sistema de aplicaciones. Ya ha visto los fundamentos del modelado de datos cuando utilizó los diagramas entidad-relación (ERD) en los capítulos anteriores. En este capítulo se analizan con mayor detalle los ERD y el modelado de datos. Por otra parte, el modelado de procesos es menos importante para un diseñador de una base de datos porque los procesos de una aplicación son preparados por diseñadores de aplicaciones y rara vez participa directamente el diseñador de una base de datos. Sin embargo, como éste debe colaborar estrechamente con el diseñador de aplicaciones en la recopilación de los requisitos de datos y el aporte de un diseño de base de datos que permita los procesos que se diseñan, el diseñador de base de datos debe cuando menos estar familiarizado con los conceptos básicos. Por esta razón, en la segunda parte de este capítulo se incluye un examen de alto nivel de los conceptos del diseño de procesos y de técnicas de diagramación.

Modelado de la relación entre entidades

El *modelado de las relaciones entre entidades* es el proceso de representar visualmente entidades, atributos y relaciones para producir un ERD. El proceso es de naturaleza iterativa porque las entidades son descubiertas durante todo el proceso de diseño. La principal ventaja de un ERD es que puede ser comprendido por personas sin conocimientos técnicos y al mismo tiempo ser de gran valor para el personal técnico. Cuando se realizan correctamente, los ERD son independientes de la plataforma e incluso pueden utilizarse para bases de datos no relacionales, si es necesario.

Formatos de ERD

Peter Chen desarrolló el formato de ERD original en 1976. Desde entonces, vendedores, científicos computacionales y académicos han desarrollado muchas variaciones, todas ellas conceptualmente iguales. Debe comprender las variaciones que se usan con mayor frecuencia porque es posible que las encuentre en uso en las organizaciones con tecnología de la información. Éstos son los elementos comunes para todos los formatos de ERD:

- Las entidades son representadas como rectángulos o cuadros.
- Las relaciones son representadas como líneas.
- Los extremos de las líneas (o los símbolos junto a ellas) indican la cardinalidad máxima de la relación (es decir, uno a varios).
- Los símbolos cerca de los extremos de las líneas (en casi todos los formatos ERD) indican la cardinalidad mínima de la relación (es decir, cuando la participación en la relación es obligatoria u opcional).
- Los atributos se incluyen de manera opcional (el formato para mostrar atributos varía bastante).

El formato de Chen

Por simplicidad, para los ejemplos de este capítulo se empleará la solución normalizada para la aplicación de facturas de Industrias Acme del capítulo 6. En la figura 7-1 se presenta el ERD que utiliza el formato de Chen.

Éstos son los detalles del formato de Chen:

- Las líneas de relaciones contienen un rombo en que se escribe una palabra o frase breve que describe la relación. Por ejemplo, la relación entre Factura y Producto se lee como "una factura *contiene* muchos productos". Algunas variaciones permiten que se emplee otra palabra o frase para leer la relación en la dirección contraria, separada con una diagonal. Si el rombo dice "Contiene/Aparece en", la relación de Producto con Factura se leería "Un producto *aparece en* varias facturas".
- Para muchas relaciones varios a varios que requieren una tabla de intersección en un RDBMS, como la que existe entre Factura y Producto, se suele dibujar un rectángulo alrededor del rombo.
- La cardinalidad máxima de cada relación se indica mediante el símbolo *1* para "uno" o *V* para "varios".

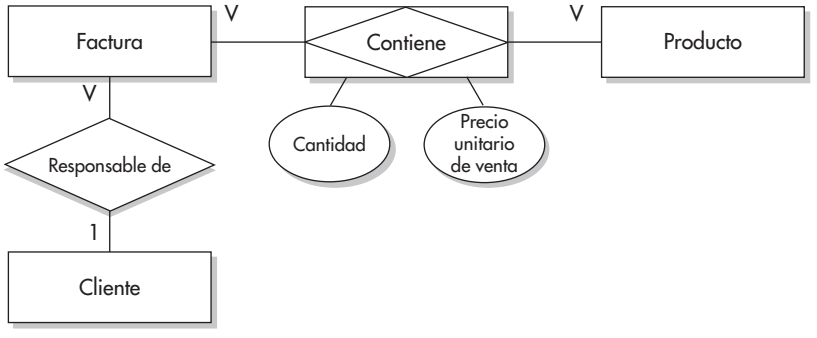


Figura 7-1 Formato de ERD lógico de Industrias Acme en el formato de Chen.

- No se muestra la cardinalidad mínima.
- Los atributos, cuando se muestran, aparecen en elipses (círculos alargados) conectados con una línea a la entidad o relación a la que pertenecen.

En la práctica, los ERD de Chen son problemáticos para los modelos de datos complicados. Los rombos ocupan mucho espacio en los diagramas a cambio del poco valor agregado que aportan. Asimismo, cualquier ERD que incluye muchos atributos difícil de leer. No obstante, se debe reconocer gran parte del trabajo pionero de Chen, quien estableció las bases para las técnicas siguientes.

El formato relacional

Con el tiempo, evolucionó un formato de ERD conocido genéricamente como *formato relacional*. Está disponible como una opción en varias de las herramientas de software para modelado de datos más conocidas, como PowerDesigner de Sybase y ER/Studio de Embarcadero Technologies, y en instrumentos de dibujo de uso popular como Visio, de Microsoft. En la figura 7-2 se expone el ERD de la figura 7-1 convertido al formato relacional. En este ejemplo, el ERD se representa en un nivel físico, lo que significa que se muestran los nombres de las tablas físicas en lugar de los nombres de las entidades lógicas, y se muestran los nombres de las columnas físicas en lugar de nombres de los atributos lógicos. Asimismo, se muestran tablas de intersección para resolver las relaciones varios a varios. Conforme el modelo lógico de datos se transforme en un diseño físico de una base de datos, es esencial tener un ERD físico que el equipo de proyecto pueda emplear para desarrollar el sistema de aplicaciones. Los inicios del modelo físico se muestran aquí para aclarar este punto.

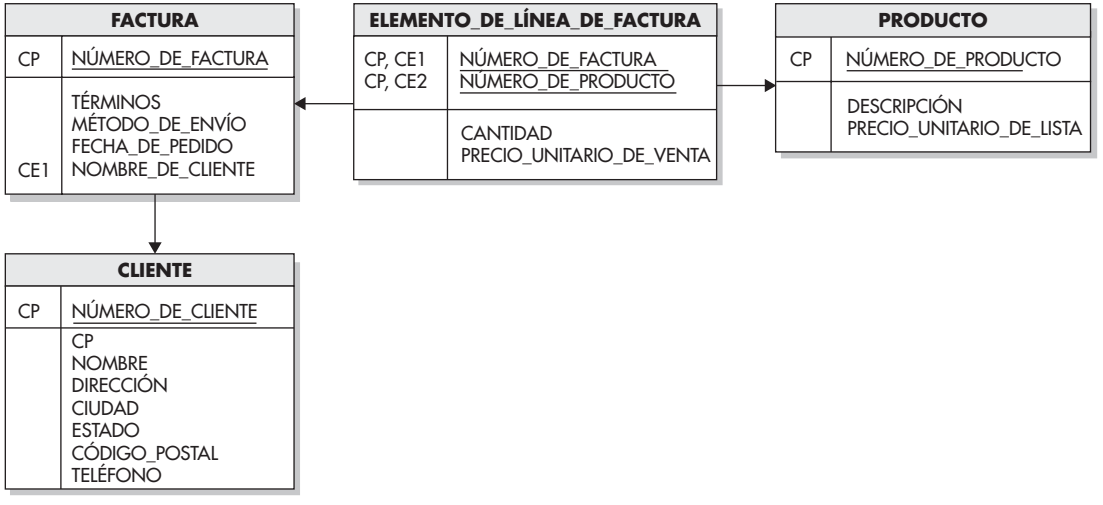


Figura 7-2 ERD físico de Industrias Acme, formato relacional.

Éstos son los conceptos específicos del formato de ERD relacional:

- La cardinalidad de una relación se presenta con una punta de flecha en el extremo de la línea para indicar "uno" y nada en el extremo de la línea para señalar "varios". Al principio esto parecerá extraño, pero se acopla bien con los diagramas de objetos, de modo que este formato es preferido por los diseñadores y desarrolladores orientados a objetos.
- Los atributos se muestran dentro del rectángulo que representa cada entidad.
- Los atributos de identificador único se exponen sobre una línea horizontal dentro del rectángulo y también se suelen presentar con las letras **CP**, de clave principal, en el margen, a la izquierda del nombre del atributo.
- Los atributos que son claves externas se muestran con las letras y un número en el margen, a la izquierda del nombre del atributo.

El formato de ingeniería de la información

El formato de ingeniería de la información (II) fue desarrollado por Clive Finkelstein, en Australia, a fines de la década de 1970. A principios de la década siguiente colaboró con James Martin para promoverlo en Estados Unidos y Europa, y fue coautor para el Savant Institute de un informe titulado *Information Engineering* (ingeniería de la información), publicado en 1981. Martin mantuvo un marcado interés por el formato y, en colaboración con Carma McClure, publicó un libro sobre el tema en 1984 (*Diagramming Techniques for Analysis and Programmers*, Prentice-Hall). Finkelstein publicó su propia versión en 1989 (*An Introduction to Information Engineering*, Addison-Wesley), que tiene ciertas variaciones menores de notación comparado con la versión de Martin. En la figura 7-3 se muestra el ERD de ejemplo convertido a la notación II. Observará que, excepto por las líneas de relaciones, es sorprendentemente similar al formato relacional.

Éstas son algunas diferencias entre la notación II y el formato relacional:

- **Relaciones identificadoras** Se presentan con una línea continua y son aquellas en que la clave externa es parte de la clave principal de la entidad secundaria.
- **Relaciones que no son identificadoras** Se muestran con una línea de guiones y son aquellas en que la clave externa es un atributo que no es una clave en la entidad secundaria. En la figura 7-3, la relación entre PRODUCTO y ELEMENTO_DE_LÍNEA_DE_FACTURA es identificadora, pero la que existe entre CLIENTE y FACTURA no es identificadora.
- **Cardinalidad máxima de la relación** Se indica con una línea perpendicular a la línea de la relación, cerca de su extremo, para indicar "uno", y una "pata de gallo" en el otro extremo para señalar "varios". Esto se comprende mejor en combinación con la cardinalidad mínima, que se describe a continuación.

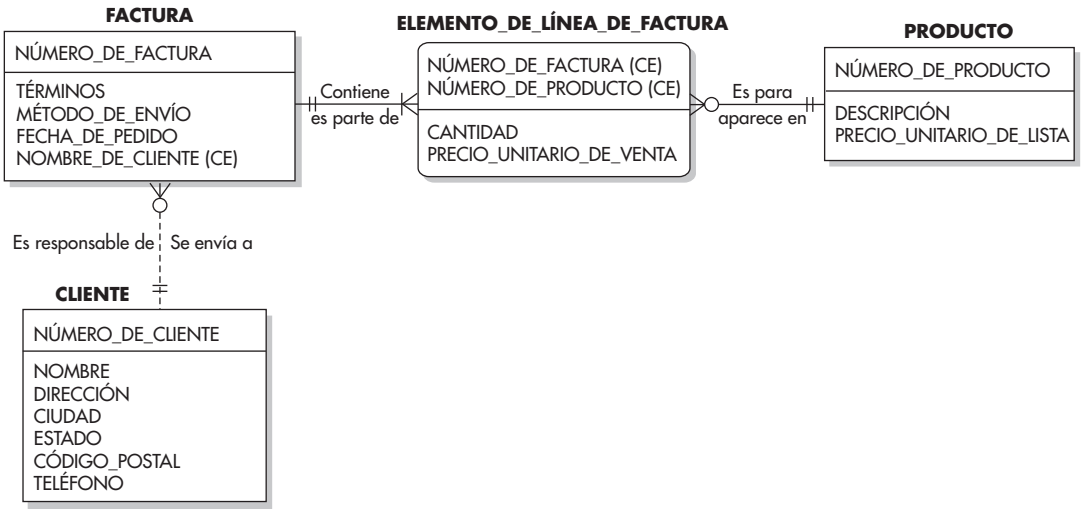


Figura 7-3 ERD físico de Industrias Acme, formato II.

- **Cardinalidad mínima de la relación** Se expone con un círculo pequeño cerca del extremo de la línea para indicar "cero" (la participación en la relación es opcional) o una breve línea perpendicular a la línea de la relación para indicar "uno" (la participación en la relación es obligatoria). En la figura 7-3 se muestran algunas combinaciones de cardinalidad mínima y máxima. Por ejemplo:
  - **Un PRODUCTO** Puede tener cero a varias ocurrencias de ELEMENTO_DE_LÍNEA_DE_FACTURA asociadas (señaladas con un círculo y una pata de gallo); un ELEMENTO_DE_LÍNEA_DE_FACTURA debe tener uno y sólo un PRODUCTO asociado (indicado con las dos barras verticales).
  - **Una FACTURA** Debe tener una o más ocurrencias de ELEMENTO_DE_LÍNEA_DE_FACTURA asociadas (expuestas con una barra vertical y una pata de gallo); un ELEMENTO_DE_LÍNEA_DE_FACTURA debe tener una y sólo una FACTURA asociada (señalada con dos barras verticales).
- **Entidades dependientes** Se indican con un rectángulo con las esquinas redondeadas, y su existencia depende de una o más de las otras entidades (es decir, no pueden existir sin otras). Por ejemplo, la entidad ELEMENTO_DE_LÍNEA_DE_FACTURA depende de las entidades PRODUCTO y FACTURA. Por lo tanto, no es posible eliminar una factura o un producto, a menos que de alguna manera se haga algo con los artículos de línea de factura relacionados. Ésta es información valiosa durante el diseño físico de una base de datos, porque debe considerar las opciones para manejar situaciones en que la aplicación intenta eliminar filas de una tabla cuando existen entidades dependientes.

El formato II es por mucho el más popular. Por tal razón, se emplea para casi todos los diagramas de este libro.

El formato IDEF1X

El Computer Systems Laboratory del National Institute of Standards and Technology publicó la norma IDEF1X para el modelado de datos en la publicación FIPS 184, en diciembre de 1993. La norma cubre un método para modelado de datos y el formato para los ERD producidos durante el esfuerzo de modelado. Se usa y comprende en todas partes en la industria de la tecnología de la información y es obligatoria para muchas dependencias del gobierno de Estados Unidos. Gracias a que tiene implícita una norma, cuenta con pocas variantes. En la figura 7-4 se muestra el ERD de ejemplo convertido al formato de la norma IDEF1X.

Las diferencias entre las notaciones II e IDEF1X ocurren en gran medida en las relaciones. A diferencia de otros formatos, los símbolos de relaciones en IDEF1X son simétricos. Cada grupo de símbolos describe una combinación de condición de opcionalidad y cardinalidad y, por lo tanto, los símbolos utilizados para opcionalidad dependen de la cardinalidad de la relación. Dicho de otro modo, algo opcional se muestra de manera diferente para los lados "varios" y "uno" de una relación. He aquí algunos puntos importantes:

- Igual que en el formato II, una línea continua indica que la clave externa es parte de la clave principal de la entidad dependiente, mientras que una línea de guiones señala que la clave externa será un atributo que no es una clave.
- Un círculo relleno junto a una entidad suele significar cero, una o más ocurrencias de esa entidad, como se aprecia en el extremo "varios" de la línea entre PRODUCTO y ELEMENTO_DE_LÍNEA_DE_FACTURA. No obstante, hay excepciones:

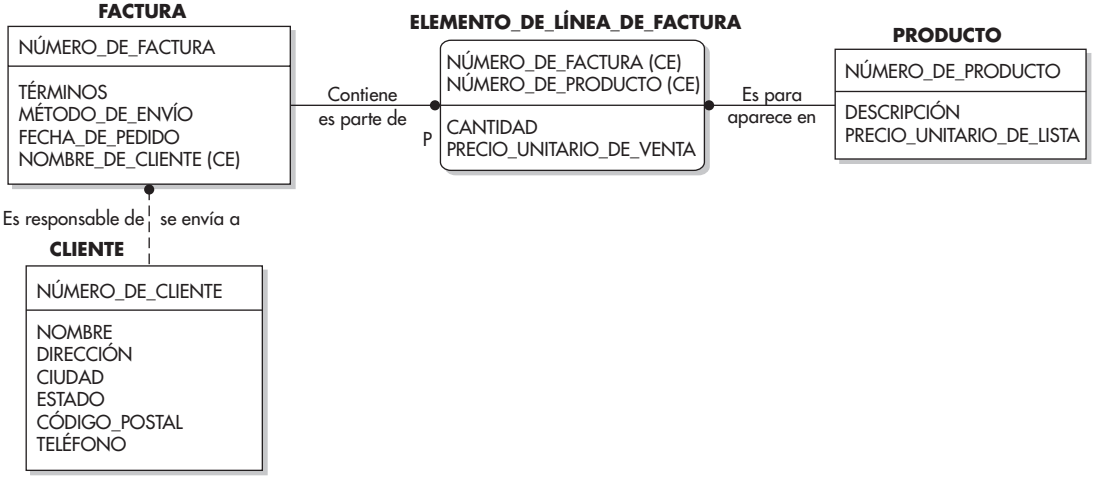


Figura 7-4 ERD físico de Industrias Acme, formato IDEF1X.

- La inclusión del símbolo *P* cerca del círculo relleno hace obligatoria la relación, lo que significa que la cardinalidad debe ser uno o más. En la figura 7-4, la relación de FACTURA con ELEMENTO_DE_LÍNEA_DE_FACTURA es uno a varios y obligatoria, lo que significa que cada factura debe tener cuando menos un artículo de línea.
- La incorporación del símbolo *I* también hace obligatoria la relación. Sin embargo, esto cambia a uno la cardinalidad de la relación. Dicho de otro modo, cambia el significado del círculo relleno de "puede ser uno o más" a "debe ser uno y sólo uno".

- La ausencia de un círculo relleno en el extremo de la línea de la relación significa que participa la única ocurrencia de la entidad. Por ejemplo, la ausencia de cualquier símbolo en el extremo de la línea junto a CLIENTE significa "uno y sólo uno". También puede modificarse la condición de opcional:
  - Si no aparece un símbolo junto a la entidad en ese extremo de la línea, la entidad es obligatoria en la relación. Por lo tanto, un ELEMENTO_DE_LÍNEA_DE_FACTURA debe relacionarse con un y sólo un PRODUCTO.
  - Si aparece un símbolo de rombo pequeño junto a la entidad, ésta es opcional. Si se agregara un rombo junto al extremo CLIENTE de la relación entre FACTURA y CLIENTE, significaría que cada FACTURA *puede* tener cero o una ocurrencia de CLIENTE relacionada.

Modelado de relaciones entre entidades con el lenguaje de modelado unificado

Con la creciente popularidad de los lenguajes de programación de objetos, también se ha vuelto más popular el *lenguaje de modelado unificado* (Unified Modeling Language, UML). UML es un lenguaje de especificación visual estandarizada para modelado de objetos que incluye una notación gráfica que sirve para crear un modelo abstracto de un sistema, conocido como modelo UML. El proceso unificado racional (RUP, Rational Unified Process), presentado brevemente en el capítulo 5, emplea exclusivamente el UML, que tiene 13 tipos de diagramas que se emplean para modelar el comportamiento y la estructura de un sistema. No obstante, el que interesa a quienes modelan datos es el diagrama de clases. En la figura 7-5 se presenta el modelo de ejemplo convertido en un diagrama de clases de UML.

Aunque las diferencias en la notación son demasiado obvias, una persona con experiencia en la lectura de diagramas ER se adapta con facilidad. En el diagrama se emplean a menudo nombres con *mayúscula inicial*; es decir, palabras unidas sin delimitadores entre ellas y sólo su primera letra en mayúsculas, porque casi todos los modeladores hacen eso. Éstos son algunos puntos importantes en relación con el modelado de entidades mediante diagramas de clases de UML:

- Cada entidad se muestra como una clase de objeto en un rectángulo. Se incluye el símbolo <<Entity>> (entidad) con el nombre de la clase para denotar el tipo de clase.

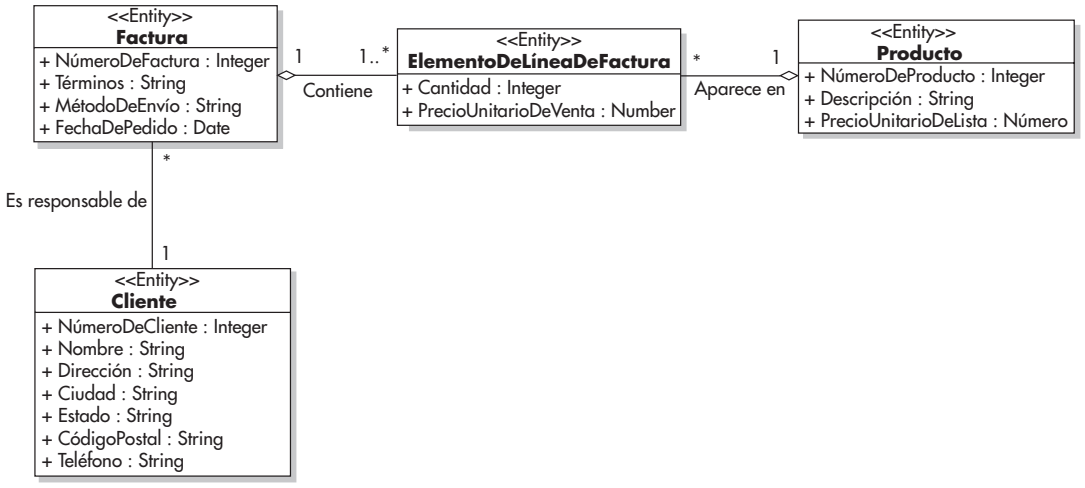


Figura 7-5 Diagrama de clases de UML de Industrias Acme.

- Los identificadores únicos (claves principales) no se muestran en los diagramas de clases; se especifican en otro lugar dentro del modelo UML.
- No se presentan las claves externas porque no se utilizan en los sistemas orientados a objetos. La tecnología orientada a objetos se analiza en el capítulo 13.
- Los atributos se exponen con un nombre y un tipo (separados con un símbolo de dos puntos). El tipo es muy similar a un tipo de datos relacional. En las entidades, se antepone a los atributos el símbolo de más (+), lo que significa que son públicos (que se pueden ver en todo el modelo).
- Las relaciones se indican con líneas.
- La cardinalidad y la condición de opcionalidad se señalan con un símbolo combinado cerca del extremo de la línea. Los símbolos disponibles aparecen en la tabla siguiente:

Símbolo	Significado
1	Uno y sólo uno.
*	Cero, uno o más.
1..*	Uno o más.
x..y	Entre x y y ocurrencias. También . x puede ser 0 o cualquier entero positivo. . y puede ser cualquier entero positivo o * para denotar "o más". . y debe ser mayor que x (si y y x son iguales, simplemente se omite y).

- El símbolo de rombo en el extremo de una línea de relación, como el de la figura 7-5 en el extremo "uno" de las dos relaciones conectadas a ElementoDeLíneaDeFactura, indica lo que el UML llama un conjunto. Un *conjunto* es una dependencia entre dos tipos de entidades que se requiere para la existencia de la entidad dependiente. En este caso, un artículo de línea no puede existir sin el producto y la factura. Si la dependencia es siempre una entidad única, aparece un rombo relleno en lugar de uno vacío.
- La generalización y la especialización (los supertipos y los tipos secundarios) se representan mediante una línea entre las dos entidades, en donde una flecha vacía apunta hacia la clase general (el supertipo).

Supertipos y tipos secundarios

Algunas entidades pueden dividirse en categorías o tipos más específicos. Cuando ocurre esto, a las entidades más detalladas se les denomina *tipos secundarios*, y la entidad más general a la que pertenecen es un *supertipo*. En la terminología de objetos, al supertipo se le denomina *superclase* o *clase base* y a los tipos secundarios se les llama *clases secundarias* de la superclase. Es esencial comprender que los tipos secundarios dividen las entidades por tipo más que por *estado*; es decir, por su modo o condición. Una manera fácil de diferenciar ambos consiste en comprender que las entidades existentes pueden cambiar de estado, pero rara vez, si llega a ocurrir, cambian de tipo. Por ejemplo, una entidad vehículo motorizado se puede dividir lógicamente por tipo en automóvil, autobús, camión, motocicleta, etc. Sin embargo, la diferencia entre vehículos que son nuevos o usados, o entre los que son operables o inoperables, es de *estado*, más que de *tipo*, porque los vehículos nuevos se convierten en usados una vez que se venden, y los vehículos cambian entre estados operable e inoperables conforme sufren averías y después son reparados.

Las decisiones acerca de cuáles entidades deben dividirse en tipos secundarios y cuán detallados deben ser los tipos secundarios se centran en una solución intermedia entre la especialización y la generalización. Por desgracia, no existen reglas firmes para llegar a una solución intermedia. Por lo tanto, la generalización en contra de la especialización se vuelve uno de los temas que evitan que el diseño de bases de datos se vuelva una ciencia exacta. El lineamiento general a seguir (además del sentido común) es que en cuantas más ocasiones los diversos tipos secundarios compartan atributos comunes y relaciones, más veces el diseñador deberá optar por combinarlos en un supertipo. Las soluciones intermedias del diseño físico correspondiente se analizan en el capítulo 8. Aquí se revisarán las del diseño lógico.

He aquí un ejemplo. Suponga por un momento que se ha implementado el diseño de la base de datos mostrado en la figura 7-3, y ahora el Departamento de Servicios al Cliente de Industrias Acme ha solicitado mejoras a la base de datos y a las aplicaciones que le permitan registrar y rastrear más información acerca de los clientes. En particular, al departamento le interesa conocer el tipo de cliente (como persona individual, propietario único, sociedad o

corporación) con el fin de resolver adecuadamente una correspondencia para cada tipo. En la figura 7-6 se expone el modelo lógico de los datos desarrollado con base en los nuevos requisitos.

En la notación II, el tipo o categoría se indica mediante un símbolo que parece un círculo con una línea debajo. Por lo tanto, usted sabe que Cliente individual y Cliente comercial son tipos secundarios de Cliente, por el símbolo que aparece en la línea que los conecta. Observe también que comparten la misma clave principal exacta y que, en los tipos secundarios, la clave principal de la entidad también es una clave externa para la entidad de supertipo. Esto es muy congruente cuando analiza el hecho de que una entidad Cliente individual *es* un Cliente, lo que significa que cualquier ocurrencia de la entidad Cliente individual tendría una tupla en la correlación Cliente, al igual que una tupla equivalente en la entidad Cliente individual.

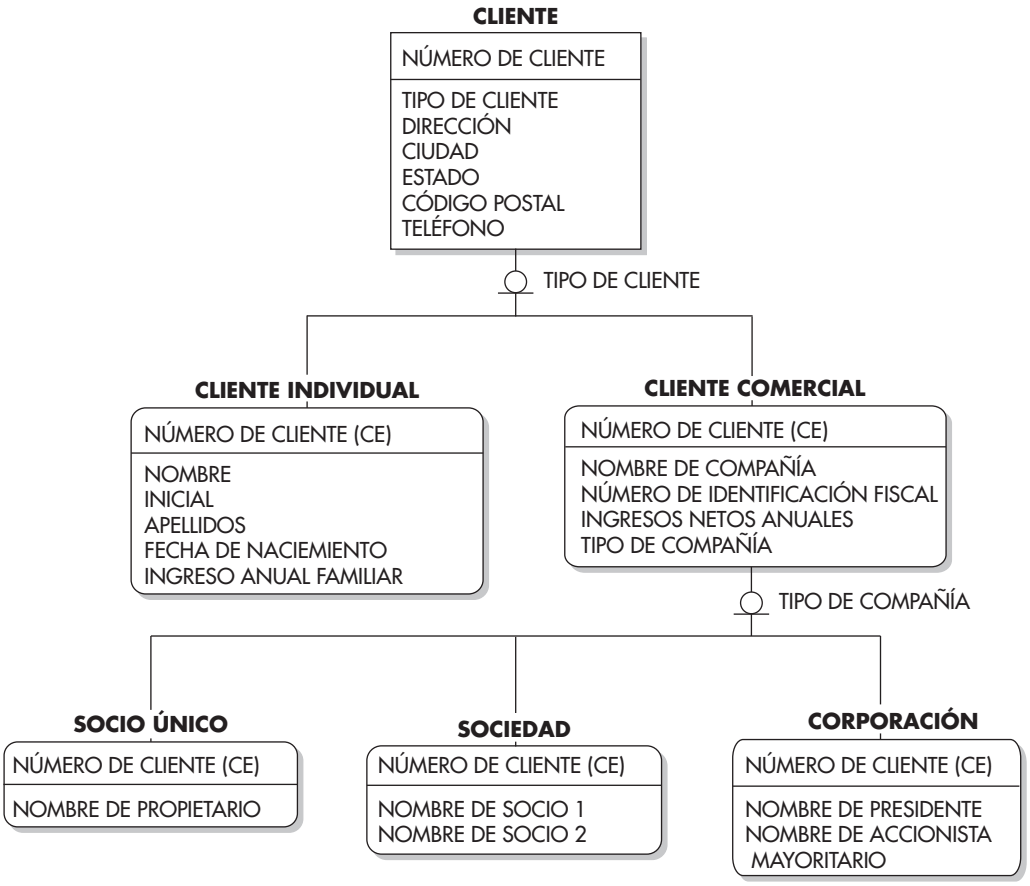


Figura 7-6 Las clases secundarias de Cliente.

Por lo general, un atributo en la entidad supertipo indica cuál tipo secundario se asigna a cada ocurrencia (tupla) de una entidad. Una vez que esto se implementa en las tablas, los usuarios de la base de datos emplean el atributo de tipo para saber dónde buscar (es decir, cuál tabla de tipo secundario contiene) el resto de la información sobre cada ocurrencia de la entidad (cada fila). A ese atributo se le denomina *discriminador de tipo* y se nombra junto al símbolo de tipo en el ERD. Por lo tanto, Tipo de cliente es el discriminador de tipo que indica si un Cliente específico es un Cliente individual o un Cliente comercial. Asimismo, Tipo de compañía es el discriminador de tipo que indica si un Cliente comercial determinado es un propietario único, una sociedad o una corporación.

Como puede imaginar, esta notación II no es el único formato utilizado en los ERD para los supertipos y los tipos secundarios. Sin embargo, es el método de uso más común. Otro formato popular consiste en dibujar las entidades de tipo secundario dentro de la entidad de supertipo (es decir, rectángulos de la entidad de tipo secundario dibujados dentro del rectángulo de la entidad de supertipo correspondiente). Aunque este formato muestra con claridad que los tipos secundarios en realidad son sólo una parte del supertipo, tiene limitaciones prácticas cuando las entidades se dividen en muchos niveles.

Como ya se mencionó, hallar el nivel correcto de especialización es un desafío importante en el diseño de una base de datos. Al revisar el diseño lógico propuesto en la figura 7-6, el equipo de diseño de la base de datos observó algo: la única diferencia entre los tipos secundarios Socio único, Sociedad y Corporación es el modo en que aparecen como atributos los nombres de las personas importantes en esos tipos de compañías. Además, el uso de dos atributos casi idénticos para los nombres de los copropietarios en el tipo secundario Sociedad puede considerarse un atributo repetido y, por lo tanto, una violación a la primera forma normal. El equipo de diseño optó por generalizar estos nombres en la entidad Cliente comercial pero, al hacerlo, reconoció los problemas en la primera forma normal y decidió colocarlos en una relación separada llamada Cliente comercial principal. Esto condujo al ERD mostrado en la figura 7-7.

Es evidente que este diseño más sencillo producirá menos tablas cuando se implemente físicamente. Ofrece una ventaja muy grande, porque no sólo no hay pérdida de funciones cuando se consolidan los tipos secundarios en el supertipo, sino que en realidad se tienen *más* funciones disponibles porque es posible agregar todos los nombres que se desee a cualquier tipo de cliente comercial.

Un análisis adicional del equipo de diseño lo ayudó a comprender la semejanza entre los atributos de nombre incluidos ahora en la entidad Cliente comercial principal y los incluidos en la entidad Cliente individual. Al analizar todavía más opciones con el Departamento de Servicio al Cliente, el equipo de diseño descubrió algunos casos en que sería conveniente que se registraran varios nombres de contacto para los clientes individuales, al igual que para los clientes comerciales. Por ejemplo, los clientes que tienen desacuerdos legales suelen solicitar que todo contacto sea a través de un abogado. Con esa información, el equipo de diseño deci-

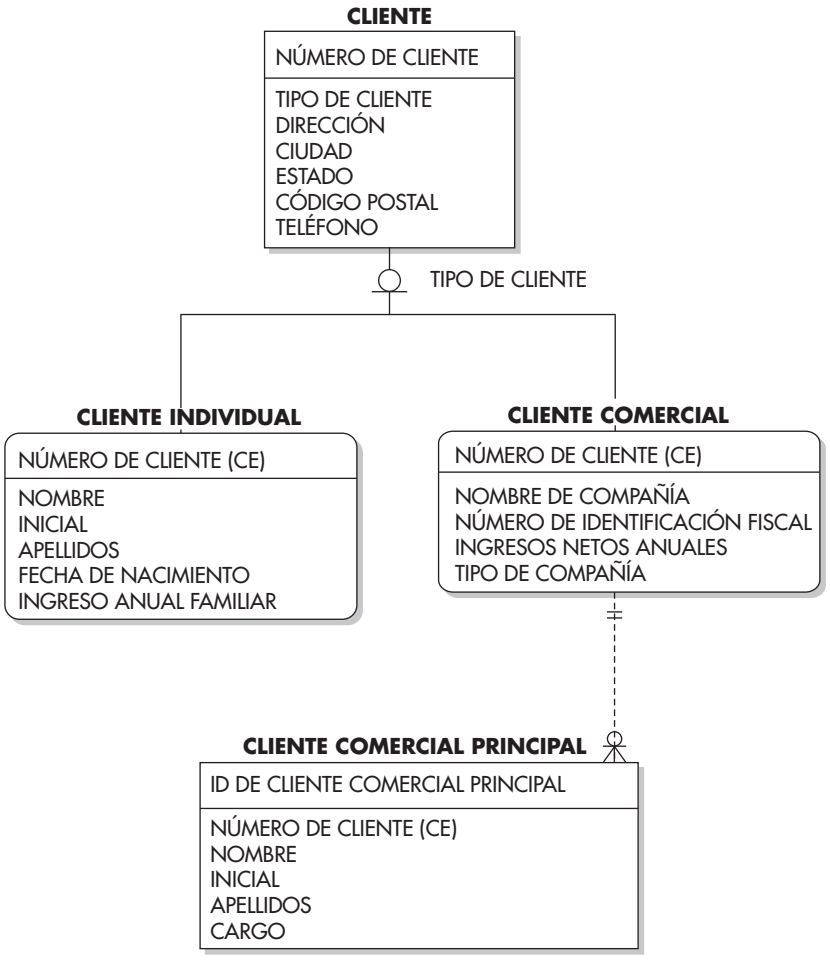


Figura 7-7 Tipos secundarios de Cliente, versión 2.

dió generalizar estos nombres y trasladó Cliente comercial principal para que fuera elemento secundario de Cliente y lo denominó Contacto de cliente, de modo que pudiera utilizarse para contener la información acerca de un director (propietario, copropietarios, socio, funcionario) del cliente o de cualquier otro contacto de éste que resulte útil para el Departamento de Servicio al cliente. El equipo de diseño comprendió además que los nombres de contactos serían más útiles si se incluyera un número telefónico. El atributo Teléfono se dejó en la entidad Cliente porque estaba diseñado para contener el número telefónico general del cliente. El número telefónico en la entidad Contacto del cliente está diseñado para contener el teléfono de un contacto individual. El diseño lógico resultante se aprecia en la figura 7-8.

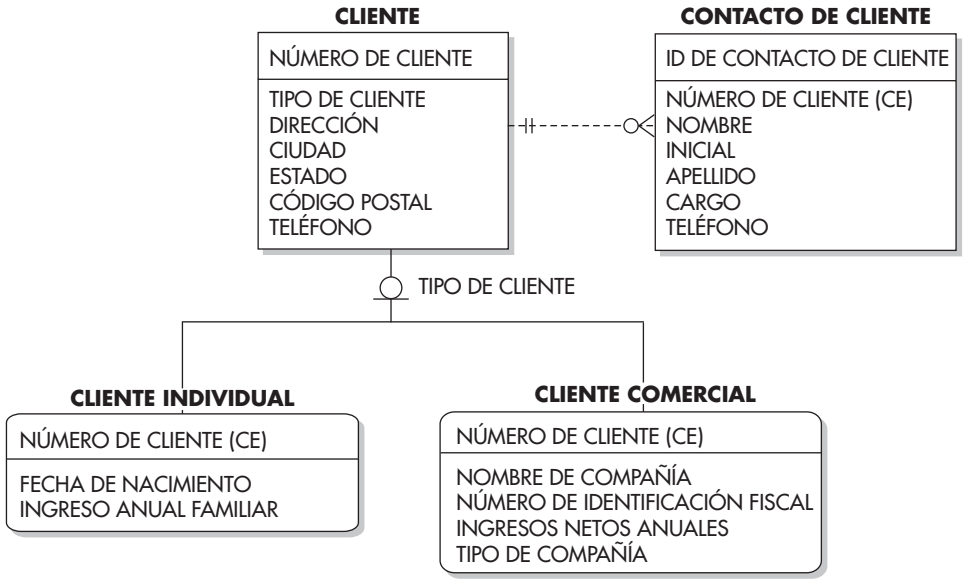


Figura 7-8 Tipos secundarios de Cliente, versión 3.

El hecho de que los tres diseños presentados sean funcionales (figuras 7-6, 7-7 y 7-8) debe subrayar el dilema de comparar generalización con especialización: no existe una respuesta "correcta". Por tal razón, el arte de diseñar una base de datos consiste en llegar al diseño que se adapte mejor a lo que se sabe sobre los usos esperados de la base de datos. Esto se consigue mejor al comparar las ventajas y desventajas relativas de cada diseño alterno. Y no existe mejor vehículo para comunicar las opciones que el ERD.

Pregunta al experto

P: **En el análisis del UML en el capítulo, mencionó que la generalización se presenta mediante líneas y flechas. ¿Cómo convertiría el supertipo y los tipos secundarios de la figura 7-8 a notación UML?**

R: En primer lugar, se eliminan los símbolos existentes (las líneas, el símbolo de tipo y el nombre del atributo discriminador). Después se dibuja una línea entre cada tipo secundario y su supertipo, y se pone una punta de flecha vacía en el extremo del supertipo de la línea, que apunte hacia la entidad de supertipo. En la figura 7-9 se presenta el mismo modelo que en la figura 7-8, pero con notación UML.

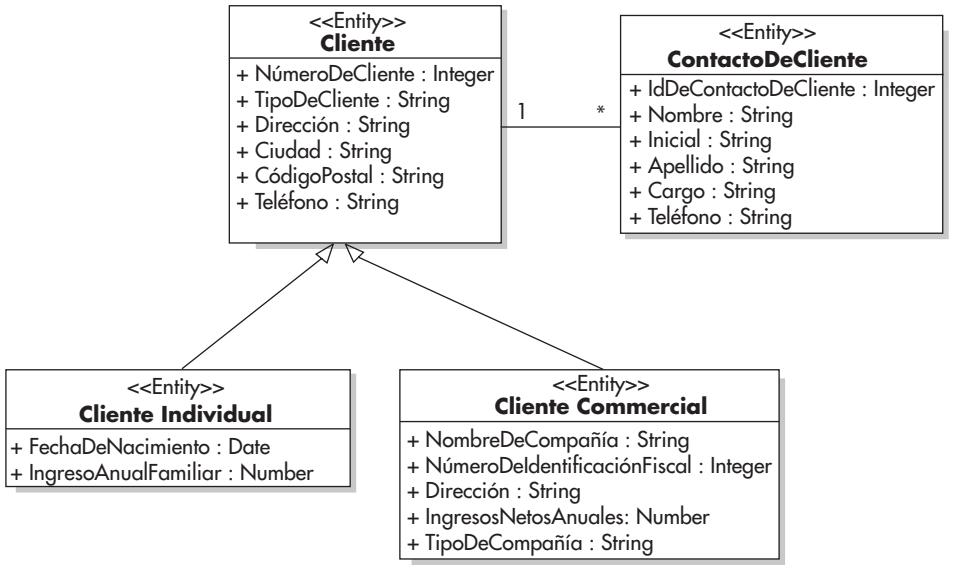


Figura 7-9 Tipos secundarios de Cliente, versión 3, convertidos a UML.

Lineamientos para trazar ERD

Éstos son los lineamientos generales que se aplican para crear ERD:

- No intente relacionar cada entidad con otra entidad. Las entidades sólo deben relacionarse cuando la clave principal *completa* de una entidad aparece como clave externa de otra.
- Excepto para los tipos secundarios, evite relaciones que impliquen más de dos entidades. Aunque dibujar menos líneas puede parecer más sencillo, es muy común que se malinterpreten las relaciones dibujadas de una entidad primaria con varias entidades secundarias al emplear una sola línea.
- Aplique con uniformidad los nombres de entidades y atributos. Desarrolle una nomenclatura y apéguese a ella.
- Emplee abreviaturas en los nombres sólo cuando sea absolutamente necesario y, en esos casos, aplique una lista estándar de abreviaturas.
- Asigne con uniformidad los nombres de claves principales y externas. Casi todos los expertos prefieren que una clave externa tenga exactamente el mismo nombre que la clave principal.
- Cuando asigne nombres a las relaciones, busque palabras de acción, y evite términos no descriptivos como “tiene”, “pertenece a”, “se asocia con” y otros por el estilo.

Modelos de procesos

Como ya se mencionó, el diseño de procesos rara vez es responsabilidad del diseñador de una base de datos o DBA, pero comprender los fundamentos le ayuda a comunicarse con los diseñadores de procesos y asegurar que el diseño de la base de datos permita ese diseño. Por lo tanto, en esta sección se presenta un breve examen de las técnicas comunes para diagramar modelos de procesos. Si quiere más detalles acerca de estas u otras técnicas de modelado de procesos, consulte un libro sobre análisis y diseño de sistemas.

Durante toda esta sección, se emplea como ejemplo el proceso de cumplimiento de pedidos de Industrias Acme, un proceso de negocios muy sencillo. Este proceso tiene los pasos siguientes:

1. Buscar todos los pedidos no enviados en la base de datos.
2. Para cada pedido, hacer lo siguiente:
   - Comprobar el inventario disponible. Si no existe inventario suficiente para el pedido, pasar al siguiente pedido.
   - Corroborar el crédito del cliente para asegurar que no excede su límite ni tiene otros problemas de crédito, como pagos vencidos. Esto suele ocurrir cuando se hace el pedido, pero necesita ocurrir aquí otra vez porque la situación crediticia de un cliente con Industrias Acme puede cambiar en cualquier momento. Si se encuentra un problema de crédito, pasar al pedido siguiente.
   - Generar los documentos requeridos para empacar y enviar el pedido (material de empaque, etiquetas de envío, etc.) y trasladarlos al departamento de envíos.
   - Cuando el departamento de envíos ha concluido con el pedido, generar la factura para el pedido y cobrar al cliente lo que corresponde.

Es obvio que este proceso puede ser mucho más complicado en una empresa grande, pero aquí se ha reducido a lo básico para que sea más fácil ilustrar los modelos de procesos.

El diagrama de flujo

Tal vez, el diagrama de flujo (o diagrama de estructuras) es la forma más antigua de documentación de sistemas computacionales. Algunos creen que los diagramas de flujo existían cuando los dinosaurios dominaban nuestro planeta y, por lo tanto, quienquiera que todavía los utilice es un dinosaurio. Al margen de polémicas, los diagramas de flujo suelen ser considerados anticuados, pero todavía tienen mucho por ofrecer en ciertas circunstancias y se usan en muchos lugares. En la figura 7-10 se presenta el diagrama de flujo del proceso de cumplimiento de pedidos de ejemplo.

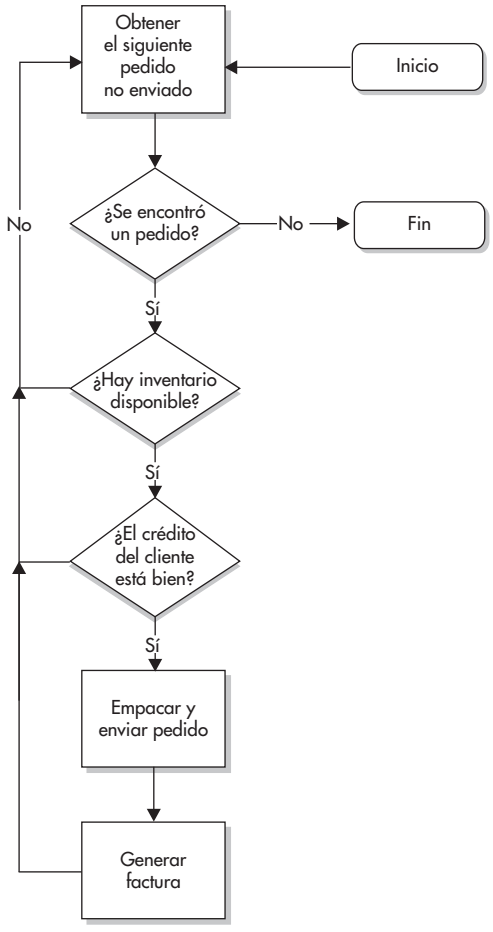


Figura 7-10 Flujograma del proceso de cumplimiento de pedidos de Industrias Acme.

Éstos son los componentes básicos del diagrama de flujo:

- Los pasos del proceso se muestran con rectángulos.
- Los puntos de decisión se presentan con rombos. En cada punto de decisión, las bifurcaciones lógicas se basan en el resultado de la decisión. Por ejemplo, una decisión puede ser "¿Es viernes hoy?" y un resultado "Sí" indica una dirección, y un resultado "No" señala otra dirección.
- Las líneas con flechas muestran el flujo del control a través del diagrama. Cuando concluye un proceso, entrega el control al siguiente proceso o punto de decisión.

- Los puntos Inicio y Fin se exponen como elipses o rectángulos redondeados. Los diagramas de flujo sirven para mostrar los procesos perpetuos que no tienen inicio y fin, pero se emplean más a menudo para mostrar procesos finitos con puntos específicos de inicio y fin.
- Se emplean símbolos conectores que parecen las bases en un campo de béisbol (y no se aprecian en la figura 7-10) para conectar líneas a los procesos o puntos de decisión, en la misma página o en otra. Suele incluirse una letra de referencia y se supone una línea de flujo de control entre cualesquiera dos conectores que tienen la misma letra.

En la figura 7-10 se muestra el flujo de un proceso de bucle sencillo. Comienza con un paso del proceso que va al siguiente pedido no enviado de la base de datos. Se agregó una decisión después de esto para detener el bucle (concluye el flujo) si no se encuentra un pedido no enviado. Si existe ese pedido, el proceso continúa con puntos de decisión que comprueban la disponibilidad del inventario y un crédito aceptable del cliente, con un resultado "No" de vuelta al inicio del bucle (el proceso "Ir al siguiente pedido no enviado"), lo que en esencia "salta" el pedido y avanza al siguiente. Si se obtiene un resultado "Sí" de todos los puntos de decisión, a continuación se invoca el proceso "Empacar y enviar pedido", seguido de "Generar factura". Una vez que concluye el proceso "Generar factura", el control regresa a "Ir al siguiente pedido no enviado" al inicio del bucle. El bucle continúa hasta que ya no se detectan pedidos no enviados.

Los diagramas de flujo tienen las ventajas siguientes:

- Para los programadores de lenguajes de procedimientos resulta fácil aprender a usarlos. Un *lenguaje de procedimientos* es un lenguaje de programación mediante el cual el programador debe describir los pasos de un proceso requeridos para hacer algo, en contraste con un *lenguaje que no es de procedimientos*, como SQL, con el que el programador sólo describe los resultados deseados. Es probable que el lenguaje de procedimientos de uso más frecuente en la actualidad sea C y sus variantes (C++, C# y demás), pero otros, como FORTRAN y COBOL, todavía se usan un poco. Asimismo, se emplean con gran frecuencia lenguajes de procedimientos especializados para las bases de datos relacionales, entre ellos PL/SQL para Oracle y Transact SQL para Sybase Ase y Microsoft SQL Server.
- Los diagramas de flujo se aplican a procedimientos fuera de un contexto de programación. Por ejemplo, se suelen emplear diagramas de flujo para conducir a los técnicos de reparaciones por los procedimientos de detección y solución de problemas del equipo que atienden.
- Los diagramas de flujo son útiles para detectar componentes reutilizables (comunes). El diseñador puede hallar con facilidad cualquier proceso que aparece varias veces en el diagrama de flujo de un sistema de aplicaciones específico.
- Los diagramas de flujo se modifican con facilidad y pueden evolucionar conforme cambian los requisitos.

Por otra parte, los diagramas de flujo tienen estas desventajas:

- No se pueden aplicar a lenguajes que no son procedimentales u orientados a objetos.
- No modelan con facilidad ciertas situaciones, como los procesos recursivos (procesos que se invocan a sí mismos).

El diagrama de jerarquía de funciones

El *diagrama de jerarquía de funciones*, como lo indica su nombre, muestra todas las funciones de un sistema de aplicaciones o un proceso de negocios específico, organizadas en un árbol jerárquico. En la figura 7-11 se muestra este tipo de diagrama de modelado de procesos, a partir del proceso de cumplimiento de pedidos de ejemplo.

Debido a que la jerarquía de funciones de un solo proceso no tiene mucho sentido fuera de contexto, se han agregado otros dos procesos a la jerarquía: Recepción de pedido y Administración de historial. Para ser eficaz, una jerarquía de funciones debe contener *todos* los procesos necesarios para efectuar la función que describe. En la figura 7-11 se intenta mostrar todos los procesos requeridos para la función de administración de pedidos en Industrias Acme. Se pretende que la recepción de pedidos cubra todos los pasos del proceso relacionado con la colocación de un pedido por parte de un cliente, y con el registro del pedido en la base de datos de Acme. La administración del historial está diseñada para cubrir todos los pasos requeridos para archivar de manera permanente, purgar los pedidos antiguos (históricos) y generar cualquier informe sobre un historial de pedidos. Estos dos procesos necesitan expandirse mediante la adición de pasos al proceso que está debajo de ellos (tal como se hizo con Cumplimiento de pedido) para formar este diagrama completo. Bajo Cumplimiento de pedido, se han incluido los cuatro pasos principales del proceso relacionado con cumplir los pedidos.

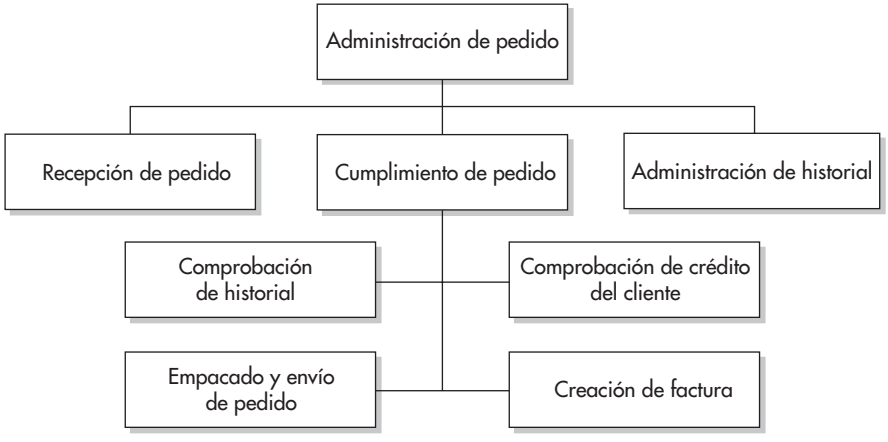


Figura 7-11 Jerarquía de funciones del proceso de cumplimiento de pedidos de Acme.

Éstas son las ventajas de los diagramas de jerarquía de funciones:

- Son fáciles de aprender y utilizar.
- Documentan con rapidez gran parte de la función (resuelven 80% de los procesos rápidamente).
- Ofrecen un buen compendio en niveles de detalle alto e intermedio.

Algunas desventajas de los diagramas de jerarquía de funciones son:

- La comprobación de la calidad resulta difícil y subjetiva.
- No manejan interacciones complejas entre las funciones.
- No muestran con claridad la secuencia de pasos del proceso ni las dependencias entre los pasos.
- No son un recurso de presentación eficaz para las jerarquías grandes o en niveles muy detallados.

El diagrama de carriles de alberca

El *diagrama de carriles de alberca* debe su nombre a los carriles verticales del diagrama, que se parecen a los de una alberca. Cada carril representa una unidad organizativa, como un departamento, y se ubican pasos del proceso en el carril de la unidad responsable de ese paso. Las líneas con flechas muestran la secuencia o flujo de control de los pasos del proceso. En la figura 7-12 se muestra el diagrama de carriles de alberca del proceso de cumplimiento de pedidos de ejemplo.

Algunas ventajas del diagrama son:

- Posee una incomparable capacidad para mostrar quién hace qué en la organización.
- Es excelente para identificar ineficiencias en los procesos existentes y se presta bien para los esfuerzos de reingeniería de los procesos empresariales.

Entre sus desventajas están:

- No representa bien procesos complicados (los que tienen muchos pasos o dependencias complejas entre los pasos).
- No muestra el manejo de errores y excepciones.

El diagrama de flujo de datos

El *diagrama de flujo de datos (DFD)* es el más centrado en los datos de todos los diagramas de procesos. En lugar de mostrar un flujo del control a través de una serie de pasos del proceso, se concentra en los datos que fluyen por los pasos del proceso. Al combinar diagramas de

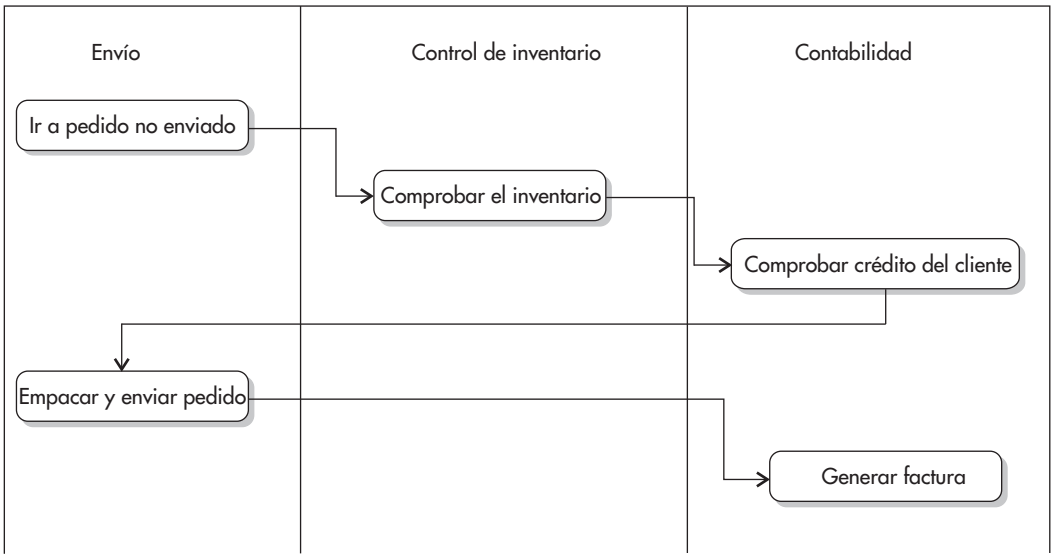


Figura 7-12 Un diagrama de carriles de alberca del proceso de cumplimiento de pedidos de Industrias Acme.

manera jerárquica, el DFD combina lo mejor del diagrama de flujo y del diagrama de funciones. Los DFD se volvieron muy populares a fines de la década de 1970 y principios de la de 1980, en gran parte debido al trabajo de Chris Gane y Trish Sarson. Cada proceso en un DFD se puede dividir con otra página completa hasta alcanzar el nivel de detalle que se prefiera. En la figura 7-13 se muestra una página del DFD del proceso de cumplimiento de pedidos de Industrias Acme.

Los componentes de un DFD son sencillos:

- Los procesos se representan con rectángulos redondeados y se suelen numerar de manera jerárquica. La primera página de un DFD puede tener procesos numerados 1, 2, 3 y 4. La página siguiente puede dividir el proceso número 1 y tener procesos numerados 1.1, 1.2, etc. Si el proceso 1.2 se dividiera en otra página, los procesos de esa página se numerarían 1.2.1, 1.2.2, etcétera.
- Los almacenes de datos son representados mediante un rectángulo con un extremo abierto. Un *almacén de datos* es una representación genérica de los datos que se han vuelto persistentes al haberse almacenado en algún otro lugar, como un archivo, una base de datos o incluso un documento impreso. El término se eligió porque no está implícito ningún tipo de almacenamiento específico. Debido a que ya existe un ERD del ejemplo, los almacenes de datos se deben alinear estrechamente con las entidades identificadas.

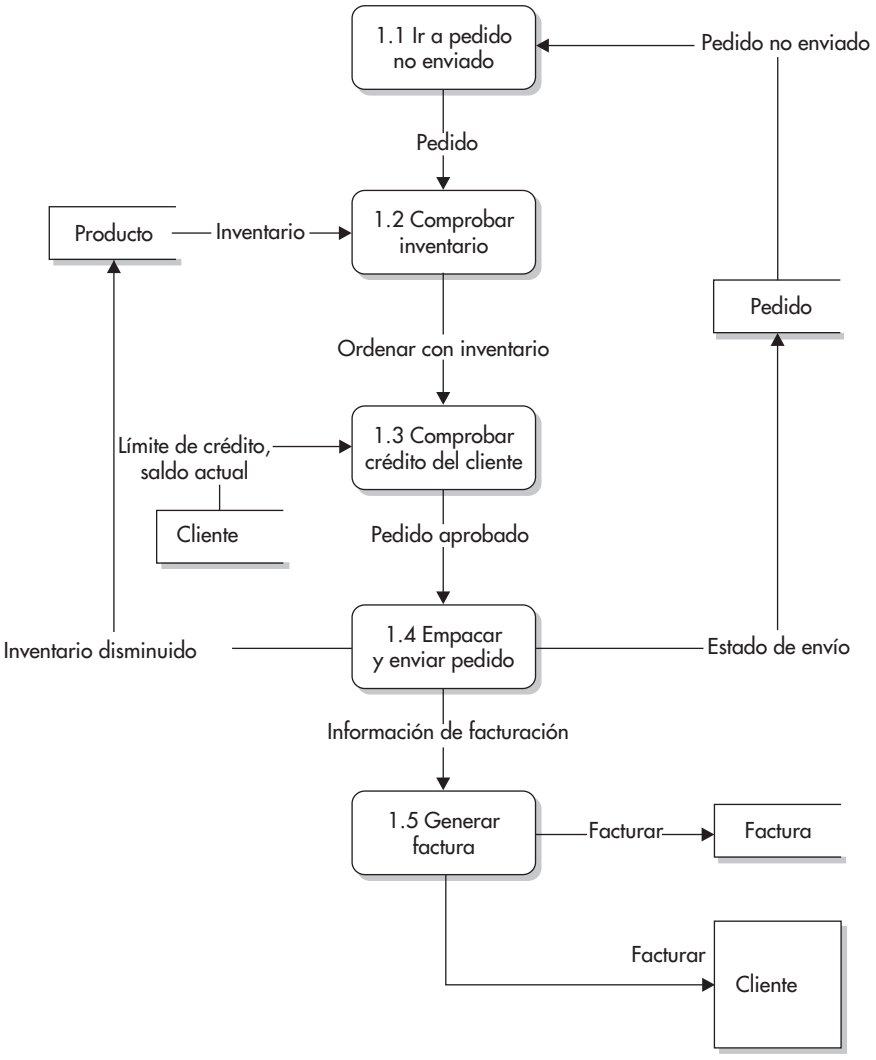


Figura 7-13 Una página de diagrama de flujo de datos del proceso de cumplimiento de pedidos de Industrias Acme.

- Los orígenes y destinos de los datos (las entidades externas en terminología relacional) se presentan mediante cuadros. En la figura 7-13 se muestra que el cliente es el destino del flujo de datos de una factura (además de un almacén de datos local que contendrá los datos de la factura). Trate de no confundir los flujos de datos con los flujos de materiales. Sí, la factura se imprime y envía al cliente, pero el flujo de datos intenta mostrar que los *datos* se envían al cliente sin considerar el medio utilizado para hacerlo.

- Los flujos de datos se muestran mediante líneas con puntas de fecha que indican la dirección del flujo. Sobre cada flujo, se emplean palabras para describir el contenido de los datos que se envían. Se permiten flujos bidireccionales, pero se suelen mostrar como flujos separados, porque los datos rara vez son exactamente iguales en ambas direcciones.

Entre las ventajas del diagrama de flujo de datos están:

- Muestra con facilidad la estructura general del sistema sin sacrificar detalles (que se presentan en las páginas posteriores que se expanden en los procesos del siguiente nivel).
- Es conveniente para el trabajo de diseño de lo general a lo particular.
- Es bueno para presentar los diseños de sistemas a la administración y a los usuarios de negocios.

Éstas son algunas desventajas del diagrama de flujo de datos:

- Se requieren bastante tiempo y mano de obra para desarrollar sistemas complejos.
- Se ha comprobado que el diseño de lo general a lo particular no es eficaz para situaciones en que los requisitos son modificables y evolucionan constantemente durante la vida del proyecto.
- Es deficiente para mostrar la lógica compleja, pero los diagramas de nivel más bajo se complementan fácilmente con otros documentos, como descripciones o tablas de decisión.

Pregunta al experto

P: **¿El UML no tiene diagrama de procesos?**

R: Sí, pero ésa es sólo una parte de la historia. Como ya se mencionó, UML 2.x ofrece 13 diagramas diferentes; seis de ellos son *diagramas de estructuras* que hacen hincapié en las cosas que deben estar en el sistema que se modela y siete son *diagramas de comportamiento* que se concentran en lo que debe ocurrir en el sistema que se modela. De éstos, ya se cubrió el diagrama de clases en este capítulo. No hay suficiente espacio en el libro para cubrirlos todos,

(*continúa*)

pero encontrará mucha información en Internet y en libros sobre el tema. En la tabla siguiente se ofrece una descripción resumida de cada diagrama de UML:

Tipo	Nombre	Descripción
Estructura	Diagrama de clases	Muestra un conjunto de elementos de un modelo estático como clases y tipos, su contenido y sus relaciones.
Estructura	Diagrama de componentes	Muestra los componentes que forman una aplicación, sistema o empresa.
Estructura	Diagrama de estructura compuesta	Describe que la estructura interna de un clasificador (como una clase, un componente o un caso de uso), incluye la interacción del clasificador que apunta a otras partes del sistema (agregadas en UML 2.x).
Estructura	Diagrama de despliegue	Expone la arquitectura de ejecución de los sistemas, lo que incluye nodos, entornos de hardware/software, y el middleware que los conecta.
Estructura	Diagrama de objetos	Representa los objetos y sus relaciones en un punto en el tiempo.
Estructura	Diagrama de paquetes	Muestra cómo se integran en paquetes los elementos del modelo, al igual que las dependencias entre los paquetes.
Comportamiento	Diagrama de actividades	Exhibe los procesos empresariales de alto nivel, entre ellos el flujo de datos.
Comportamiento	Diagrama de máquina de estado	Describe los estados en que puede estar un objeto o una interacción, y las transiciones entre los estados.
Comportamiento	Diagrama de casos de uso	Expone a los actores, los casos de uso, y sus interacciones.
Comportamiento	Diagrama de comunicaciones	Señala las instancias de clases, sus interrelaciones y el flujo de mensajes entre ellas.
Comportamiento	Diagrama de panorama de interacción	Una variante de un diagrama de actividades que representa un panorama del flujo de control dentro de un sistema o proceso empresarial (agregado en UML 2.x).
Comportamiento	Diagrama de secuencias	Expone el orden de los tiempos de los mensajes entre los clasificadores, lo que muestra la lógica secuencial del sistema.
Comportamiento	Diagrama de tiempos	Representa el cambio de condición o estado de una instancia de un clasificador o su función en el tiempo (agregado en UML 2.x).

Observe que algunas referencias muestran un tipo secundario de diagrama de interacción bajo diagrama de comportamiento, que contiene los diagramas de secuencias, de panorama de interacción, de comunicaciones y de tiempos.

Determinación de relaciones entre entidades y procesos

Una vez que el diseñador de una base de datos ha concluido el diseño lógico y un ERD propuestos y, al mismo tiempo, los diseñadores de procesos han finalizado su modelo de procesos, ¿cómo se puede confiar en que las dos partes colaborarán para resolver los problemas de negocios que se supone que el nuevo proyecto habrá de atender? Una parte de la respuesta estriba en una técnica de graficación diseñada para mostrar cómo interactúan las entidades y los procesos, conocida como la matriz CRUD.

CRUD son las siglas en inglés de Create, Read, Update y Delete (crear, leer, actualizar y eliminar), que son las letras utilizadas en el cuerpo del diagrama. El concepto de la matriz CRUD es muy sencillo:

- Un eje de la matriz representa los procesos principales del sistema de aplicaciones.
- El otro eje designa a las entidades principales utilizadas por el sistema de aplicaciones.
- En cada celda de la matriz, se escribe la combinación adecuada de letras:
  - *C*, si el proceso crea ocurrencias nuevas de la entidad.
  - *R*, si el proceso lee información sobre una entidad desde un origen de datos.
  - *U*, si el proceso actualiza uno o más atributos para la entidad.
  - *D*, si el proceso elimina ocurrencias de la entidad.

He aquí una matriz CRUD de ejemplo para la función de administración de pedidos en Industrias Acme, que sigue los principales procesos presentados en el diagrama de jerarquía de funciones (ver la figura 7-11). Para que sea eficaz, la matriz debe mostrar sólo los procesos de alto nivel y las entidades de supertipo. Los detalles excesivos oscurecen el efecto del diagrama.

	ENTIDAD: Producto	**Pedido**	**Cliente**	**Factura**
PROCESO: Recepción de pedido	R	CRU	RU	
Cumplimiento de pedido	RU	RU	R	C
Administración del historial		RD	R	

La matriz CRUD es valiosa para verificar la consistencia de los diseños de procesos y de datos (entidades). De un vistazo, se detectan estos posibles problemas:

- Entidades que no tienen un proceso de creación.
- Entidades que no tienen un proceso de eliminación.

- Entidades que nunca son actualizadas.
- Entidades que nunca son leídas.
- Procesos que eliminan o actualizan entidades sin leerlas.
- Procesos que sólo leen (no hay acciones de creación, eliminación o actualización).

El ejemplo tiene varios problemas, lo que sólo demuestra que el diseño de procesos está incompleto (es decir, es probable que falten algunos procesos importantes del sistema de aplicaciones). En la conclusión de la base del diseño lógico de un proyecto, la matriz CRUD es un excelente vehículo para una revisión final del trabajo completado. El paso siguiente en el ciclo de vida de una base de datos consiste en concluir su diseño físico, que se analiza en el capítulo 8.

Pruebe esto 7-1 Dibujo de un ERD en un formato de ingeniería de la información (II)

En este ejercicio dibujará un ERD que compruebe casi todos los conceptos presentados hasta este momento, entre ellos entidades (tablas), relaciones, relaciones recursivas, supertipos y tipos secundarios. Puede dibujarlo como un modelo lógico o físico. No obstante, mi solución está en la forma de un modelo físico y, por lo tanto, las instrucciones emplearán los términos del modelo físico (como tabla y columna).

Paso a paso

1. Dibuje una tabla para PERSONA con columnas ID_DE_PERSONA (clave principal), NOMBRE, APELLIDOS, FECHA_DE_NACIMIENTO y GÉNERO. Deje espacio para otras dos columnas, que agregará en el paso siguiente.
2. Dibuje dos relaciones recursivas uno a varios: una para el padre y una para la madre de la persona. Recuerde que las relaciones recursivas tienen la misma tabla como padre e hijo. En este caso, las relaciones deben ser opcionales en ambas direcciones porque no tendrá los padres de cada persona en la base de datos y no todas las personas tienen hijos. La tabla PERSONA necesitará dos claves externas para permitir las relaciones recursivas: una para Id de persona del padre y otra para Id de persona de la madre.
3. Dibuje una tabla dependiente llamada MATRIMONIO con columnas ID_DE_PERSONA_1, ID_DE_PERSONA_2, FECHA_DE_MATRIMONIO y FECHA_DE_FINAL. La clave principal debe estar formada con las primeras tres columnas para que sea única bajo todas las circunstancias. ID_DE_PERSONA_1 e ID_DE_PERSONA_2 serán las claves externas de las dos personas que están casadas.

4. Dibuje dos relaciones uno a varios de PERSONA a MATRIMONIO: una en que ID_DE_PERSONA_1 sea la clave externa y otra en que ID_DE_PERSONA_2 sea la clave externa. Estas relaciones son obligatorias-opcionales (cada matrimonio debe tener dos personas, pero algunas personas nunca se casan).
5. Dibuje una tabla EMPLEADO con columnas ID_DE_PERSONA (clave principal), ID_DE_EMPLEADO, FECHA_DE_CONTRATACIÓN y FECHA_DE_FINAL.
6. Dibuje una tabla CLIENTE con columnas NÚMERO_DE_CLIENTE (clave principal), NOMBRE, DIRECCIÓN, CIUDAD, ESTADO, CÓDIGO_POSTAL y TELÉFONO.
7. Dibuje una tabla CONTACTO_DE_CLIENTE con columnas ID_DE_PERSONA (clave principal) e ID_CLIENTE.
8. Dibuje las líneas y símbolos necesarios para que EMPLEADO y CONTACTO_DE_CLIENTE sean tipos secundarios de PERSONA.
9. Trace una relación uno a varios obligatoria-opcional de CLIENTE a CONTACTO_DE_CLIENTE, y haga que ID_DE_CLIENTE y CONTACTO_DE_CLIENTE sean claves externas.

Resumen de Pruebe esto

En este ejercicio creó cinco tablas y cinco relaciones (dos de ellas recursivas) y obtuvo dos tipos secundarios de tablas de otra tabla. Mi solución aparece en el apéndice B.

✓ Autoexamen Capítulo 7

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. ¿Por qué es importante que un diseñador de base de datos comprenda el modelo de procesos?
   - **A** El diseño de procesos es principalmente una responsabilidad del DBA.
   - **B** El modelo de procesos debe ser concluido antes del modelo de datos.
   - **C** El modelo de datos debe ser concluido antes del modelo de procesos.
   - **D** El diseñador de una base de datos debe colaborar estrechamente con el diseñador de procesos.
   - **E** El diseño de una base de datos debe permitir el modelo de procesos buscado.

2. El formato de ERD de Peter Chen representa "varios" con _____.

3. El rombo en el formato de Peter Chen representa _____.

4. El formato de ERD relacional representa "varios" con _____.

5. El formato ERD de IDEF1X

- **A** Fue publicado en 1983.
- **B** Sigue una norma desarrollada por el National Institute of Standards and Technology.
- **C** Tiene muchas variantes.
- **D** Ha sido adoptado como una norma por muchas dependencias gubernamentales de Estados Unidos.
- **E** Cubre los modelos de datos y de procesos.

6. El formato de ERD de IDEF1X muestra

- **A** Relaciones de identificación con una línea continua.
- **B** La cardinalidad mínima con una combinación de círculos pequeños y líneas verticales sobre la línea de una relación.
- **C** La cardinalidad máxima con una combinación de pequeñas líneas verticales y una pata de gallo dibujadas sobre la línea de una relación.
- **D** Las entidades dependientes con un rectángulo de esquinas cuadradas.
- **E** Las entidades independientes con un rectángulo de esquinas redondeadas.

7. Un tipo secundario

- **A** Es un subconjunto de un supertipo.
- **B** Tiene una relación uno a varios con el supertipo.
- **C** Posee una relación uno a uno condicional con el supertipo.
- **D** Muestra diversos estados del supertipo.
- **E** Es un superconjunto de un supertipo.

8. Algunos ejemplos de tipos secundarios posibles de un supertipo de la entidad Pedido son

- **A** Artículos de línea de pedido.
- **B** Pedido enviado, pedido no enviado, pedido facturado.

C Pedido de artículos de oficina, pedido de servicios profesionales.

D Pedido aprobado, pedido pendiente, pedido cancelado.

E Pedido de pieza de automóvil, pedido de piezas de aeronave, pedido de piezas de camión.

9. En la notación II, los tipos secundarios

A Se muestran con un nombre de atributo discriminador de un tipo.

B Se conectan al supertipo mediante un símbolo formado por un círculo con una línea debajo.

C Tienen la clave principal del tipo secundario mostrado como clave externa en el supertipo.

D Suelen tener la misma clave principal que el supertipo.

E Se señalan mediante una pata de gallo.

10. Cuando se consideran los tipos secundarios del diseño de una base de datos, existe una solución intermedia entre _____ y _____.

11. En un diagrama de flujo, los pasos de un proceso se muestran como _____ y los puntos de decisión se presentan como _____.

12. Éstas son algunas ventajas de los diagramas de flujo

A Son naturales y fáciles de usar para programadores de lenguajes de procedimientos.

B Son útiles para detectar componentes reutilizables.

C Sólo son específicos para la programación de aplicaciones.

D También son útiles para los lenguajes que no son de procedimientos y orientados a efectos.

E Se pueden modificar fácilmente conforme cambian los requisitos.

13. Los componentes básicos de un diagrama de jerarquía de funciones son

A Elipses para mostrar atributos.

B Rectángulos para mostrar funciones de procesos.

C Líneas que conectan los procesos en su orden de ejecución.

D Una jerarquía para presentar cuáles funciones están subordinadas a otras.

E Rombos para indicar puntos de decisión.

14. Las ventajas de un diagrama de jerarquía de funciones son

- **A** La comprobación de la calidad es fácil y directa.
- **B** Se modelan fácilmente las interacciones complejas entre las funciones.
- **C** El aprendizaje de su uso es rápido y fácil.
- **D** Muestra con claridad la secuencia de pasos de un proceso.
- **E** Proporciona un buen panorama de los niveles de detalle alto e intermedio.

15. Los componentes básicos de un diagrama de carril de alberca son

- **A** Líneas con flechas para mostrar la secuencia de pasos de un proceso.
- **B** Rombos para señalar puntos de decisión.
- **C** Carriles verticales para exponer las unidades organizacionales que efectúan los pasos de un proceso.
- **D** Elipses para indicar los pasos de un proceso.
- **E** Rectángulos con un extremo abierto para exhibir los almacenes de datos.

16. El diagrama de flujo de datos (DFD)

- **A** Es el más centrado en los datos de todos los modelos de procesos.
- **B** Fue desarrollado en la década de 1980.
- **C** Combina las páginas de un diagrama de manera jerárquica.
- **D** Fue desarrollado por E. F. Codd.
- **E** Combina lo mejor de un diagrama de flujo y un diagrama de funciones.

17. En un DFD, los almacenes de datos se representan como _____, y los procesos se indican mediante _____.

18. Entre las ventajas de un DFD están

- **A** Es bueno para trabajar un diseño de lo general a lo particular.
- **B** Su desarrollo es rápido y fácil, incluso para sistemas complejos.
- **C** Expone la estructura general sin sacrificar detalles.
- **D** Presenta fácilmente la lógica compleja.
- **E** Es estupendo para presentaciones ante la administración.

19. Los componentes de una matriz CRUD son

- **A** Elipses para mostrar atributos.
- **B** Los procesos principales se muestran sobre un eje.
- **C** Las entidades principales se presentan en el otro eje.
- **D** Números de referencia para señalar la jerarquía de procesos.
- **E** Letras para indicar las operaciones que efectúan los procesos sobre las entidades.

20. La matriz CRUD ayuda a detectar los siguientes problemas:

- **A** Entidades que nunca son leídas.
- **B** Procesos que nunca son eliminados.
- **C** Procesos que sólo leen.
- **D** Entidades que nunca son actualizadas.
- **E** Procesos que no tienen una entidad de creación.

Capítulo 8

Diseño de la base de datos física

Habilidades y conceptos clave

- Diseño de tablas.
- Integración de las reglas de negocios y la integridad de los datos.
- Diseño de vistas.
- Adición de índices para mejorar el rendimiento.

Tal como se presentó en el capítulo 5 (figura 5-1), una vez que se concluye la fase de diseño lógico de un proyecto, es el momento de pasar al diseño físico. Otros integrantes de un equipo de proyectos normal definirán el hardware y el software del sistema requeridos para el sistema de aplicaciones. Aquí la atención se centrará en la labor del diseñador de una base de datos para generar un diseño físico, que consiste en transformar el diseño lógico en uno o más diseños físicos para la base de datos. En el caso de las situaciones en que se desarrolla un sistema de aplicaciones para uso interno, es normal tener sólo un diseño físico de base de datos por cada diseño lógico. Sin embargo, si la organización vende software, por ejemplo, el sistema de aplicaciones debe funcionar en todas las plataformas diferentes y versiones del RDBMS que emplean los clientes de la organización, y eso requiere varios diseños físicos. En este capítulo se cubre cada uno de los pasos principales relacionados con el diseño físico de una base de datos.

Diseño de tablas

El primer paso en el diseño físico de una base de datos consiste en asignar a tablas las relaciones normalizadas en el diseño lógico. Resulta evidente la importancia de este paso, porque las tablas son la unidad de almacenamiento principal en las bases de datos relacionales. Sin embargo, si el trabajo de diseño lógico ha sido adecuado, se vuelve mucho más fácil la traducción a un diseño físico. Conforme avance por este capítulo, recuerde que el capítulo 2 contiene una introducción para cada componente del modelo físico de una base de datos, y el capítulo 4 incluye la sintaxis de SQL para los comandos del lenguaje de manipulación de datos (DML, Data Manipulation Language) requeridos para crear los diversos componentes físicos de una base de datos (tablas, restricciones, índices, vistas, etc.). En pocas palabras, éste es el proceso:

1. Cada relación normalizada se convierte en una tabla. Ocurre una excepción común cuando se incluyen supertipos y tipos secundarios, tal como se analiza en la sección siguiente.
2. Cada atributo de una relación normalizada se convierte en una columna de la tabla correspondiente. Recuerde que una columna es la división más pequeña de datos significativos en

la base de datos, de modo que las columnas no deben tener componentes secundarios que tengan sentido por sí solos. Para cada columna, debe especificarse lo siguiente:

- *Un nombre de columna único dentro de la tabla*. En general, debe adaptarse con la mayor semejanza posible el nombre de un atributo del diseño lógico. Sin embargo, tal vez se necesiten reajustes para evitar las palabras reservadas y para apegarse a las convenciones de nomenclatura del RDBMS específico que se utiliza. Es posible que, en el ejemplo siguiente, observe ciertas diferencias de nombres entre la relación Cliente y la tabla CLIENTE. La razón de este cambio se analiza en la sección "Convenciones de nomenclatura", más adelante en este capítulo.
- *Un tipo de datos y, para ciertos tipos de datos, una longitud y, tal vez, una precisión*. Los tipos de datos varían de un RDBMS a otro, y por esta razón se requieren diferentes diseños físicos para cada RDBMS que se emplea.
- *Si se requieren o no valores de columnas*. Esto adquiere la forma de una cláusula NULL o NOT NULL para cada columna. Tenga cuidado con los valores predeterminados, pueden confundirlo. Por ejemplo, cuando no se especifica esta cláusula, Oracle supone NULL, pero Sybase ASE y Microsoft SQL Server suponen NOT NULL (aunque este comportamiento predeterminado puede modificarse para una instancia o una base de datos). Siempre es mejor especificar esto y confirmar lo que se obtiene.
- *Restricciones de comprobación*. Éstas se pueden agregar a las columnas para imponer reglas de negocios sencillas. Por ejemplo, se puede implementar una regla de negocios que especifique que el precio unitario en una factura siempre debe ser mayor o igual a cero mediante una restricción de comprobación, pero una regla de negocios que requiera que el precio unitario sea más bajo en ciertos estados no puede emplear una restricción de comprobación. En general, este tipo de restricción está limitada a comparar el valor de una columna con un valor único, un rango o lista de valores, u otros valores de la columna en la misma fila de datos de una tabla.

3. El identificador único de la relación se define como la clave primaria de la tabla. Las columnas que participan en la clave primaria deben especificarse como NOT NULL y, en casi todos los RDBMS, la definición de una restricción de clave principal provoca la definición automática de un índice único en las columnas de clave principal. Las columnas de clave externa deben incluir una cláusula NOT NULL si la relación es obligatoria; de lo contrario, pueden tener una NULL.
4. Cualquier otro grupo de columnas que debe ser único dentro de la tabla puede tener definida una restricción de unicidad. Igual que con las restricciones de clave principal, las restricciones de unicidad en casi todos los RDBMS provocan la definición automática de un índice único en las columnas únicas. Sin embargo, a diferencia de las restricciones de clave principal, una tabla puede tener *varias* restricciones de unicidad, y las columnas en una

restricción de unicidad pueden contener valores nulos (es decir, pueden estar especificadas con la cláusula NULL).

5. La manera en que se entrelazan las relaciones normalizadas se vuelven restricciones referenciales en el diseño físico. En las raras situaciones en que el modelo lógico contiene una relación uno a uno, es posible implementarla al colocar la clave principal de una de las tablas como clave externa en la otra (haga esto sólo para *una* de las dos tablas) *y* al colocar una restricción de unicidad en la clave externa para evitar valores duplicados. Por ejemplo, la figura 2-2 del capítulo 2 presenta una relación uno a uno entre Empleado y Automóvil, y se eligió colocar ID_DE_EMPLEADO como una clave externa en la tabla AUTOMÓVIL. También se debe aplicar una restricción de unicidad a ID_DE_EMPLEADO en la tabla AUTOMÓVIL para que un empleado se pueda asignar sólo a un automóvil en cualquier momento.
6. Las tablas grandes (es decir, las que tienen un tamaño superior a varios gigabytes) deben particionarse, si lo permite el RDBMS que se utiliza. Las *particiones* son una función de las bases de datos que permiten dividir una tabla en varios componentes físicos, cada uno guardado en archivos de datos separados, de manera que sea evidente para el usuario de la base de datos. Los métodos comunes de separación de tablas en particiones emplean un rango o lista de valores para una columna de tabla específica (llamada *columna de partición*) o utilizan un método aleatorio conocido como *hashing* que distribuye equitati-

Pregunta al experto

P: Para una relación uno a uno, ¿por qué se debe colocar una clave externa sólo en una de las dos tablas?

R: El problema de colocar una clave externa en ambos lados de una relación uno a uno es que, en realidad, eso establecería dos relaciones (una para cada clave externa) y la relación redundante fácilmente conduce a una falta de uniformidad de los datos. En el ejemplo de empleado y automóvil, se coloca ID_DE_EMPLEADO en la tabla AUTOMÓVIL y VIN (número de identificación de vehículo) en la tabla EMPLEADO; el DBMS no puede garantizar que los valores de clave externa siempre serán uniformes. Por ejemplo, la fila del empleado 125 puede contener el VIN para un vehículo específico, pero cuando se busca ese vehículo en la fila, tal vez contenga un empleado distinto, por ejemplo 206.

P: Comprendo. Entonces, ¿importa cuál de las dos tablas en la relación uno a uno contiene la definición de la clave externa?

R: Si se supone que se coloca un índice único (o una restricción de unicidad) en la columna de la clave externa, en realidad no hay una diferencia en el desempeño. Sin embargo, tal vez exista una ligera ventaja en poner la clave externa en la tabla que se consulta con mayor frecuencia.

vamente las filas de la tabla entre las particiones disponibles. Algunos beneficios de dividir en particiones las tablas grandes son una administración más fácil (en particular para operaciones de respaldo y recuperación) y un mejor desempeño, que se consigue cuando el RDBMS puede ejecutar una consulta de SQL en paralelo contra todas (o algunas de) las particiones y después combinar los resultados. Una partición es sólo un problema del diseño físico que nunca es abordado en los diseños lógicos. Después de todo, una tabla particionada todavía es *una* tabla. Existe una amplia variación en el modo en que los vendedores de bases de datos han implementado las particiones en sus productos, de manera que necesita consultar los detalles en la documentación de su RDBMS.

7. El modelo lógico puede ser para un sistema de base de datos completo, mientras que el proyecto actual tal vez sea una implementación de un subconjunto de ese sistema completo. Cuando ocurre esto, el diseñador de la base de datos física elegirá e implementará sólo el subconjunto de tablas requeridas para satisfacer las necesidades actuales.

Éste es el diseño lógico para Industrias Acme del capítulo 6:

```
PRODUCTO: Número de producto (CP), Descripción,
          Precio unitario de lista

CLIENTE: Número de cliente (CP), Nombre de cliente,
         Dirección de cliente, Ciudad de cliente, Estado de cliente,
         CP de cliente, Teléfono de cliente

FACTURA: Número de factura (CP), Número de cliente, Términos,
         Método de envío, Fecha de pedido

ELEMENTO DE LÍNEA DE FACTURA: Número de factura (CP), Número de producto (CP),
                              Cantidad, Precio unitario de venta
```

Y aquí está el diseño de la tabla física que se creó a partir del diseño lógico, presentado en la forma de instrucciones del Lenguaje de definición de datos (DDL) de SQL. Estas instrucciones están escritas para Oracle y requieren ciertas modificaciones, sobre todo de los tipos de datos, para funcionar en otros RDBMS. Si quiere ejecutarlas en su propia base de datos de Oracle, es recomendable que cree una nueva cuenta de usuario sólo para este propósito, de modo que las tablas nuevas sean creadas en su propio esquema, en lugar de mezclarse con los otros objetos de la base de datos. En otros RDBMS, como MySQL y SQL Server, debe crear una base de datos nueva y ejecutarlas en esa base de datos. En las bases de datos diferentes a las de Oracle, es probable que tenga que ajustar de algún modo los tipos de datos. Por último, tal vez tenga que ejecutar estas instrucciones de una en una o en lotes pequeños, todo dependiendo del cliente de SQL que utilice.

```
CREATE TABLE PRODUCTO
 (NÚMERO_DE_PRODUCTO          VARCHAR(10)    NOT NULL,
  DESCRIPCIÓN_DE_PRODUCTO     VARCHAR(100)   NOT NULL,
  PRECIO_UNITARIO_DE_LISTA    NUMBER(7,2)    NOT NULL);
```

```
ALTER TABLE PRODUCTO
  ADD CONSTRAINT PRODUCTO_CP_ NÚM_PRODUCTO
      PRIMARY KEY (NÚMERO_DE_PRODUCTO);

CREATE TABLE CLIENTE
 (NÚMERO_DE_CLIENTE   NUMBER(5)      NOT NULL,
  NOMBRE              VARCHAR(25)    NOT NULL,
  DIRECCIÓN           VARCHAR(255)   NOT NULL,
  CIUDAD              VARCHAR(50)    NOT NULL,
  ESTADO              CHAR(2)        NOT NULL,
  CÓDIGO_POSTAL       VARCHAR(10));

ALTER TABLE CLIENTE
  ADD CONSTRAINT CLIENTE_CP_ NÚM_CLIENTE
      PRIMARY KEY (NÚMERO_DE_CLIENTE);

CREATE TABLE FACTURA
 (NÚMERO_DE_FACTURA   NUMBER(7)      NOT NULL,
  NÚMERO_DE_CLIENTE   NUMBER(5)      NOT NULL,
  TÉRMINOS            VARCHAR(20)    NULL,
  MÉTODO_DE_ENVÍO     VARCHAR(30)    NULL,
  FECHA_DE_PEDIDO     DATE           NOT NULL);

ALTER TABLE FACTURA
  ADD CONSTRAINT FACTURA_CP_ NÚM_FACTURA
      PRIMARY KEY (NÚMERO_DE_FACTURA);

ALTER TABLE FACTURA
  ADD CONSTRAINT FACTURA_CE_NÚM_CLIENTE
      FOREIGN KEY (NÚMERO_DE_CLIENTE)
      REFERENCES CLIENTE (NÚMERO_DE_CLIENTE);

CREATE TABLE ELEMENTO_DE_LÍNEA_DE_FACTURA
 (NÚMERO_DE_FACTURA          NUMBER(7)    NOT NULL,
  NÚMERO_DE_PRODUCTO         VARCHAR(10)  NOT NULL,
  CANTIDAD                   NUMBER(5)    NOT NULL,
  PRECIO_UNITARIO_DE_VENTA   NUMBER(7,2)  NOT NULL);

ALTER TABLE ELEMENTO_DE_LÍNEA_DE_FACTURA
  ADD CONSTRAINT LI_FACTURA_CP_NÚM_PROD_FAC
      PRIMARY KEY (NÚMERO_DE_FACTURA, NÚMERO_DE_PRODUCTO);

ALTER TABLE ELEMENTO_DE_LÍNEA_DE_FACTURA
  ADD CONSTRAINT FACTURA_CH_PRECIO_UNIT_VENTA
      CHECK (PRECIO_UNITARIO_DE_VENTA >= 0);

ALTER TABLE ELEMENTO_DE_LÍNEA_DE_FACTURA
  ADD CONSTRAINT LI_FACTURA_CE_NÚM_FACTURA
  FOREIGN KEY (NÚMERO_DE_FACTURA)
  REFERENCES FACTURA (NÚMERO_DE_FACTURA);
```

```
ALTER TABLE ELEMENTO_DE_LÍNEA_DE_FACTURA
  ADD CONSTRAINT LI_FACTURA_CE_NÚMERO_DE_PRODUCTO
      FOREIGN KEY (NÚMERO_DE_PRODUCTO)
      REFERENCES PRODUCTO (NÚMERO_DE_PRODUCTO);
```

Implementación de supertipos y tipos secundarios

Casi todas las personas que modelan datos tienden a especificar todos los tipos secundarios imaginables en el modelo lógico de datos. En realidad esto no es un problema, porque se supone que el diseño lógico abarca no sólo dónde están las cosas en la actualidad, sino también adónde es probable que se dirijan en el futuro. Por lo tanto, el diseñador de la base de datos física debe tomar ciertas decisiones para elegir si implementa o no los supertipos y los tipos secundarios representados en el modelo lógico. Aquí los motivos de peso deben ser la racionalidad y el sentido común. Éstos, junto con las opiniones de los diseñadores de aplicaciones y los usuarios de negocios acerca del uso que pretenden dar a la base de datos, conducirán a las mejores decisiones.

Si revisa la figura 7-8 del capítulo 7, recordará que concluyó con dos tipos secundarios de la entidad Cliente: Cliente individual y Cliente comercial. Usted posee básicamente tres opciones para implementar de manera física este diseño lógico, que se analizarán en las secciones siguientes.

Implementación de los tipos secundarios como están

A ésta se le denomina la solución de las "tres tablas" porque implica crear una tabla para el supertipo, y una tabla para cada uno de los tipos secundarios (dos, en este ejemplo). Este diseño es más adecuado cuando muchos atributos son específicos de los tipos secundarios individuales. En el ejemplo, sólo dos atributos son específicos del tipo secundario Cliente individual (Fecha de nacimiento e Ingreso anual familiar) y cuatro son específicos del tipo secundario Cliente comercial. En la figura 8-1 se muestra el diseño físico de esta opción.

Se prefiere esta opción de diseño cuando se utilizan muchos atributos comunes (ubicados en la tabla del supertipo) al igual que muchos atributos específicos para un tipo secundario u otro (ubicados en las tablas de tipos secundarios). En cierto sentido, este diseño es más sencillo que las otras opciones, porque nadie debe recordar cuáles atributos se aplican a cuál tipo secundario. Por otra parte, también es más complicado utilizarlo, porque el usuario de la base de datos debe combinar la tabla CLIENTE ya sea con la tabla CLIENTE_INDIVIDUAL o con la tabla CLIENTE_COMERCIAL, dependiendo del valor de TIPO_DE_CLIENTE. Es evidente que los puristas del modelado de datos en su equipo de proyecto prefieran este método, pero los programadores de aplicaciones que deben escribir el SQL para consultar las tablas pueden adoptar una posición opuesta.

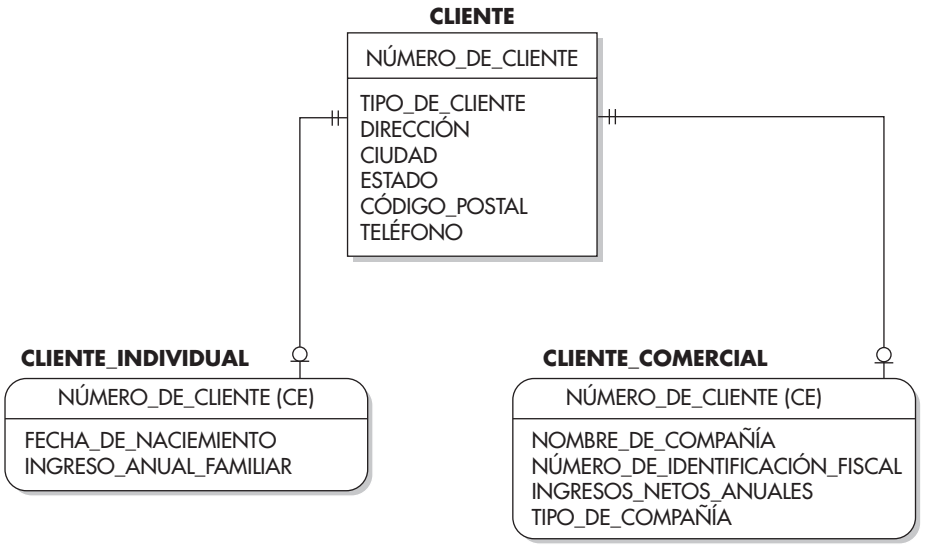


Figura 8-1 Clases secundarias de Cliente: diseño físico de tres tablas.

Implementación de cada tipo secundario como una tabla separada

A ésta se le denomina la solución de las "dos tablas" porque implica crear una tabla para cada tipo secundario e incluir todas las columnas de la tabla supertipo en cada tipo secundario. Al principio, parecería que incluye datos redundantes, pero en realidad no existe un almacenamiento redundante, porque un cliente específico sólo puede ser uno de los dos tipos secundarios. Sin embargo, algunas columnas son definidas de manera redundante. En la figura 8-2 se expone el diseño físico para esta opción.

Se prefiere esta opción cuando muy pocos atributos son comunes entre los tipos secundarios (es decir, cuando la tabla de supertipo contiene muy pocos atributos). En el ejemplo, la si-

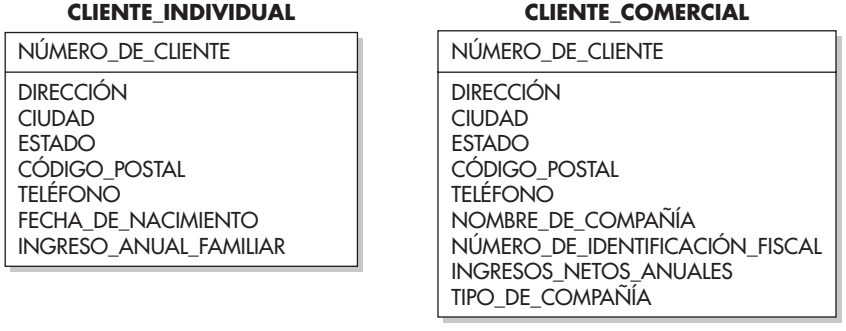


Figura 8-2 Clases secundarias de Cliente: diseño físico de dos tablas.

tuación se complica más debido a la tabla CONTACTO_DE_CLIENTE, que es secundaria de la tabla de supertipo (CLIENTE). No puede (o cuando menos no *debe*) hacer que una tabla sea secundaria de dos tablas primarias diferentes con base en la misma clave externa. Por lo tanto, si elimina la tabla CLIENTE, debe crear dos versiones de la tabla CONTACTO_DE_CLIENTE: una como secundaria de CLIENTE_INDIVIDUAL y otra como secundaria de CLIENTE_COMERCIAL. Aunque puede ser una solución viable en ciertas ocasiones, en este caso el hecho de complicar la tabla CONTACTO_DE_CLIENTE la vuelve una elección deficiente.

Contracción de los tipos secundarios dentro de la tabla de supertipo

Ésta es la solución de "una tabla" porque requiere la creación de una sola tabla que abarque el supertipo y los dos tipos secundarios. En la figura 8-3 se muestra el diseño físico de esta opción. Se requieren restricciones para imponer las columnas opcionales. Cuando las columnas que son obligatorias en los tipos secundarios se consolidan en la tabla supertipo, es común que se definan para permitir valores nulos, porque no se aplican a todos los tipos secundarios. Para el valor TIPO_DE_CLIENTE que significa "Individual", se permitiría (o se requeriría) que FECHA_DE_NACIMIENTO e INGRESO_ANUAL_FAMILIAR contuvieran valores, y se requeriría que NOMBRE_DE_COMPAÑÍA, NÚMERO_DE_IDENTIFICACIÓN_FISCAL, INGRESOS_NETOS_ANUALES y TIPO_DE_COMPAÑÍA fueran nulos. Para el valor TIPO_DE_CLIENTE que significa "Comercial", el comportamiento requerido sería justo lo opuesto.

NOTA

Las restricciones mencionadas aquí pueden implementarse en la base de datos mediante restricciones de comprobación o desencadenadores, que se analizan más adelante en el capítulo, o en la lógica de las aplicaciones. La elección del método que se va a aplicar depende mucho de las opciones del DBMS.

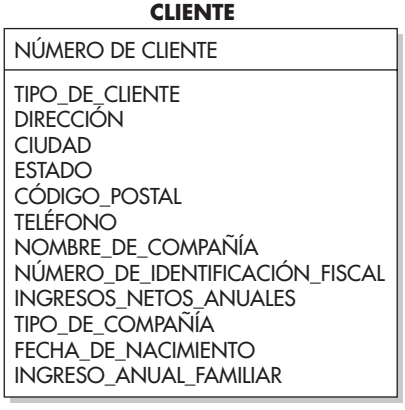


Figura 8-3 Clases secundarias de Cliente: diseño físico de una tabla.

Se prefiere esta opciones cuando relativamente pocos atributos son específicos de cualquier tipo secundario determinado. En cuanto a acceso a los datos, es evidente que resulta la alternativa más sencilla, porque no se requieren combinaciones. Sin embargo, tal vez sea más complicada en cuanto a la lógica, porque siempre debe tener en cuenta cuáles atributos se aplican a cada tipo secundario (es decir, cuál valor de TIPO_DE_CLIENTE en este ejemplo). Con sólo dos tipos secundarios, y un total de seis atributos determinados por ambos tipos secundarios, ésta parece una opción muy atractiva para este ejemplo.

Convenciones de nomenclatura

Las convenciones de nomenclatura son importantes porque ayudan a promover la uniformidad en los nombres de tablas, columnas, restricciones, índices y otros objetos de una base de datos. Cada organización debe desarrollar un grupo estándar de convenciones de nomenclatura (con las variaciones necesarias cuando se utilizan varios RDBMS), publicarlo e imponer su uso. Las convenciones que se ofrecen aquí son sugerencias basadas en las mejores prácticas actuales en la industria.

Convenciones de nomenclatura para tablas

Éstas son algunas convenciones de nomenclatura sugeridas para las tablas de una base de datos:

- Los nombres de tabla deben basarse en el nombre de la entidad que representan. Deben ser descriptivos y, al mismo tiempo, concisos.
- Los nombres de tabla deben ser únicos en toda la organización (es decir, en todas las bases de datos), excepto donde la tabla es un duplicado exacto de otra (es decir, una copia duplicada).
- Algunos diseñadores prefieren palabras en singular para los nombres de tablas, mientras que otros prefieren nombres en plural (por ejemplo, CLIENTE en comparación con CLIENTES). Oracle Corporation recomienda nombres en singular para las entidades y en plural para las tablas (una convención que nunca he comprendido). No importa cuál convención adopte, siempre y cuando la aplique de manera *uniforme* en *todas* sus tablas, de modo que establezca una u otra como su estándar.
- No incluya palabras como “tabla” o “archivo” en los nombres de tablas.
- Emplee sólo letras mayúsculas, y separe las palabras con un guión bajo. No todos los RDBMS tienen nombres de objetos sensibles a mayúsculas, de modo que los nombres con mayúsculas y minúsculas limitan la posibilidad de aplicación entre diferentes vendedores. Muchos productos de RDBMS, como Oracle y DB2, permiten nombres con mayúsculas y minúsculas en SQL pero convierten todos a mayúsculas cuando son procesados. Los nombres en los metadatos de catálogo se guardan en mayúsculas, y cuando los consulta después con una herramienta popular para DBA o desarrollador, se vuelven difíciles de

descifrar. Por ejemplo, una tabla creada con el nombre HistDePuestEmpl se mostraría como HISTDEPUESTEMPL.

- Cuando sea necesario, utilice abreviaturas para acortar los nombres que sean más largos que el máximo del RDBMS (por lo general, alrededor de 30 caracteres). En realidad, es una buena idea mantener menos caracteres que el máximo del RDBMS para permitir la inclusión de sufijos, cuando se requiera. Todas las abreviaturas deben incluirse en una lista estándar y no se recomienda el uso de abreviaturas no estándar.
- Evite nombres limitantes como VENTAS_OESTE. Algunas organizaciones incorporan a los nombres de tablas un prefijo de dos o tres caracteres para indicar la sección de la organización que posee los datos en la tabla. Sin embargo, ésta no se considera una buena práctica porque puede provocar que no se compartan los datos. Además, colocar nombres de unidades organizacionales o geográficas en los nombres de tabla puede provocar un caos cada vez que cambia la organización.

Convenciones de nomenclatura para columnas

Éstas son algunas convenciones de nomenclatura sugeridas para las columnas de una tabla:

- Los nombres de columnas deben basarse en el nombre del atributo mostrado en el modelo lógico de datos. Deben ser descriptivos y, al mismo tiempo, concisos.
- Los nombres de columnas deben ser únicos dentro de una tabla, pero en donde sea posible, es mejor si son únicos en toda la organización. Algunas convenciones permiten excepciones para atributos comunes como Ciudad, que pueden describir varias entidades como Cliente, Empleado y Ubicación de compañía.
- Utilice sólo mayúsculas, y separe las palabras con guiones bajos. No todos los RDBMS tienen nombres de objetos sensibles a mayúsculas, de modo que los nombres con mayúsculas y minúsculas limitan la posibilidad de aplicación entre varios vendedores.
- La inclusión de un sufijo de entidad en los nombres de columnas causa controversia. Algunos prefieren nombres con sufijos. Por ejemplo, en una tabla CLIENTE, emplearían nombres de columnas como NUMERO_DE_CLIENTE, NOMBRE_DE_CLIENTE, DIRECCION_DE_CLIENTE, CIUDAD_DE_CLIENTE y así por el estilo. Otros (entre quienes me incluyo) prefieren incluir sufijos sólo en el nombre de la columna de clave principal (por ejemplo, NUMERO_DE_CLIENTE), lo que conduce fácilmente a que las columnas de clave principal y de clave externa correspondientes tengan exactamente los mismos nombres. Unos más prefieren descartar cualquier sufijo, y terminar con un nombre de columnas como Id para la clave principal de cada tabla individual.
- Utilice abreviaturas, cuando sea necesario, para acortar los nombres que sean más largos que el máximo del RDBMS (por lo general, alrededor de 30 caracteres). Todas las abreviaturas deben incluirse en una lista estándar y no se recomienda el uso de abreviaturas no estándar.

- Sin tomar en cuenta cualquier otra convención, casi todos los expertos prefieren que las columnas de clave externa siempre tengan exactamente el mismo nombre que su columna de clave principal correspondiente. Esto ayuda a otros usuarios de la base de datos a comprender cuáles columnas se utiilizarán cuando codifican combinaciones en SQL.

Convenciones de nomenclatura para restricciones

En casi todos los RDBMS, el mensaje de error generado cuando se viola una restricción contiene el nombre de ésta. A menos que quiera atrapar las preguntas de los usuarios de una base de datos cada vez que aparezca uno de estos mensajes, debe asignar a las restricciones un nombre estándar que los usuarios de la base de datos comprendan con facilidad. Casi todos los diseñadores de bases de datos prefieren una convención similar a la que se presenta aquí.

Los nombres de restricciones deben estar en el formato *NOMTABLA_TIPO_NOMCOLUMNA*, en que:

- *NOMTABLA* es el nombre de la tabla en que se define la restricción, abreviado, si es necesario.
- *TIPO* es el tipo de restricción.
  - CP para restricciones de clave principal.
  - CE para restricciones de clave externa.
  - UN para restricciones de unicidad.
  - CK para restricciones de comprobación.
- *NOMCOLUMNA* es el nombre de la columna en que se define la restricción, abreviado, si es necesario. En caso de restricciones definidas a través de varias columnas, se pueden reemplazar con otra palabra o frase descriptiva si los nombres de columnas son demasiado extensos (incluso cuando se abrevian), para que tengan sentido.

Convenciones de nomenclatura para índices

Los índices definidos automáticamente por el RDBMS para permitir restricciones de clave principal o de unicidad suelen recibir el mismo nombre que la restricción, de modo que rara vez debe preocuparse por ellos. En el caso de otros tipos de índices, es inteligente emplear una convención de nomenclatura para que identifique la tabla y las columnas en que están definidos sin tener que buscar nada más. Se sugiere la convención siguiente.

Los nombres de restricciones deben estar en el formato *NOMTABLA_TIPO_NOMCOLUMNA*, en el que:

- *NOMTABLA* es el nombre de la tabla en que se define el índice, abreviado, si es necesario.
- *TIPO* es el tipo de índice.
  - UX para índices únicos.
  - IX para índices que no son únicos.

- *NOMCOLUMNA* es el nombre de la columna en que se define el índice, abreviado, si es necesario. En el caso de los índices definidos a través de varias columnas, se pueden reemplazar con otra palabra o frase descriptiva, si los nombres de columnas son demasiado extensos (aunque se abrevien), para que tengan sentido.

  Cualquier abreviatura utilizada debe documentarse en una lista estándar.

Convenciones de nomenclatura para vistas

Los nombres de vistas presentan un dilema interesante. Los nombres de objetos utilizados en la cláusula FROM de las instrucciones de SQL pueden ser para tablas, vistas o sinónimos. Un *sinónimo* es un alias (sobrenombre) de una tabla o vista. Entonces, ¿cómo sabe el DBMS si un nombre de objeto en la cláusula FROM es una tabla, una vista o un sinónimo? No lo sabe hasta que busca el nombre en una tabla de metadatos que cataloga todos los objetos de la base de datos. Claro que esto significa que los nombres de tablas, vistas y sinónimos deben provenir del mismo *espacio de nombres* o lista de nombres. Por lo tanto, un nombre de vista debe ser único entre todos los nombres de tablas, vistas y sinónimos.

Es útil que por lo menos algunos usuarios de la base de datos sepan si hacen referencia a una tabla o una vista: por ello, una práctica común consiste en aplicar nombres distintivos a las vistas mediante un estándar que agrega *VW* al inicio o al final de cada nombre, lo que además representa un modo fácil de asegurar que los nombres sean únicos. Una vez más, la convención exacta elegida importa mucho menos que seleccionar *una* convención estándar y apegarse a ella para todos sus nombres de vistas. Ésta es una convención sugerida:

- Todos los nombres de vistas deben terminar con *_VW* para que se diferencien con facilidad de los nombres de tablas.
- Los nombres de vistas deben contener el nombre de la tabla base más importante incluida en la vista, abreviado si es necesario.
- Los nombres de vistas deben describir el propósito de las vistas o el tipo de datos incluidos en ellas. Por ejemplo, CLIENTES_CALIFORNIA_VW y CLIENTES_POR_CODIGO_POSTAL_VW son nombres de vistas razonablemente descriptivos, mientras que LISTA_DE_CLIENTES_VW y COMBINACION_DE_CLIENTES_VW son mucho menos significativos.
- Las abreviaturas utilizadas deben documentarse en la lista de abreviaturas estándar.

Integración de las reglas de negocios y la integridad de los datos

Las reglas de negocios determinan cómo funciona una organización y cómo utiliza sus datos. Las reglas de negocios existen como un reflejo de las políticas y procedimientos operativos de

una organización y porque aportan control. La *integridad de los datos* es el proceso de asegurar que los datos estén protegidos y se mantengan intactos a través de las restricciones definidas que se aplican a los datos. A éstas se les llama *restricciones de la base de datos* porque evitan cambios en los datos que violarían una o más reglas de negocios. El beneficio principal de imponer reglas de negocios que utilicen restricciones de integridad de datos en la base de datos es que las restricciones no pueden evitarse. A diferencia de las reglas de negocios impuestas por los programas de aplicaciones, las restricciones de una base de datos son impuestas sin importar *cómo* se conecta alguien. El único modo de evitar las restricciones de una base de datos es que el DBA las elimine o las deshabilite. Por otra parte, los desarrolladores tienen preferencia por ser ellos mismos quienes controlan la imposición de una regla, en lugar de relegarlas a un DBA, y algunas reglas se prueban mejor antes de enviar los datos a la base de datos para procesamiento. En raros casos, por lo general relacionados con las reglas de negocios más importantes, es posible que quiera imponerlas en ambos lugares: en la base de datos porque la regla no puede ser evitada, y en la aplicación, para que el usuario se entere de inmediato cuando viola la regla.

Las reglas de negocios se implementan en la base de datos del modo siguiente:

- Restricciones NOT NULL.
- Restricciones de clave principal.
- Restricciones referenciales (de clave externa).
- Restricciones de unicidad.
- Restricciones de comprobación.
- Tipos de datos, precisión y escala.
- Desencadenadores.

En las secciones secundarias siguientes se analiza cada una de estas técnicas de implementación y los efectos de las restricciones sobre el procesamiento de una base de datos. Durante todo este análisis, se utiliza como ejemplo la definición de tabla que aparece a continuación (en Oracle SQL). Un *comentario* (una instrucción que comienza con dos guiones) aparece antes de cada componente, como ayuda para identificarlo. Observe que la tabla FACTURA utilizada aquí tiene una diferencia de columna: TÉRMINOS es sustituida con NÚMERO_OC_DE_CLIENTE, lo que es necesario para ilustrar algunos conceptos importantes. Se incluye una instrucción DROP para descartar la tabla FACTURA en caso de que la haya creado al seguir los ejemplos anteriores.

```
-- Descartar tabla FACTURA (en caso de que ya exista una)
DROP TABLE FACTURE CASCADE CONSTRAINTS;
-- Crear tabla FACTURA
CREATE TABLE FACTURA
 (NÚMERO_DE_FACTURA      NUMBER(7)     NOT NULL
  NÚMERO_DE_CLIENTE      NUMBRE(5)     NOT NULL
  NÚMERO_OC_DE_CLIENTE   VARCHAR(10)   NULL,
```

```
  MÉTODO_DE_ENVÍO         VARCHAR(30)    NULL,
  FECHA_DE_PEDIDO         DATE           NOT NULL);

-- Crear restricción de clave principal
ALTER TABLE FACTURA
  ADD CONSTRAINT FACTURA_CP_NÚMERO_DE_FACTURA
      PRIMARY KEY (NÚMERO_DE_FACTURA);

-- Crear restricción referencial
ALTER TABLE FACTURA
  ADD CONSTRAINT FACTURA_CE_NÚMERO_DE_FACTURA
      FOREIGN KEY (NÚMERO_DE_CLIENTE)
      REFERENCES CLIENTE (NÚMERO_DE_CLIENTE);

-- Crear restricción de unicidad
ALTER TABLE FACTURA
  ADD CONSTRAINT FACTURA_UNC_OC_NÚM_FACTURA
      UNIQUE (NÚMERO_DE_CLIENTE, NÚMERO_OC_DE_CLIENTE);

-- Crear restricción de comprobación (CHECK)
ALTER TABLE FACTURA
  ADD CONSTRAINT FACTURA_CK_NÚM_FACTURA
      CHECK (NÚMERO_DE_FACTURA >0);
```

Restricciones NOT NULL

Como ya ha visto, las reglas de negocios que declaran cuáles atributos se requieren se traducen como cláusulas NOT NULL en las columnas correspondientes en el diseño de tablas. De hecho, la cláusula NOT NULL es el modo en que se define una restricción NOT NULL en las columnas de una tabla. Las claves principales siempre deben especificarse como NOT NULL (Oracle lo hace automáticamente, a diferencia de casi todos los demás productos de RDBMS). Y, como ya se mencionó, las claves externas que participan en una relación obligatoria también deben especificarse como NOT NULL.

En nuestro ejemplo, si se intenta insertar una fila en la tabla FACTURA y no se proporciona un valor para cualquiera de las columnas que tienen restricciones NOT NULL (es decir, las columnas NÚMERO_DE_FACTURA, NÚMERO_DE_CLIENTE y FECHA_DE_PEDIDO), la inserción fracasará y mostrará un mensaje de error que indica la violación de la restricción. Asimismo, si se intenta actualizar cualquier fila existente y establecer una de esas columnas con un valor NULL, la instrucción de actualización fracasará.

Restricciones de clave principal

Las restricciones de clave principal requieren que las columnas que forman ésta contengan valores únicos para cada fila de la tabla. Además, las columnas de clave principal deben definirse con restricciones NOT NULL. Una tabla sólo puede tener una restricción de clave principal. Casi todos los RDBMS crean automáticamente un índice para ayudar a imponer la restricción de clave principal.

En la tabla FACTURA de ejemplo, si se intenta insertar una fila sin especificar un valor para la columna NÚMERO_DE_FACTURA, la inserción fracasará debido a la restricción NOT NULL sobre la columna. Si en lugar de eso se intenta insertar una fila con un valor para la columna NÚMERO_DE_FACTURA que ya existe en la tabla FACTURA, la inserción fracasará y mostrará un mensaje de error que indica una violación de restricción de clave principal. Este mensaje suele contener el nombre de la restricción; por eso es una buena idea asignar nombres significativos a las restricciones. Por último, si se supone que el RDBMS que se utiliza permite actualizaciones a los valores de clave principal (algunos no lo permiten), si se intenta actualizar la columna NÚMERO_DE_FACTURA en una fila existente y se proporciona un valor que ya es utilizado por otra fila en la tabla, la actualización fracasará.

Restricciones referenciales (de clave externa)

La restricción referencial sobre la tabla FACTURA define NÚMERO_DE_CLIENTE como clave externa de la tabla CLIENTE. Se requiere tiempo para acostumbrarse, pero las restricciones referenciales siempre se definen en la tabla secundaria (es decir, la tabla en el lado "varios" de la relación). El propósito de la restricción referencial es comprobar que los valores de clave externa en las filas de la tabla secundaria *siempre* tengan valores de clave principal que coincidan en la tabla primaria.

En el ejemplo de la tabla FACTURA, si se intenta insertar una fila sin proporcionar un valor para NÚMERO_DE_CLIENTE, la inserción fracasará debido a la restricción NOT NULL sobre la columna. Sin embargo, si se intenta insertar una fila y se aporta un valor para NÚMERO_DE_CLIENTE que no coincide con la clave principal de una fila en la tabla CLIENTE, la inserción fracasará debido a la restricción referencial. Asimismo, si se intenta actualizar el valor de NÚMERO_DE_CLIENTE para una fila existente en la tabla FACTURA y el nuevo valor no tiene una fila que coincida en la tabla CLIENTE, la actualización fracasará, otra vez debido a la restricción referencial.

Recuerde siempre que las restricciones referenciales funcionan en ambas direcciones, de modo que pueden evitar que una fila de una tabla secundaria se convierta en "huérfana", lo que significa que posee un valor que no coincide con un valor de clave principal en la tabla primaria. Por lo tanto, si se intenta eliminar una fila de la tabla CLIENTE que tiene filas de FACTURA que hacen referencia a ella, la instrucción fracasará debido a que puede provocar que algunas filas de la tabla secundaria violen la restricción. Sucede lo mismo si se intenta actualizar el valor de clave principal de esa fila. Sin embargo, muchos RDBMS proporcionan una función con las restricciones referenciales escritas como **ON DELETE CASCADE**, lo que provoca que las referencias a las filas de una tabla secundaria sean eliminadas *automáticamente* cuando se descarta la fila primaria. Por supuesto, esta opción no es adecuada en todas las situaciones, pero es conveniente tenerla en caso de que se necesite.

Pregunta al experto

P: **Mencionó que la instrucción ON DELETE CASCADE no es adecuada en todas las situaciones. ¿Cuándo sería apropiada?**

R: **ON DELETE CASCADE** es conveniente cuando las filas de la tabla secundaria no pueden existir sin las filas de la tabla primaria, situación conocida como *dependencia de existencia*. Por ejemplo, un artículo de línea en una factura no puede existir sin la factura, de modo que es lógico eliminar los artículos de línea automáticamente cuando una instrucción de SQL intenta eliminar la factura. Sin embargo, esta opción puede ser peligrosa en otras situaciones. Por ejemplo, sería peligroso configurar la base de datos para que las facturas sean eliminadas automáticamente cuando alguien intenta eliminar a un cliente; como las facturas son registros financieros, sería más seguro obligar a las personas a que primero eliminen las facturas de manera explícita. Por supuesto, éstas son decisiones de una regla de negocios que dependen de los requisitos.

Restricciones de unicidad

Al igual que las restricciones de clave principal, las de unicidad aseguran que dos filas de una tabla no tengan valores duplicados para las columnas mencionadas en la restricción. Sin embargo, las restricciones de unicidad tienen dos diferencias importantes:

- Aunque una tabla sólo puede tener una restricción de clave principal, puede tener todas las restricciones de unicidad necesarias.
- Las columnas que participan en una restricción de unicidad no necesitan tener restricciones NOT NULL.

Tal como ocurre con una restricción de clave principal, se crea automáticamente un índice para ayudar al DBMS en la imposición eficiente de la restricción.

En el ejemplo, se define una restricción de unicidad en las columnas NÚMERO_DE_CLIENTE y NÚMERO_OC_DE_CLIENTE, para imponer una regla de negocios que declara que los clientes pueden crear sólo una vez un número de orden de compra (OC). Debe comprender que la *combinación* de los valores en las dos columnas debe ser única. Pueden existir varias facturas para cualquier NÚMERO_DE_CLIENTE determinado, y varias filas en la tabla FACTURA pueden tener el mismo NÚMERO_DE_OC (no es posible evitar que los clientes utilicen el mismo número de OC, ni se pretende hacerlo). Sin embargo, dos filas para el mismo número de cliente no pueden tener el mismo número de OC.

Igual que con una restricción de clave principal, si se intenta insertar una fila con valores para las columnas NÚMERO_DE_CLIENTE y NÚMERO_OC_DE_CLIENTE que ya son utilizados por otra fila, la inserción fallará. Asimismo, no es posible actualizar una fila en la tabla FACTURA si la actualización hace que se tenga una combinación duplicada de NÚMERO_DE_CLIENTE y NÚMERO_DE_OC.

Restricciones de comprobación

Las restricciones de comprobación (CHECK) se emplean para imponer las reglas de negocios que limitan una columna a una lista, un rango de valores o alguna condición que puede verificarse mediante una comparación única con una constante, un cálculo o un valor de otra columna en la misma fila. Las restricciones de comprobación *no* pueden usarse para comparar valores de columnas entre filas diferentes, ya sea en la misma tabla o no. Las restricciones de comprobación se escriben como funciones condicionales que siempre deben ser verdaderas. La terminología proviene del hecho de que la base de datos siempre debe "comprobar" la condición para asegurar que se evalúa como verdadera antes de permitir una inserción o una actualización a una fila de la tabla.

En el ejemplo, una restricción de comprobación requiere que NÚMERO_DE_FACTURA sea mayor que 0. Esto impone una regla de negocios que requiere números de factura positivos. Recuerde que la condición sólo se comprueba cuando se inserta o actualiza una fila en la tabla FACTURA, de modo que no se aplicará a las filas existentes en la tabla (en caso de que exista alguna) cuando se agrega la restricción. Cuando la restricción está impuesta, si se intenta insertar o actualizar una fila con un NÚMERO_DE_FACTURA establecido en cero o un número negativo, la instrucción fallará.

Tipos de datos, precisión y escala

El tipo de datos asignado a las columnas de una tabla limita automáticamente los datos a valores que coincidan con el tipo de datos. Por ejemplo, cualquier cosa colocada en una columna con un formato de fecha debe ser una fecha válida. No es posible incluir caracteres no numéricos en columnas numéricas. Sin embargo, es posible poner casi cualquier cosa en una columna de caracteres.

Para los tipos de datos que permiten la especificación de la precisión (el tamaño máximo) y la escala (las posiciones a la derecha del punto decimal), estas especificaciones también limitan los datos. No es posible incorporar una cadena de caracteres o un número más grande que el tamaño máximo de la columna dentro de la base de datos. Tampoco se pueden especificar posiciones decimales más allá de las permitidas para la escala de un número.

En el ejemplo, NÚMERO_DE_CLIENTE sólo debe contener dígitos numéricos y no puede ser mayor de 99 999 (cinco dígitos) ni menor de -99 999 (otra vez, cinco dígitos). Asimismo, debido a que la escala es 0, no puede tener dígitos decimales (es decir, debe ser un entero). Puede parecer torpe permitir valores negativos para NÚMERO_DE_CLIENTE, pero ningún tipo de datos de SQL limita una columna a enteros positivos. Sin embargo, es fácil limitar una columna a números positivos mediante una restricción de comprobación, si se requiere esa restricción.

Desencadenadores

Como recordará, un *desencadenador* es una unidad de código de programa que se ejecuta en forma automática con base en cierto evento que ocurre en la base de datos, como la inserción, actualización o eliminación de datos en una tabla específica. Los desencadenadores deben escribirse en un lenguaje permitido por el RDBMS. Para Oracle, ésta es una extensión patentada para SQL llamada PL/SQL (lenguaje procedimental/SQL, Procedural Language/SQL) o Java

(disponible en Oracle8*i* o posterior). Para Sybase ASE y Microsoft SQL Server, el lenguaje permitido es Transact-SQL. Algunos RDBMS no tienen soporte a desencadenadores, mientras que otros permiten un lenguaje de programación más general, como C. El código de un desencadenador debe terminar de manera normal, lo que permite que concluya de manera normal la instrucción de SQL que hizo que se ejecutara el desencadenador, o debe generar un error de la base de datos, lo que a su vez hace que también falle la instrucción de SQL que hizo que se iniciara el desencadenador.

Los desencadenadores imponen las reglas de negocios que no se pueden imponer a través de restricciones de la base de datos. Debido a que se escriben mediante un lenguaje de programación con características completas, pueden hacer todo lo que está permitido con una base de datos y un programa (algunos RDBMS aplican algunas restricciones a los desencadenadores). No siempre es fácil decidir si una regla de negocios debe imponerse en el código de aplicaciones normal o mediante el uso de un desencadenador. Quienes desarrollan aplicaciones suelen tener el deseo de controlar esas cosas, pero por otra parte, el beneficio principal de los desencadenadores es que se ejecutan automáticamente y no pueden evitarse (a menos que el DBA los elimine o deshabilite), aunque alguien se conecte directamente a la base de datos y pase por alto la aplicación.

Un uso común de los desencadenadores, en los RDBMS que no permiten una instrucción **ON DELETE CASCADE** en las restricciones referenciales, consiste en efectuar una eliminación en cascada. Por ejemplo, si se prefiere que los artículos de línea se eliminen automáticamente de la tabla ELEMENTO_DE_LÍNEA_DE_FACTURA cuando se elimine la factura correspondiente en la tabla FACTURA, es posible escribir un desencadenador que realice eso. El desencadenador se establece para ejecutarse cuando ocurra una eliminación en la tabla FACTURA. A continuación emite una eliminación para todas las filas secundarias relacionadas con la factura primaria (las que tienen el mismo valor de clave principal de la factura que se elimina) y después termina normalmente, lo que permite que concluya la eliminación de la factura original (para ese momento ya habrán desaparecido las filas secundarias a las que se hace referencia, por lo que la eliminación no viola la restricción referencial).

Diseño de vistas

Tal como se explicó en el capítulo 2, las vistas se consideran tablas virtuales. Sin embargo, son tan sólo instrucciones guardadas de SQL que no contienen ningún dato. Éstos pueden seleccionarse de las vistas igual que se eligen de las tablas y, con ciertas restricciones, es posible insertar, actualizar y eliminar datos de las vistas. Éstas son las restricciones:

- Para las vistas que contienen combinaciones, cualquier instrucción de DML (es decir, inserción, actualización o eliminación) emitida contra la vista debe hacer referencia a una sola tabla.
- No se permiten inserciones en las vistas cuando ha sido omitida cualquier columna requerida (NOT NULL).
- Cualquier actualización en contra de una vista puede hacer referencia sólo a columnas que se relacionan directamente con columnas de la tabla base. Tampoco pueden actualizarse columnas calculadas y derivadas.

- Se requieren privilegios adecuados (igual que con las tablas base).
- Cuando se utilizan vistas, se aplican diversas restricciones específicas de un producto, de modo que siempre debe consultarse la documentación del RDBMS.

Las vistas se diseñan para proporcionar las ventajas siguientes:

- En algunos RDBMS, las vistas aportan una ventaja de desempeño sobre las instrucciones comunes de SQL. Las vistas se compilan con anticipación, de modo que se conservan los recursos requeridos para verificar la sintaxis de la instrucción y prepararla para procesamiento cuando se hace referencia frecuente a ellas. Sin embargo, no existe esa ventaja con los RDBMS que ofrecen un caché automático para instrucciones de SQL, tal como lo hace Oracle. Además, es posible incluir en una vista SQL mal escrito, de modo que incorporar SQL a una vista no es una solución mágica para los problemas de desempeño.
- Asimismo, en algunos RDBMS, los procedimientos guardados funcionan mejor que las vistas. (Un *procedimiento guardado* es un programa escrito en un lenguaje permitido por el RDBMS y guardado en la base de datos. Se invoca con una instrucción SQL y devuelve un conjunto de resultados, tal como lo hace una vista.) Los procedimientos guardados pueden efectuar mucha más manipulación de datos que la conseguida con una vista.
- Las vistas pueden adaptarse a las necesidades de un departamento individual, por lo que sólo proporcionan las filas y las columnas necesarias, y también es posible cambiar el nombre de las columnas con términos que comprendan con mayor facilidad los usuarios específicos.
- Debido a que las vistas ocultan a los usuarios los nombres reales de tablas y columnas, impiden que los usuarios cambien esos nombres en las tablas bases.
- El uso de datos se simplifica mucho al ocultar las combinaciones y los cálculos complicados a los usuarios de la base de datos. Por ejemplo, las vistas calculan con facilidad las edades con base en las fechas de nacimiento, y resumen los datos de casi cualquier manera imaginable.
- Se respetan las necesidades de seguridad al filtrar las filas y las columnas que no deben ver los usuarios. Algunos productos de RDBMS contienen seguridad en el nivel de columna, y se conceden privilegios por columna y por tabla a los usuarios, pero la implementación y el mantenimiento de las vistas es mucho más fácil. Además, una cláusula WHERE en la vista filtra las filas con facilidad.

Una vez creadas, las vistas deben ser administradas igual que cualquier otro objeto de una base de datos. Si muchos integrantes de un proyecto crean y actualizan vistas, es muy fácil perder el control. Además, las vistas pueden invalidarse cuando se realiza el mantenimiento de la base de datos, de modo que su situación debe revisarse de manera periódica.

Adición de índices para mejorar el rendimiento

Los índices proporcionan un medio rápido y eficiente para encontrar filas de datos en las tablas, de manera muy similar a como un índice al final de un libro ayuda a encontrar referencias específicas con rapidez. Aunque su implementación en una base de datos es más complicada que esto, resulta más fácil visualizar un índice como una tabla con una columna que

contiene el valor de la clave y otra que contiene un apuntador al lugar de la tabla en que reside físicamente la fila con el valor de la clave, en forma de una Id de fila o una dirección del bloque relativa (RBA, Relative Block Address). En el caso de los índices que no son únicos, la segunda columna contiene una lista de los apuntadores correspondientes.

Los índices proporcionan búsquedas más rápidas que la exploración de tablas, por dos razones: en primer lugar, las entradas de un índice son mucho más breves que las filas de tabla comunes, de modo que caben muchas entradas de índice más por bloque de archivo físico que filas correspondientes en una tabla. Por lo tanto, cuando la base de datos debe examinar el índice de manera secuencial en busca de filas que coincidan, obtiene muchas entradas de índice más con una sola lectura del archivo en el disco que una lectura correspondiente del archivo que contiene la tabla. En segundo lugar, las entradas de índice siempre se conservan en la secuencia de las claves, lo que no siempre es cierto con las tablas. El software de RDBMS puede aprovechar esto mediante técnicas de búsqueda binaria que reducen notablemente los tiempos de búsqueda y los recursos requeridos para la misma.

Sin embargo, todo tiene un precio y éste es el de los índices: ocupan espacio y deben recibir mantenimiento. El espacio de almacenamiento deja de ser un problema con cada día que transcurre, porque los dispositivos de almacenamiento cada día se vuelven más económicos. No obstante, todavía cuestan algo, requieren mantenimiento y deben ser respaldados. Casi todos los vendedores de RDBMS proporcionan herramientas para calcular el espacio de almacenamiento requerido para los índices, que le ayudarán a calcular los requisitos de almacenamiento. La consideración más importante es el mantenimiento del índice. Dondequiera que se inserte una fila en una tabla, también debe insertarse una entrada nueva en cada índice definido en esa tabla. Cuando se eliminan filas, también deben retirarse las entradas en el índice. Y cuando se actualizan las columnas que tienen un índice definido sobre ellas, el índice también debe actualizarse. Es fácil olvidar esto porque el RDBMS hace este trabajo de manera automática, pero cada índice tiene un efecto de desgaste en el desempeño de las inserciones, actualizaciones y eliminaciones en los datos de una tabla. En esencia, se suele requerir una solución intermedia normal, al sacrificar un poco del desempeño de las instrucciones del DML por ganancias considerables en el desempeño de una instrucción SELECT.

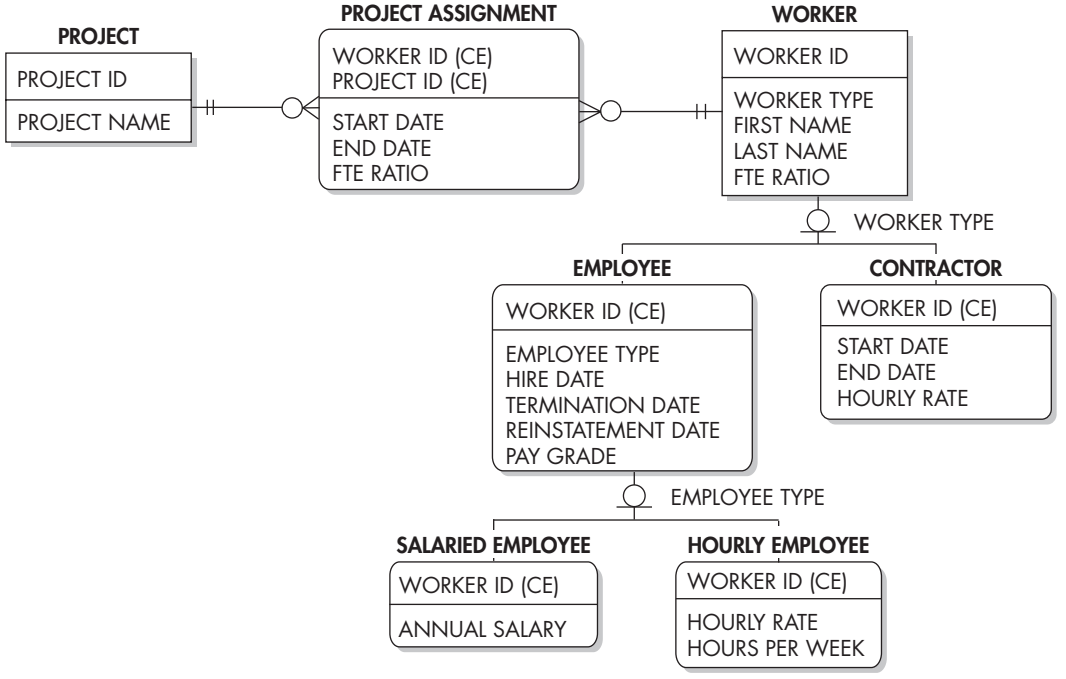
Éstos son algunos lineamientos generales en relación con el uso de índices:

- Recuerde que casi todos los RDBMS crean automáticamente índices en las columnas de claves en las restricciones de clave principal y de unicidad.
- Los índices en claves externas mejoran de manera notable el desempeño de las combinaciones.
- Considere el uso de índices en columnas a las que se hace referencia frecuente en las cláusulas WHERE.
- Cuanto más grande sea una tabla, menos recomendable será que cualquier consulta a la base de datos tenga que explorar una tabla completa (en otras palabras, será más probable que quiera que *cada* consulta utilice un índice).
- Cuantas más veces se actualice una tabla, menor será la cantidad de índices que habrá de contener, en particular en las columnas que se actualizan más a menudo.

- En el caso de las tablas relativamente pequeñas (menos de 1000 filas), es probable que las exploraciones secuenciales de la tabla sean más eficientes que los índices. Casi todos los RDBMS tienen optimizadores que deciden cuándo debe emplearse un índice, y por lo general preferirán una exploración de tabla a un índice hasta que existan cuando menos varios cientos de filas en la tabla.
- En el caso de las tablas con filas relativamente cortas que se consultan con más frecuencia mediante la clave principal, considere el uso de una *tabla organizada de índice* (en los RDBMS que la permiten), en donde se conservan todos los datos de la tabla en el índice. Ésta puede ser una estructura muy eficiente para las tablas de búsqueda (las que contienen poco más que las columnas de código y de descripción).
- Pondere con atención las consecuencias en el desempeño antes de definir más de dos o tres índices en una sola tabla.

Pruebe esto 8-1 Ubicación de un modelo lógico en el diseño físico de una base de datos

La implementación de tipos secundarios y supertipos en las bases de datos relacionales es tal vez la parte más desafiante del diseño físico de una base de datos. Este ejercicio le ofrece la oportunidad para practicar esta habilidad esencial. La ilustración 8-1 exhibe una parte del modelo lógico de una sección de la aplicación HR (recursos humanos). Los pasos de este ejercicio lo guían a la conversión de este modelo en un modelo físico de datos.



Paso a paso

1. Dado que las entidades Salaried Employee (empleado asalariado) y Hourly Employee (empleado por horas) tienen tan pocos atributos, parece mejor contraerlas en la correlación Employee. Traslade los atributos Annual Salary (salario anual) y Hourly Rate (tarifa por horas) a Employee.
2. En un análisis adicional, observa que las entidades Employee y Contractor (contratista) tienen un atributo Hourly Rate. Por lo tanto, necesita mover Hourly Rate a la entidad Worker (trabajador).
3. Después de un análisis con los analistas de negocios que trabajan en el proyecto, concluye que Hours Per Week (horas por semana) se deriva fácilmente de FTE Ratio (cociente de equivalencia de tiempo completo) de la entidad Worker. Por ejemplo, un FTE de 0.5 significa que la persona trabaja 20 horas por semana (40 * 0.5 = 20). Esto sencillamente fue pasado por alto en análisis anteriores porque Hours Per Week tiene dos capas bajo la jerarquía de tipos secundarios de FTE Ratio. Sólo debe eliminar Hours Per Week del modelo.
4. Después de analizar los tipos secundarios Employee y Contractor, llega a la conclusión de que deben permanecer como entidades (tablas) separadas en el modelo físico. Existen demasiados atributos distintos entre los dos tipos secundarios para considerar combinarlos en la entidad Worker. Al mismo tiempo, dividir el supertipo Worker en dos tipos secundarios no es una opción atractiva porque la relación varios a varios entre Worker y Project se aplica a ambos tipos secundarios y, por lo tanto, tendría que implementarse de manera redundante (y complicada) si se eliminara la entidad Worker. Prepare relaciones uno a uno entre Worker y Employee y entre Worker y Contractor.

Resumen de Pruebe esto

En este ejercicio recorrió las consideraciones comunes para convertir un modelo lógico que contiene supertipos y tipos secundarios en un modelo físico. Mi solución se aprecia en el apéndice B.

Es posible que haya observado que este diseño en particular no maneja el almacenamiento de los datos históricos. Por ejemplo, si un empleado contratado finaliza un contrato y regresa tiempo después con otro contrato, tal vez no aparezcan ambos contratos en la base de datos al mismo tiempo, porque sólo tiene un grupo de fechas de inicio y fin por empleado. Asimismo, si un empleado se va por un tiempo y es recontratado tiempo después, no puede mantener ambas relaciones de empleo en la base de datos al mismo tiempo. Esto es típico de las bases de datos OLTP (procesamiento de transacciones en línea) modernas, en donde se contempla una base de datos diferente, como un almacén de datos, para contener los datos históricos. Los almacenes de datos y otras estructuras de datos para OLAP (procesamiento analítico en línea) se cubren en el capítulo 12.

Autoexamen Capítulo 8

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. Las reglas de negocios se implementan en la base de datos mediante _____.

2. Dos diferencias importantes entre las restricciones de unicidad y de clave principal son _____ y _____.

3. Las relaciones en el modelo lógico se vuelven _____ en el modelo físico.

4. Los nombres de restricciones son importantes porque _____.

5. Al momento de diseñar tablas,

- **A** Cada correlación normalizada se vuelve una tabla.
- **B** Cada atributo en la correlación se vuelve una columna de tabla.
- **C** Las relaciones se convierten en restricciones de comprobación.
- **D** Los identificadores únicos se vuelven desencadenadores.
- **E** Las columnas de clave principal deben definirse como NOT NULL.

6. Los supertipos y los tipos secundarios

- **A** Deben implementarse exactamente como se especifica en el diseño lógico.
- **B** Se pueden contraer en el diseño físico de una base de datos.
- **C** Pueden tener las columnas del supertipo dobladas hacia cada tipo secundario en el diseño físico.
- **D** Suelen tener la misma clave principal en las tablas físicas.
- **E** Se aplican sólo al diseño lógico.

7. Los nombres de tablas

- **A** Deben basarse en los nombres de atributos del diseño lógico.
- **B** Siempre deben incluir la palabra “tabla”.
- **C** Sólo deben utilizar letras mayúsculas.
- **D** Deben incluir los nombres de la organización o la ubicación.
- **E** Pueden contener abreviaturas, cuando sea necesario.

8. Los nombres de columnas

- **A** Deben ser únicos dentro de la base de datos.
- **B** Deben basarse en los nombres de atributos correspondientes del diseño lógico.

C Deben emplear como sufijo el nombre de la tabla.

D Deben ser únicos dentro de la tabla.

E Deben emplear abreviaturas, cuando sea posible.

9. Las restricciones referenciales

A Definen las relaciones identificadas en el modelo lógico.

B Siempre se definen en la tabla primaria.

C Requieren que las claves externas se definan como NOT NULL.

D Deben tener nombres descriptivos.

E Asignan nombres a las tablas primaria y secundaria y a la columna de clave externa.

10. Las restricciones de comprobación

A Pueden utilizarse para obligar a una columna a que coincida con una lista de valores.

B Pueden aplicarse para que una columna coincida con un rango de valores.

C Pueden emplearse para forzar a una columna a coincidir con otra en la misma fila.

D Pueden usarse para obligar a una columna a coincidir con una que existe en otra tabla.

E Pueden servir para imponer una restricción de clave externa.

11. Los tipos de datos

A Evitan que se inserten datos incorrectos en una tabla.

B Sirven para evitar que se guarden caracteres alfabéticos en columnas numéricas.

C Se utilizan para evitar que se guarden caracteres numéricos en columnas con un formato de caracteres.

D Requieren que también se especifiquen la precisión y la escala.

E Se emplean para evitar que se guarden fechas no válidas en columnas de fechas.

12. ¿Cuáles son restricciones de vistas?

A Las vistas que contienen combinaciones nunca se actualizan.

B Se prohíben las actualizaciones a columnas calculadas en las vistas.

C Se requieren privilegios para actualizar datos mediante las vistas.

D Si una vista omite una columna obligatoria, no se permiten inserciones en la vista.

E Cualquier actualización relacionada con una vista puede hacer referencia sólo a las columnas de una tabla.

13. Algunas ventajas de las vistas son

- **A** Siempre proporcionan ventajas en el desempeño.
- **B** Pueden evitar que los usuarios de una base de datos cambien los nombres de tablas y columnas.
- **C** Sirven para ocultar las combinaciones y los cálculos complejos.
- **D** Pueden filtrar columnas o filas que los usuarios no deben ver.
- **E** Se ajustan a las necesidades de los departamentos individuales.

14. Los índices

- **A** Se utilizan como ayuda en las restricciones de clave principal.
- **B** Sirven para mejorar el desempeño de una consulta.
- **C** Se pueden emplear para mejorar el desempeño de inserciones, actualizaciones y eliminaciones.
- **D** Suelen ser más pequeños que las tablas a las que hacen referencia.
- **E** Son más lentos para examinar en forma secuencial que las tablas correspondientes.

15. ¿Cuáles son reglas generales aplicadas a los índices?

- **A** Cuanto más grande sea una tabla, más importantes serán los índices.
- **B** La inclusión de índices en columnas de clave externa suele ayudar en el desempeño de las combinaciones.
- **C** Las columnas que se actualizan con frecuencia siempre deben contener índices.
- **D** Cuantas más veces se actualice una tabla, más ayudarán los índices al rendimiento.
- **E** En las tablas muy pequeñas, los índices tienden a no ser muy útiles.

Parte III

Implementación de una base de datos

Capítulo 9

Conexión de bases de datos al mundo exterior

Habilidades y conceptos clave

- Modelos de despliegue.
- Conexión de bases de datos Web.
- Conexión de bases de datos a aplicaciones.

Este capítulo comienza con una revisión de la evolución de los *modelos de despliegue* de una base de datos: las maneras como las bases de datos se han conectado con sus usuarios y otros sistemas computacionales dentro de la *infraestructura* de computación de una empresa (la estructura interna que organiza todos los recursos de computación de una empresa, que incluye bases de datos, aplicaciones, computadoras y la red). Después se exploran los métodos utilizados para conectar las bases de datos a las aplicaciones que emplean un navegador Web como la principal interfaz de usuario; éste es el modo en que se construyen muchos sistemas de aplicaciones modernos. Por último, se analizan los métodos actuales para conectar las bases de datos a aplicaciones, por ejemplo mediante conexiones ODBC (para casi todos los lenguajes de programación) y diversos métodos para conectar las bases de datos a aplicaciones escritas en Java (un lenguaje orientado a objetos de uso común).

Modelos de implementación

La historia de la industria de la tecnología de la información (TI) es un estudio muy interesante, porque comprueba con claridad el antiguo adagio de que la historia se repite. En ningún lugar es esto más cierto que en el modo en que se implementan las bases de datos, y los sistemas computacionales en general, en las redes empresariales. En las secciones siguientes se reseñan los principales modelos de implementación que se han utilizado. Casi todos estos modelos todavía están en uso activo.

Modelo centralizado

El modelo centralizado, mostrado en la figura 9-1, fue el método original utilizado para conectar bases de datos a la infraestructura de equipos de cómputo de una empresa. Al principio, los usuarios de una base de datos utilizaban "terminales tontas" que ofrecían muy poco poder de procesamiento o programación inteligente. Las únicas funciones de las terminales eran presentar pantallas de datos provenientes de la red, mover el cursor por la pantalla y capturar lo que escribía el usuario, lo que era devuelto a la red. En el otro extremo de la red estaba una supercomputadora u otros servidores grandes centralizados que alojaban todas las otras fun-

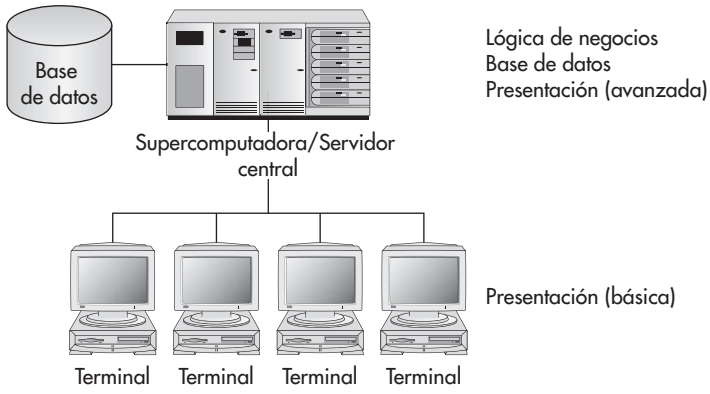


Figura 9-1 El modelo de implementación centralizada.

ciones, entre ellas la lógica de negocios (en los programas de aplicaciones), la base de datos y otras funciones de presentación avanzadas, como la capacidad para preparar imágenes y tablas, y elegir colores para desplegarlas (si las terminales conectadas eran a color).

Comparada con la tecnología que prevalece en la actualidad, esta disposición parece primitiva, pero recuerde que todavía no se habían inventado las computadoras personales. Cuando las PC entraron en escena, algunos de sus primeros usos fueron reemplazar a las terminales tontas, dar a los usuarios un dispositivo personal que pudieran emplear cuando menos para otros propósitos, como procesamiento de textos (o tal vez practicar alguno de los primeros juegos computacionales). Los programas en las primeras PC, llamados *emuladores de terminales*, se hicieron cargo de la conexión a una red de un modo en que la supercomputadora todavía suponía que estaba conectada a laa terminales tontas originales.

El modelo centralizado ofrecía los beneficios siguientes:

- **Administración fácil** Las actualizaciones y el mantenimiento eran directos, porque toda la lógica de aplicaciones y las bases de datos estaban en un punto central.
- **Costos más bajos de mano de obra para desarrollo** Se requerían menos especialistas, porque todo se ejecutaba en una plataforma.
- **Opciones de elevar la productividad al introducir datos** Algunos estudios han demostrado que las elegantes pantallas GUI que aparecieron después, en realidad hicieron más lentos a los usuarios que realizaban tareas repetitivas. Muchos usuarios experimentados de Windows pueden efectuar algunas tareas mucho más rápido mediante el símbolo del sistema (ventana de DOS) en lugar de las herramientas de GUI disponibles. En gran medida, esto se debe al tiempo requerido para mover una mano entre las teclas que se emplean para escribir y el dispositivo para apuntar (ratón, *trackball* y demás). Si todos tuviéramos una tercera mano, o si de algún modo pudiéramos emplear algo diferente para

controlar el dispositivo apuntador (por ejemplo, mediante el movimiento de los ojos o los pies), tal vez esto podría superarse.

Éstas son las desventajas:

- La supercomputadora o el servidor centralizado ofrecía un punto único para las fallas.
- Las pantallas eran muy primitivas, lo que limitaba la interfaz del usuario.
- Hasta la aparición de la PC, una terminal tonta ocupaba un espacio de escritorio excesivo para los propósitos limitados que atendía.

Modelo distribuido

Conforme las redes de computadoras fueron más accesibles a fines de la década de 1970 y principios de la de 1980, la industria de la TI se enamoró del concepto de las bases de datos y aplicaciones distribuidas. En este caso, *distribuidas* significa la división en partes de la aplicación, la base de datos, o ambas, y la colocación de diferentes partes en distintos equipos de cómputo, todas conectadas mediante una red. Cuando se hace correctamente, esta distribución es *transparente* para los usuarios, lo que significa que el sistema oculta los detalles de la distribución a los usuarios, y hace que todo parezca que proviene de un solo origen. En la figura 9-2 se muestra un modelo distribuido simple que emplea dos servidores centralizados.

Por desgracia, la euforia de comercialización que conllevó la aparición inicial del modelo distribuido nunca funcionó debido a los altos costos y problemas de confiabilidad y desempeño. Entre otras cosas, la tecnología de redes no estaba suficientemente madura para manejar la carga. De muchas maneras, las versiones iniciales eran soluciones en busca de problemas para resolver. De una manera muy parecida al Ford Edsel, estas novedosas ideas estaban adelante de su tiempo. Esta arquitectura ha reaparecido desde el surgimiento de redes más avanzadas, entre ellas Internet, y ahora se emplea con éxito para respaldar centros de datos, almacenes

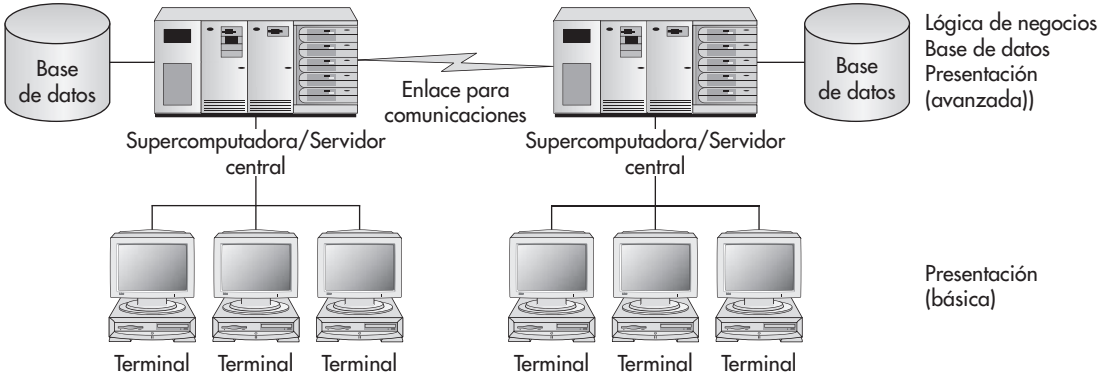


Figura 9-2 El modelo de implementación distribuida.

de datos, sistemas computacionales departamentales, y mucho más. En algunas arquitecturas orientadas a objetos, un agente conocido como *gestor de solicitud de objeto* administra los objetos distribuidos a través de una red, para que las aplicaciones accedan a los objetos sin tomar en cuenta su ubicación. Además, las tendencias actuales en la *computación en rejilla* (*supercomputadoras virtuales formadas por un grupo de computadoras conectadas en red y acopladas libremente*) se pueden considerar extensiones del modelo distribuido original. En verdad la historia se repite.

Los beneficios del modelo de implementación distribuida son:

- Mejoró la tolerancia a fallas, porque cualquier componente implementado en más de un dispositivo ya no es un punto de falla único.
- Mejoró el posible desempeño al colocar los datos y la lógica de aplicaciones más cerca de los usuarios que los necesitan (es decir, los sistemas computacionales departamentales).

Éstas son las desventajas de una implementación distribuida:

- Es mucho más complicado que el modelo centralizado.
- Existen posibles problemas de desempeño relacionados con la sincronización de las actualizaciones de datos de los datos guardados en forma redundante.
- Es mucho más costoso que el modelo centralizado.
- No existen lineamientos e información de las mejores prácticas para particionar los datos y las aplicaciones a través de los dispositivos de cómputo disponibles.

Modelo cliente/servidor

El modelo cliente/servidor incorpora una o más computadoras compartidas, llamadas *servidores*, que se conectan mediante una red a las estaciones de trabajo de los usuarios individuales, llamadas *clientes*. La computación cliente/servidor apareció a fines de la década de 1980, montada sobre una ola de euforia de comercialización por parte de los vendedores de hardware y software nunca antes vista en la industria de la tecnología de la información. Al modelo original utilizado se le llama ahora *modelo cliente/servidor de dos niveles*, que evolucionó hacia lo que se conoce como *modelo cliente/servidor de tres niveles* y, por último, se volvió el *modelo cliente/servidor de N niveles*, también conocido como *modelo de computación por Internet*. Cada uno de ellos se analiza en las secciones siguientes.

Modelo cliente/servidor de dos niveles

El modelo cliente/servidor de dos niveles, que se muestra en la figura 9-3, es casi lo opuesto del modelo centralizado en que todos los negocios y la lógica de presentación se colocan en la estación de trabajo cliente, que suele ser una computadora personal de gran capacidad. Lo único que queda en un servidor centralizado es la base de datos.

El modelo cliente/servidor de dos niveles fue diseñado para aprovechar la superior calidad de presentación e interfaz de usuario de una estación de trabajo moderna. Sin embargo, la

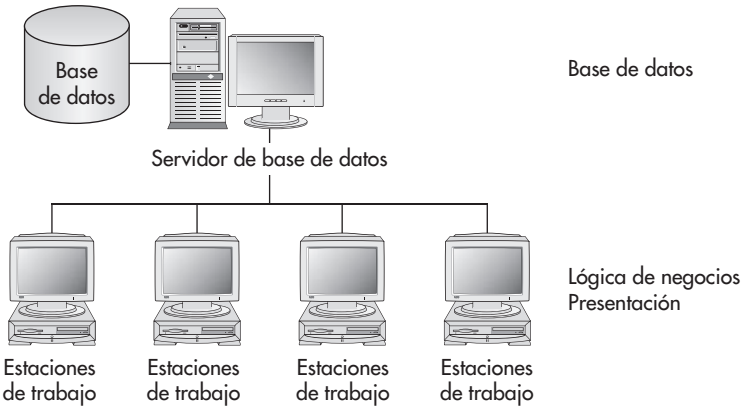


Figura 9-3 El modelo de implementación cliente/servidor de dos niveles.

euforia de comercialización de fines de la década de 1980 y principios de la de 1990 prometía un desarrollo *más rápido* de *mejores* sistemas de aplicaciones a un costo *más bajo*. No resultó de este modo, y es imposible conseguirlo, tal como se analizó en la sección "El triángulo de un proyecto" en el capítulo 5. No obstante, los vendedores ofrecían una solución que "abarcaba todo" y los administradores de negocios de la época estaban muy dispuestos a creerles.

La mentira "blanca" estaba en las comparaciones de costos entre las supercomputadoras y los servidores centrales y las estaciones de trabajo. Los vendedores solían mostrar comparaciones de costos en dólares por millones de instrucciones por segundo (MIPS). El problema era que una instrucción aplicada en las computadoras personales actuales hacía mucho menos que una instrucción aplicada en una supercomputadora o un servidor de gran capacidad. En verdad se comparaban manzanas con naranjas. Los cínicos se referían a los MIPS como "indicador sin sentido de la velocidad de un procesador" y no andaban muy equivocados. El otro factor que casi todos pasaban por alto era que la velocidad con que las computadoras personales leían de sus discos o escribían en ellos no se acercaba remotamente a la conseguida por las supercomputadoras y los servidores de gran capacidad. De modo que trasladar todos los programas de aplicaciones (la lógica de negocios) a las estaciones de trabajo clientes parecía una solución mucho menos costosa, aunque en realidad era un falso ahorro.

Casi todos los proyectos cliente/servidor de dos niveles concluían tarde y muy por arriba de su presupuesto. Además, ocurrieron fallas que regresaron todo a la normalidad. Por ejemplo, el Departamento de Automotores de California gastó 44 millones de dólares en un sistema de registro de vehículos que resultó más lento y menos funcional que el sistema de modelo centralizado al que se suponía que iba a reemplazar. Al final fue desechado con una pérdida total: incluso el hardware era tan especializado que no podía utilizarse para ningún otro propósito, de modo que se fue a la basura. No obstante, algunos proyectos cliente/servidor tuvieron éxito. Por ejemplo, PeopleSoft creó un sistema cliente/servidor de dos niveles

para recursos humanos que fue implementado con éxito por muchas empresas grandes. Por cierto, en los años siguientes PeopleSoft (ahora propiedad de Oracle) migró a un modelo cliente/servidor de N niveles (que se describe más adelante en el capítulo) sin ejecución de código en las estaciones de trabajo clientes, aparte de un navegador Web estándar, y que creció hasta convertirse en una *suite* de aplicaciones para planeación de los recursos de la empresa (ERP) con funciones completas.

Los beneficios del modelo cliente/servidor de dos niveles son:

- Mejoró mucho la interfaz del usuario comparada con los sistemas que empleaban terminales tontas.
- Abrió la posibilidad de un mejor rendimiento, porque el procesador de la estación realizaba todo el trabajo y no tenía que ser compartido con otra persona.

Entre las desventajas del modelo cliente/servidor de dos niveles están:

- Se requerían estaciones de trabajo cliente muy caras porque toda la lógica de la aplicación se ejecutaba en el cliente. La estación de trabajo cliente costaba a menudo entre diez y veinte mil dólares. Para ser justos, los precios del hardware eran mucho más elevados en esa época.
- Ocurrían pesadillas administrativas porque una aplicación se instalaba en todas las estaciones de trabajo clientes, y todas tenían que actualizarse con una nueva versión del software al mismo tiempo.
- El desarrollo resultaba mucho más complicado (y a menudo más costoso) porque el servidor de la base de datos y las estaciones de trabajo clientes casi siempre estaban en plataformas completamente diferentes, que requerían un conjunto distinto de habilidades.

Modelo cliente/servidor de tres niveles

Las numerosas fallas del modelo cliente/servidor de dos niveles provocaron varias revisiones severas. El resultado fue el modelo cliente/servidor de tres niveles, que en esencia devolvió la lógica de las aplicaciones de la estación de trabajo cliente a un servidor central, ahora llamado *servidor de aplicaciones*. En la figura 9-4 se muestra esta arquitectura, que demostró ser muy funcional.

Algunos beneficios del modelo cliente/servidor de tres niveles son:

- Resolvió los problemas administrativos del modelo de dos niveles al centralizar la lógica de las aplicaciones en el servidor de aplicaciones.
- Mejoró la posibilidad de crecimiento, porque se podrían incorporar varios servidores de aplicaciones, conforme fuera necesario. (Se pudo hacer lo mismo con los servidores de base de datos; pero eso requeriría una tecnología de base de datos distribuida para sincronizar todas las actualizaciones de datos a través de todas las copias de los datos.)
- Mantuvo las ventajas de la interfaz de usuario del modelo de dos niveles.

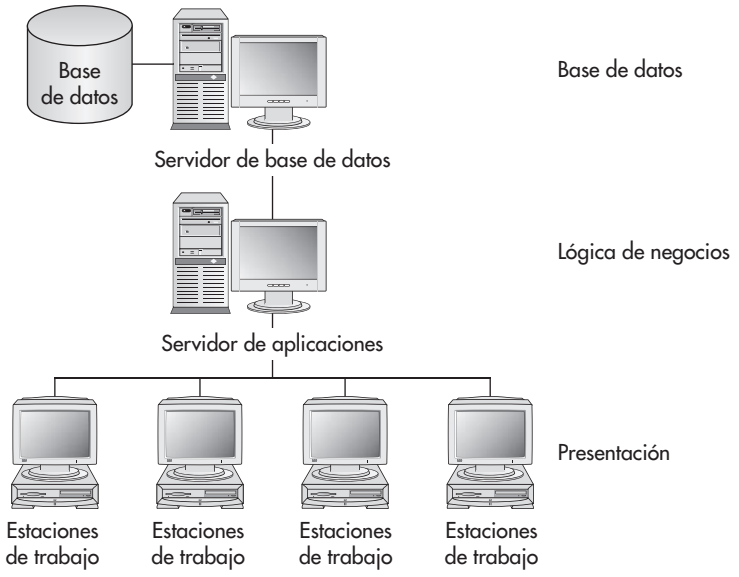


Figura 9-4 El modelo de implementación cliente/servidor de tres niveles.

- Las estaciones de trabajo clientes resultaron mucho menos costosas (las computadoras personales comunes podían fácilmente hacer el trabajo).

Éstas son algunas de las desventajas del modelo cliente/servidor de tres niveles:

- Todavía es más complicado, comparado con el modelo centralizado.
- Los métodos de presentación personalizados y la lógica aumentaban el costo y limitaban la portabilidad entre las plataformas clientes.

Modelo cliente/servidor de N niveles (de computación por Internet)

Conforme los navegadores Web aparecieron en todas partes, los sistemas computacionales empresariales migraron para emplear páginas Web como el principal método de presentación. El modelo cliente/servidor de N niveles (que algunos llaman *modelo de computación por Internet*) se expone en la figura 9-5.

La evolución de tres a N niveles requirió la incorporación de un servidor Web para manejar las respuestas a las solicitudes de los clientes y la representación (formación) de páginas Web, al igual que el cambio de una lógica de implementación patentada en la estación de trabajo a un navegador Web común. La interacción entre el cliente y el servidor Web ocurre así:

1. Mediante el navegador Web, el cliente presenta una solicitud en forma de un localizador uniforme de recursos (URL, Uniform Resource Locator).

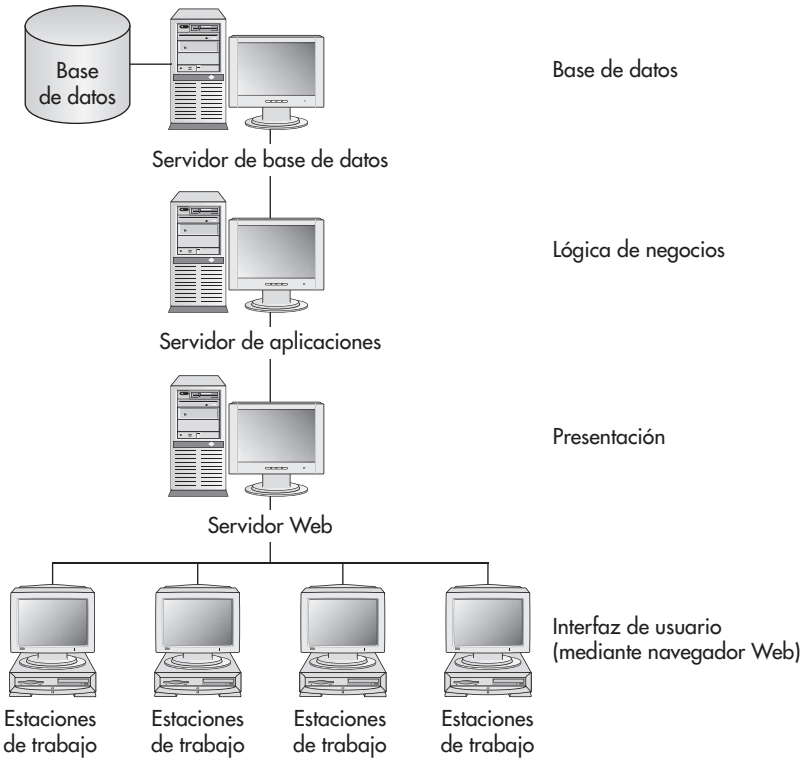


Figura 9-5 El modelo de implementación cliente/servidor de N niveles.

2. El servidor Web procesa la solicitud, integra la página Web solicitada y la envía al cliente.

3. El usuario en la estación de trabajo cliente trabaja con la página Web y, al concluir, presenta una nueva solicitud al servidor Web, y el ciclo se repite.

Esta arquitectura ha alcanzado un enorme éxito en la implementación de sistemas de negocios modernos. Los beneficios del modelo cliente/servidor de N niveles son los siguientes:

- Ofrece un método de presentación estándar en la industria mediante páginas Web.
- Se puede emplear la misma arquitectura para aplicaciones internas (intranet) y externas (Internet).
- Mantiene todos los beneficios del modelo cliente/servidor de tres niveles.
- Puede reducirse la escala de las estaciones de trabajo clientes hasta algo que se denomina *dispositivos de computación en red*, que no poseen un disco duro: se puede decir que es una versión "inteligente" de las originales terminales "tontas".

Éstas son algunas desventajas del modelo cliente/servidor de N niveles:

- Existen desafíos a la seguridad porque Internet y World Wide Web no fueron diseñados con la seguridad en mente.
- Puede requerir equipos de proyecto de desarrollo de mayor tamaño, porque cada capa requiere un especialista.
- Tal vez necesite más hardware. Es posible combinar algunos servidores en dispositivos comunes, pero éste es un método que sólo se recomienda en raras ocasiones porque una separación por funciones mejora la seguridad.

Conexión de bases de datos a Web

Se requiere una extensa "pila de tecnología" para implementar en Internet un sistema de aplicaciones y su base de datos correspondiente. Los componentes básicos se presentan en la figura 9-6. Para ofrecer información integral, se revisa cada componente. Sin embargo, aquí se presta más importancia a la base de datos, por lo que puede consultar otras publicaciones para obtener detalles sobre los demás componentes.

Introducción a Internet y Web

Internet es un conjunto mundial de redes de computadoras interconectadas. Comenzó a fines de la década de 1960 y principios de la de 1970 como ARPANET, del Departamento de Defensa de Estados Unidos. Fue diseñado como medio para conectar sus instalaciones con las de las universidades a las que otorgaba subvenciones para investigación. El protocolo de control de transmisiones/protocolo de Internet (TCP/IP, Transmission Control Protocol/Internet Protocol) fue adoptado como norma en 1982. Otros protocolos son el protocolo para transferencia de archivos (FTP, File Transfer Protocol), el protocolo simple para transferencia de correo (SMTP, Simple Mail Transfer Protocol), Telnet (protocolo de conexión remota), el sistema de nombres de dominio (DNS, Domain Name System), y el protocolo de oficina postal (POP).

Una *intranet* es un segmento de red, que incluye un sitio Web o un grupo de sitios Web, al que sólo pueden acceder los integrantes de una organización. Una *extranet* es una intranet que permite la consulta a personas autorizadas del exterior. Ambas suelen estar protegidas por una *firewall*, que es una puerta de enlace de red dedicada que aplica precauciones de seguridad, de modo que sólo permite el paso al tráfico de la red que cumple ciertos criterios.

World Wide Web es un sistema de hipermedios que proporciona una manera sencilla de estilo "apuntar y hacer clic" para consultar la información de Internet mediante *hipervínculos*. Éstos permiten a los usuarios el desplazamiento por las páginas de una manera no secuencial. Los clientes emplean un navegador Web para presentar las páginas. El servidor Web aloja (guarda y exhibe) páginas y responde a las solicitudes de clientes. Las páginas Web pueden ser *estáticas* (siempre iguales) o *dinámicas* (se pueden modificar y personalizar para una

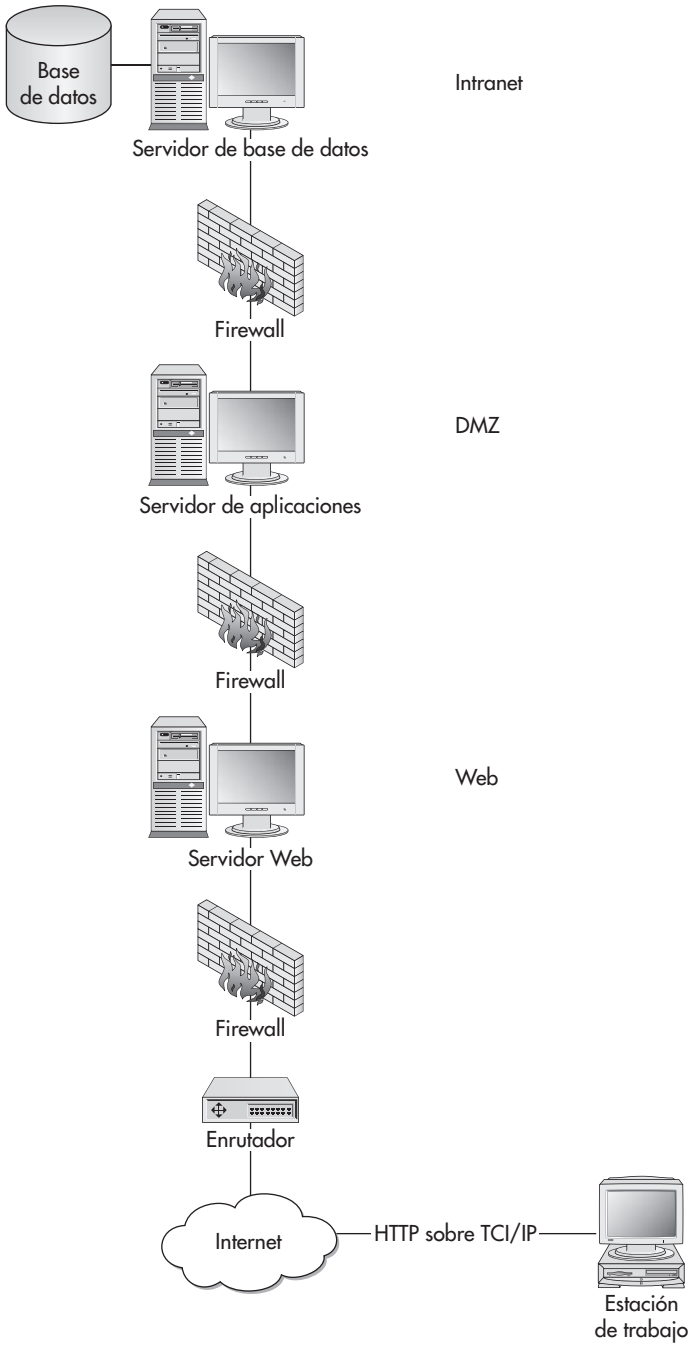


Figura 9-6 Bases de datos conectadas a Web.

solicitud en particular). Las páginas dinámicas son de especial interés en el mundo de las bases de datos, porque son los vehículos para enviar los datos solicitados de la base de datos al usuario de negocios. Una página dinámica suele tener una porción estática (título, texto de ayuda, encabezados de campos de datos) y una dinámica en forma de marcadores de posición en que se colocan el contenido actual y datos convenientes (por ejemplo, un número de cliente y un nombre de cliente) cuando se atiende una solicitud específica del cliente.

El localizador uniforme de recursos (URL) es una cadena de caracteres alfanuméricos que representa la ubicación o dirección de un recurso en Internet y el modo de acceder a él. En algún momento debe traducirse a una dirección IP, un puerto y un protocolo (por ejemplo, HTTP). El formato general de un URL es:

< protocolo>://<host>[:<puerto>]/<ruta absoluta>[¿argumentos?]

En casi todos los navegadores, se entiende que el protocolo es HTTP, si se omite. El host puede ser una dirección IP, pero es más común que sea un nombre de host (por ejemplo, www.Microsoft.com) que se resuelve al buscar la dirección IP del host que emplea un DNS. El puerto predeterminado suele ser 80 (el estándar para HTTP), si se omite. La ruta absoluta identifica la página específica (u otro recurso) solicitada, y el servidor Web selecciona un valor predeterminado, si se omite la ruta. Los argumentos son variables transferidas al servidor Web y se consideran opcionales.

El protocolo de transferencia de hipertexto (HTTP, HyperText Transfer Protocol) se utiliza para transferir páginas Web a través de Internet. Emplea un paradigma basado en solicitudes "sin estado", lo que significa que cada solicitud es tratada como una transacción independiente. La falta de estado dificulta el soporte al concepto de sesión, que es esencial para las transacciones básicas de un DBMS. Es común que se oculten datos en la página Web o en los argumentos del URL para ayudar a Web y a los servidores de aplicaciones a diferenciar las páginas de una sesión de usuario de las de otras.

El lenguaje de marcado de hipertexto (HTML, HyperText Markup Language) es el empleado para aplicar formato a los documentos utilizados para diseñar casi todas las páginas Web. El sistema HTML para marcar, o *etiquetar*, un documento para su publicación en Web fue derivado del lenguaje de marcado general estandarizado (SGML, Standarized General Markup Language), una norma del ISO 1986.

El lenguaje de marcado extensible (XML, eXtensible Markup Language) es una especificación de propósito general para crear lenguajes de marcado personalizados. Mientras que HTML describe una presentación mediante un conjunto fijo de etiquetas, XML describe el contenido y permite a los desarrolladores crear sus propias etiquetas. Aunque XML y HTML no son el mismo lenguaje, algunos llaman a XML "HTML con esteroides". Entre las características de XML está la capacidad para definir un esquema XML, que permite guardar datos en un árbol jerárquico de etiquetas XML dentro del documento XML. Ahora, diversos proveedores de RDBMS dan soporte directo a XML como tipos de datos, y también existen

bases de datos XML de propietario. Sin embargo, las empresas se han rehusado a abandonar las bases de datos relacionales y a aplicar un cambio de paradigma importante en el modo en que organizan y almacenan sus datos. De modo que, hasta el momento, XML se emplea sobre todo para intercambiar datos entre organizaciones en formatos XML que son estándares en la industria. Los comités de normas trabajan en los vocabularios del XML estándar (es decir, etiquetas de datos, estructuras de esquemas y convenciones para utilizarlos) para áreas de datos específicas, como HR-XML Consortium, Inc., que sólo trabaja en datos de recursos humanos (HR). XML se cubre con mayor detalle en el capítulo 13.

Componentes de la "pila de tecnología" de Web

La siguiente es una lista de los componentes mostrados en la figura 9-6, junto con sus funciones:

- La estación de trabajo cliente ejecuta un navegador Web y se comunica en Internet mediante HTTP sobre TCP/IP.
- El sitio Web se encuentra detrás de un *enrutador*, que envía paquetes entre las redes, y una firewall. El enrutador decide cuáles paquetes son transferidos entre Internet y la red secundaria en que reside el servidor Web. Aunque algunos enrutadores aplican un filtro rudimentario, se considera que la protección adicional de una firewall es el mejor medio para proteger al servidor Web de los intrusos.
- El servidor Web es responsable de alojar y exhibir las páginas Web.
- Los URL manejados por el servidor Web provocan que se ejecuten transacciones en el servidor de aplicaciones (esto se detalla en la sección siguiente). El servidor Web suele residir entre un par de firewalls para aislarlo del servidor Web y la intranet, en donde suele residir el servidor de la base de datos. A esta área también se denomina zona desmilitarizada (DMZ, Demilitarized Zone), término tomado de las zonas intermedias entre países rivales en un conflicto militar.
- El servidor de aplicaciones presenta solicitudes SQL (o un lenguaje similar) al servidor de la base de datos cuando se requieren datos de ella.

Invocación de transacciones desde páginas Web

La información que se encuentra en una solicitud Web recibida por el servidor Web invoca una transacción de varias maneras en el servidor de aplicaciones. Estos métodos se detallan en las secciones siguientes.

Interfaz de puerta de enlace común

La interfaz de puerta de enlace común (CGI, Common Gateway Interface) es una especificación para transferir información entre un servidor Web y un programa CGI. El programa de CGI (al que suele llamársele *secuencia de comandos CGI*) se ejecuta en el servidor Web o el de aplicaciones. CGI define cómo se comunican las secuencias de comandos con los ser-

vidores Web. El URL apunta a la secuencia de comandos de CGI, y el servidor la activa. La secuencia de comandos real se puede escribir en diversos lenguajes, como Perl, C o C++. En esencia, en lugar de que el URL de la solicitud que llega apunte directamente a un documento HTML, lo hace a una secuencia de comandos. Se ejecuta ésta, y el resultado es un documento HTML que luego se devuelve al cliente como respuesta a la solicitud.

Entre las ventajas de CGI están las siguientes:

- Sencillez.
- Independencia del lenguaje y el servidor Web.
- Amplia aceptación.

Éstas son algunas desventajas de CGI:

- El servidor Web siempre está entre el cliente y la base de datos.
- No se permiten transacciones (sin estado).
- No está diseñada para intercambios extensos.
- Cada ejecución de CGI activa un proceso (o subproceso) nuevo, lo que genera problemas de recursos.
- CGI no es inherentemente segura.

Inclusiones del lado del servidor (SSI)

Las inclusiones del lado del servidor (Server Side Includes, SSI) hacen que se incrusten comandos en el documento, los que permiten que el servidor Web ejecute un programa (igual que con CGI) e incorpore el resultado en el documento. En esencia, SSI está en una *macro* de HTML. El URL en la solicitud apunta a un documento HTML, pero el servidor examina la sintaxis del documento y maneja cualquier comando de SSI antes de devolver el documento al cliente solicitante. SSI resuelve algunos problemas de desempeño de CGI, pero ofrece algunas ventajas o desventajas diferentes.

Puertas de enlace que no son CGI

Las puertas de enlace que no son CGI funcionan igual que las CGI, excepto que cada una es una extensión patentada de un servidor Web específico de un vendedor. Las dos opciones más populares durante la llamada "era punto com" fueron Netscape Server API y las páginas activas de servidor (Active Server Pages, ASP), parte de los servicios de información de Internet (Internet Information Services, IIS) de Microsoft. Más adelante, la Netscape Server API fue adquirida por Sun Microsystems e incorporada a su línea de productos.

Éstas son algunas ventajas de las puertas de enlace que no son CGI:

- Mejor desempeño que CGI.
- Características y funciones adicionales.
- Ejecución en el espacio de dirección del servidor, en lugar de procesos o subprocesos nuevos.

Entre las desventajas de las puertas de enlace que no son CGI están:

- Son una solución patentada que no se puede trasladar de un servidor Web a otro.
- Posible inestabilidad.
- Son mucho más complejas en comparación con CGI.

Conexión de bases de datos a aplicaciones

Ahora que ha visto cómo interactúan la capa de Web con la del servidor de aplicaciones, necesita comprender cómo se conectan e interactúan las aplicaciones en el servidor de aplicaciones con la base de datos. Casi todas las conexiones entre el servidor de aplicaciones y las bases de datos remotas (es decir, las que se ejecutan en otro servidor) emplean una *interfaz de programación de aplicaciones* (API, Application Programming Interface) común.

Una API es un conjunto de convenciones de llamada que permite que un programa de aplicaciones acceda a los servicios, ya sea proporcionados por el sistema operativo o por otros productos de software, como el DBMS. La API proporciona un *nivel de abstracción* (una capa de generalización que oculta los detalles de una implementación), que permite que la aplicación sea portátil entre diversos sistemas operativos y vendedores.

Conexión de bases de datos mediante ODCB

La *conectividad abierta de base de datos* (Open DataBase Connectivity, ODBC) es una API estándar para conectar programas de aplicaciones a los DBMS. ODBC se basa en una *interfaz en el nivel de la llamada* (Call Level Interfacce, CLI), una convención que define el modo en que se hacen llamadas a los servicios, lo que fue definido primero por el SQL Access Group y publicado en septiembre de 1992. Aunque Microsoft fue la primera compañía que ofreció un producto comercial basado en ODBC, no es un estándar de Microsoft, y de hecho existen versiones para Unix, Macintosh y otras plataformas.

ODBC es independiente de cualquier lenguaje, sistema operativo o sistema de bases de datos específicos. Una aplicación escrita para la API de ODBC puede trasladarse a otra base de datos o sistema operativo con sólo modificar el controlador de ODBC. Este controlador enlaza la API con la base de datos y la plataforma específicas, y una definición conocida como *origen de datos de ODBC* contiene la información necesaria para que una aplicación determinada se conecte con un servicio de bases de datos. En los sistemas Windows, los controladores de ODBC más populares se incluyen con el sistema operativo, porque es un programa de utilerías para definir orígenes de datos de ODBC (se encuentra en el Panel de control o el Panel de herramientas administrativas, dependiendo de la versión de Windows).

Casi todos los productos de software comerciales y casi todas las bases de datos comerciales son compatibles con ODBC, lo que facilita a los vendedores de software vender y dar soporte a productos con una amplia variedad de sistemas de bases de datos. Una excepción

notable son las aplicaciones escritas en Java. Emplean una API diferente conocida como Java Database Connectivity (JDBC), que se examina en la sección "Conexión de bases de datos a aplicaciones de Java", más adelante en el capítulo.

Un dilema común es que los vendedores de bases de datos relacionales no manejan las funciones avanzadas de la misma manera. Este problema se evita al utilizar una cláusula ESCAPE, que indica al controlador de ODBC que pase intactas las instrucciones de SQL a través de la API de ODBC. Por supuesto, la desventaja de este método es que las aplicaciones escritas de esta manera no se pueden trasladar al DBMS de un vendedor diferente (y a veces ni siquiera a una versión diferente del DBMS del mismo vendedor).

Conexión de bases de datos mediante OLE DB

OLE DB (también denominada OLE-DB u OLEDB, Object Linking Embedding: vinculación e incrustación de objetos de bases de datos) es una API diseñada por Microsoft para acceder a diferentes tipos de datos guardados de manera uniforme. Pretende ser un reemplazo de nivel superior de ODBC que permite conexiones a una amplia variedad de bases de datos y archivos no relacionales, como bases de datos y hojas de cálculo. Aunque su nombre incluye OLE, la única semejanza entre OLE y OLE DB es que ambas incluyen interfaces que usan el modelo común de objetos (COM, Common Object Model).

Conexión de bases de datos a aplicaciones de Java

Java comenzó como lenguaje de programación patentado (originalmente llamado OAK) desarrollado por Sun Microsystems. Poco después se convirtió en el lenguaje de programación estándar *de facto* para la computación Web, cuando menos en ambientes diferentes a los de Microsoft. Java es un lenguaje de programación orientado a objetos con seguridad en asignación de tipos, que se emplea para crear componentes de clientes (applets) al igual que componentes de servidor (servlets). Tiene una arquitectura independiente de la máquina, lo que lo vuelve muy portátil entre plataformas de hardware y de sistema operativo.

Es probable que también encuentre los términos *JavaScript* y *JScript*. Se trata de lenguajes de producción de secuencias de comandos con una sintaxis estilo Java, diseñados para efectuar funciones simples en sistemas clientes, como modificar fecha. No son implementaciones con funciones completas de Java y no están diseñados para manejar interacciones con bases de datos, pero pueden efectuar la misma función que una secuencia de comandos de CGI, si se prefiere.

Conectividad de base de datos de Java (JDBC)

JDBC (Java Database Connectivity) es una API, modelada a partir de ODBC, para conectar aplicaciones de Java a una amplia variedad de productos de DBMS relacionales. Algunos controladores de JDBC traducen la API de JDBC a las llamadas de ODBC correspondientes y, por lo tanto, se conectan a la base de datos mediante una fuente de datos ODBC. Otros con-

troladores se traducen directamente a la API cliente propietaria de la base de datos relacional específica, como la Oracle Call Interface (OCI). Igual que con ODBC, está disponible una cláusula ESCAPE para pasar instrucciones de SQL patentado a través de la interfaz.

La API de JDBC ofrece las características siguientes:

- **SQL incrustado para Java** El programador de Java codifica instrucciones de SQL como variables de cadena; las cadenas son transferidas a los métodos de Java, y un procesador de SQL incrustado traduce el SQL de Java a llamadas de JDBC.
- **Asignación directa de tablas de RDBMS a clases de Java** Los resultados de las llamadas a SQL se ubican automáticamente en variables de clases de Java. Después, el programador en Java puede operar sobre los datos devueltos como si fueran objetos nativos de Java.

Java SQL

Java SQL (JSQL) es un método para incrustar instrucciones de SQL en Java sin tener que crear una codificación especial para colocar las instrucciones en cadenas de Java. Se trata de una extensión de la norma ISO/ANSI para SQL, incrustada en otros lenguajes de host, como C. En el programa de origen se ejecuta un programa especial llamado *precompilador*, que traduce automáticamente a Java puro las instrucciones de SQL escritas por el programador en Java. Este método ahorra un considerable esfuerzo de desarrollo.

Soluciones de middleware

El *middleware* se define como software que sirve de mediador entre un programa de aplicaciones y los servicios disponibles en una red, o entre dos programas de aplicaciones dispares. En el caso de conexiones a una base de datos de Java, los productos de middleware como Java Relational Binding (JRB) de O2 Technology (adquirida por Unidata en 1997) pueden hacer que el RDBMS se vea como si se estuviera ejecutando una base de datos orientada a objetos en un servidor remoto. Luego, el programador en Java accede a la base de datos mediante métodos comunes de Java, y el producto de middleware se ocupa de la traducción entre los objetos y los componentes de la base de datos relacional.

Pruebe esto 9-1 Exploración de World Wide Web

En este ejercicio explorará algunos aspectos de World Wide Web, con los que repasará los conceptos descritos en este capítulo.

Paso a paso

1. Consulte varios de sus sitios Web favoritos, en especial si tiene una cuenta que le permite revisar información dinámica como transacciones, puntos de recompensa por viajes, objetos para subastas, etcétera.

(*continúa*)

2. Observe cuáles partes de las páginas son estáticas y cuáles dinámicas (que cambian de acuerdo con lo que escribe o las opciones que elige).
3. Analice cuáles partes son listas de datos que podrían provenir de una base de datos. Algunas de las bases de datos más grandes del mundo dan soporte a sitios Web para operaciones bancarias, subastas en línea y funciones similares. Sin embargo, es probable que no pueda identificar de cuál vendedor es el DBMS detrás del sitio, con sólo revisar las páginas Web.
4. Fíjese en el URL que muestra su navegador cuando se desplaza por las páginas. ¿Puede detectar algunos argumentos (parámetros que comienzan con un signo de interrogación)? ¿Puede identificar algún URL que apunte a un contenido diferente que las páginas HTML (tipos de archivos como JSP, PHP y CGI)? Si ve tipos de archivos que no reconoce, utilice su motor de búsqueda preferido para identificarlos.

Resumen de Pruebe esto

En este ejercicio utilizó World Wide Web para observar algunos de los conceptos presentados en este capítulo.

✓ Autoexamen Capítulo 9

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. En el modelo de implementación centralizada,

   A Un servidor Web contiene todas las páginas Web.
   B Se utiliza una terminal “tonta” como estación de trabajo cliente.
   C La administración es muy fácil porque todo está centralizado.
   D No existen puntos únicos de fallas.
   E Los costos de desarrollo suelen ser muy altos.

2. En el modelo de implementación distribuida,

   A La base de datos, la aplicación, o ambas están particionadas y se implementa en varios sistemas computacionales.
   B Las implementaciones iniciales tuvieron mucho éxito.

C La distribución puede ser transparente para el usuario.

D Los costos y la complejidad son reducidos en comparación con el modelo centralizado.

E La tolerancia a fallas es mejor en comparación con el modelo centralizado.

3. En el modelo cliente/servidor de dos niveles,

A Toda la lógica de las aplicaciones se ejecuta en un servidor de aplicaciones.

B Un servidor Web contiene las páginas Web.

C La estación de trabajo cliente maneja toda la lógica de presentación.

D La base de datos se aloja en un servidor centralizado.

E Las estaciones de trabajo cliente deben ser sistemas con gran potencia.

4. En el modelo cliente/servidor de tres niveles,

A Toda la lógica de las aplicaciones se ejecuta en un servidor de aplicaciones.

B Un servidor Web contiene las páginas Web.

C La estación de trabajo cliente maneja toda la lógica de presentación.

D La base de datos se aloja en un servidor centralizado.

E Las estaciones de trabajo cliente deben ser sistemas con gran potencia.

5. En el modelo cliente/servidor de N niveles,

A Toda la lógica de las aplicaciones se ejecuta en un servidor de aplicaciones.

B Un servidor Web contiene las páginas Web.

C La estación de trabajo cliente maneja toda la lógica de presentación.

D La base de datos se aloja en un servidor centralizado.

E Las estaciones de trabajo cliente deben ser sistemas con gran potencia.

6. Internet

A Comenzó como ARPANET del Departamento de Educación de Estados Unidos.

B Data de fines de la década de 1960 y principios de la de 1970.

C Siempre utilizó TCP/IP como estándar.

D Es un conjunto mundial de redes de computadoras interconectadas.

E Permite varios protocolos, entre ellos HTTP, FTP y Telnet.

7. Un URL puede contener

A Un protocolo.

B Un nombre de host o una dirección IP.

C Un puerto.

D La ruta absoluta hacia un recurso en un servidor Web.

E Argumentos.

8. Una intranet está disponible para _____.

9. Una extranet está disponible para _____.

10. World Wide Web emplea _____ para desplazarse por las páginas.

11. HTTP no permite directamente el concepto de sesión porque _____.

12. XML es extensible porque es posible definir _____.

13. Las soluciones de middleware para conexiones de Java hacen que el RDBMS se vea como _____.

14. La "pila de tecnología" de Web incluye

A Una estación de trabajo cliente que ejecuta un navegador Web.

B Un servidor Web.

C Un servidor de aplicaciones.

D Un servidor de la base de datos.

E Hardware de red (firewall, enrutadores y demás).

15. Las ventajas de CGI son

A No tiene estado.

B Su uso es sencillo.

C Resulta inherentemente segura.

D Es ampliamente aceptada.

E Es independiente del lenguaje y del servidor.

16. Las inclusiones del lado del servidor (SSI)

A Son comandos incrustados en un documento Web.

B Son puertas de enlace que no son CGI.

C Son macros de HTML.

D Resuelven algunos de los problemas de desempeño de CGI.

E Son inherentemente seguras.

17. Algunas ventajas de una puerta de enlace que no es CGI son

A Es reconocida por su estabilidad.

B Es una solución patentada.

C Ofrece mejor seguridad que las soluciones de CGI.

D Es más sencilla que CGI.

E Funciona en el espacio de direcciones de un servidor.

18. ODBC es

A Una API estándar para conectarse a un DBMS.

B Independiente de cualquier lenguaje, sistema operativo o DBMS específico.

C Un estándar de Microsoft.

D Utilizada por programas de Java.

E Flexible para manejar el SQL patentado.

19. JDBC es

A Una API estándar para conectarse a un DBMS.

B Independiente de cualquier lenguaje, sistema operativo o DBMS específicos.

C Un estándar de Microsoft.

D Utilizada por programas de Java.

E Flexible para manejar el SQL patentado.

20. JSQL es

A Una norma de Sun Microsystems.

B Un método para incrustar instrucciones SQL en Java.

C Una extensión de una norma ISO/ANSI.

D Una solución de middleware.

E Independiente de cualquier lenguaje, sistema operativo o DBMS específicos.

Capítulo 10

Seguridad de una base de datos

Habilidades y conceptos clave

- ¿Por qué es necesaria la seguridad?
- Seguridad del servidor de base de datos.
- Seguridad de clientes y aplicaciones de base de datos.
- Seguridad del acceso a una base de datos.
- Monitoreo y auditoría de la seguridad.

La seguridad se ha convertido en una consideración esencial en los sistemas modernos. Nada puede ser más vergonzoso para una organización que una historia en los medios relacionada con datos confidenciales o secretos comerciales que fueron sustraídos electrónicamente de sus sistemas de cómputo. En este capítulo se analiza la necesidad de seguridad, las consideraciones de seguridad para desplegar servidores de bases de datos y clientes que accedan a esos servidores, y los métodos para implementar seguridad en el acceso a la base de datos. Concluye con un análisis de la vigilancia y la verificación de la seguridad.

¿Por qué es necesaria la seguridad?

La Ley de Murphy afirma que si algo puede salir mal, saldrá mal. Los profesionales de tecnología de la información con experiencia en seguridad le dirán que Murphy era un optimista. Servidores colocados en Internet con las configuraciones y las contraseñas predeterminadas han quedado en riesgo en *minutos.* Las contraseñas predeterminadas de una base de datos y las vulnerabilidades de seguridad comunes son ampliamente conocidas. A principios de 2003, el gusano Slammer infectó decenas de miles de bases de datos de Microsoft SQL Server configuradas con una cuenta de administrador del sistema (SA) predeterminada que no tenía contraseña. Y resulta curioso que el peor daño fue la pérdida del servicio cuando las computadoras infectadas enviaron cientos de miles de paquetes a la red en busca de otras computadoras para infectar. Si se cree que esto no puede ocurrirle, piénselo dos veces.

Éstas son algunas razones por las que debe diseñar la seguridad de sus sistemas computacionales:

- Las bases de datos conectadas a Internet, o cualquier otra red, son vulnerables a hackers y otros delincuentes que buscan dañar o robar datos. Entre ellos están los siguientes:
  - Espías de sus competidores que están detrás de sus secretos.
  - Hackers interesados en alcanzar notoriedad por haber penetrado en sus sistemas.

  - Personas interesadas en obtener algo con un valor económico.
  - Empleados descontentos; es extraño que nunca nos enteremos de los empleados contentos, sólo de los descontentos.
  - Fanáticos interesados en hacer una declaración política a costa de su organización.
  - Personas emocionalmente desequilibradas o simplemente perniciosas.
- Empleados (u otras personas) que pretenden cometer un fraude. Cualquier auditor bancario le dirá que 80% de los fraudes son cometidos por empleados. No suponga que su sistema es inmune sólo porque no se puede acceder a él desde Internet.
- Los errores cometidos por usuarios autorizados que son honestos pueden provocar riesgos para la seguridad, pérdida de datos y errores de procesamiento.
- Los controles de seguridad hacen que las personas sigan siendo honradas del mismo modo en que lo hacen las cerraduras en los hogares y oficinas.

Seguridad del servidor de base de datos

Esta sección se concentra en las consideraciones de seguridad para el servidor de una base de datos. Cuando reflexione sobre la seguridad, debe comenzar por un extremo de la red u otro (en la estación de trabajo cliente del usuario o en el servidor de base de datos) y recorrer sistemáticamente todos los componentes intermedios. Éste es el único modo de que no pase nada por alto. Aquí comenzará con el servidor de la base de datos y avanzará desde ahí.

Seguridad física

La seguridad física del servidor es un ingrediente esencial. El servidor debe estar en una habitación cerrada, a la que sólo tenga acceso el personal autorizado. Nada es más embarazoso que alguien robe o dañe el servidor de una base de datos o los discos duros que guardan la información. Una vez que un ladrón se ha ido con el hardware, tiene todo el tiempo del mundo y toda la privacidad que necesita para entrar al sistema hasta tener acceso a los datos. Además, es más fácil poner en riesgo los sistemas a través de la consola del servidor que de manera remota; por lo tanto, el acceso "inmediato" a los servidores debe controlarse estrictamente. Dependiendo de la condifencialidad de los datos que contiene la base de datos, es posible que se requieran las medidas adicionales siguientes:

- Se puede instalar un sistema de vigilancia con video.
- Se pueden emplear dispositivos de seguridad que deben poseer los administradores para obtener acceso. Éstos van desde tarjetas con llaves que deben insertar en el servidor para conseguir el acceso, hasta dispositivos criptográficos en que debe introducirse un número de identificación personal (PIN, Personal Identification Number) para obtener una contraseña. Algunos de estos dispositivos están sintonizados con satélites y cambian cada minuto la clave de cifrado utilizada para generar contraseñas.

- Se pueden instalar dispositivos biométricos, que examinan una huella digital o la retina para permitir el acceso.
- Se pueden crear políticas que obliguen a que cuando menos dos empleados estén en la habitación cada vez que alguien se conecta al servidor.
- Se pueden establecer políticas relacionadas con el retiro del hardware y software del lugar de trabajo. Éste es un ejemplo real: una vez trabajé en una institución financiera en donde se revisaba a los empleados cuando se alejaban de las instalaciones. Estaba estrictamente prohibido el retiro de cualquier hardware o material, como listas de computadoras, documentos microfilmados o medios como cintas y discos. Sin embargo, existía un hueco risible. Se permitía colocar *cualquier cosa* en un sobre dirigido a un domicilio propio (o al de otra persona) e introducirlo en los buzones para envío de correo. No sólo reenviaba el sobre sin una inspección, sino que incluso la empresa *pagaba el porte,* sin hacer preguntas. No piense mal, la única vez que vi aplicada esta técnica fue para enviar juegos de computadora a otro lugar, pero el riesgo para la seguridad era enorme.

Seguridad de red

Debe ser obvio que la seguridad física no es suficiente cuando se tiene acceso al servidor de base de datos a través de una red. Los intrusos que consiguen obtener una conexión de red con el servidor pueden trabajar fuera de la habitación del servidor o, en el caso de los servidores conectados a Internet, desde cualquier parte del mundo. Además, debido a que los clientes u otros servidores (como el servidor de aplicaciones) pueden conectarse al servidor de la base de datos, debe adoptar un método holístico para la seguridad de la red, y no sólo comprobar que la red sea segura, sino también que *todo* el sistema de equipos de cómputo conectados a esa red lo sea.

Los detalles completos de la manera de asegurar una red están más allá del alcance de este libro. Sin embargo, en las secciones siguientes se presenta un resumen de los problemas de seguridad de red que debe considerar. Observe que se utiliza el término *red empresarial* para denominar a una red privada que conecta los recursos de computación de la empresa.

Aislamiento entre la red empresarial e Internet

Si la red empresarial está conectada a Internet, debe aislarse para que los hackers que pululan en Internet no puedan ver las interioridades de la red empresarial ni consigan un acceso fácil a ella. Entre las medidas que deben considerarse están las siguientes:

Configurar el enrutador El enrutador que conecta la red empresarial a Internet debe configurarse de manera adecuada. Recuerde que un *enrutador* es un dispositivo que envía paquetes de datos entre las redes mediante reglas contenidas en una *tabla de enrutamiento*. Un paquete es sólo un segmento de un mensaje que se transmite en una red. Los dispositivos de red dividen los mensajes en paquetes de tamaño uniforme para un manejo eficaz. El enrutador

debe configurarse para que sólo los paquetes de datos adecuados sean enrutados hacia la red local. Algunos enrutadores efectúan un filtrado limitado de paquetes, pero no suelen analizar su contenido más allá de la dirección IP destino, que se encuentra en el encabezado del paquete, y toman decisiones acerca del mejor modo de enrutar un paquete con base en la dirección destino y la tabla de enrutamiento.

Emplear una firewall En una red empresarial, cada capa debe estar protegida por una firewall, y las reglas de seguridad aplicadas por ésta deben ser más estrictas en cada capa. En la figura 9-6 del capítulo 9 se muestra esta disposición. Una firewall se implementa mediante software en una computadora de propósito general o un dispositivo de hardware especializado que incluye su propio sistema operativo y software de filtrado. El propósito de la firewall es evitar el acceso no autorizado al segmento de la red que protege (es decir, los recursos computacionales conectados a la parte de la red que está *dentro* de la firewall). Todos los paquetes de datos que pasan de la red externa al segmento (al que suele llamársele *subred*) que se encuentra dentro de la firewall deben aprobar los criterios de seguridad impuestos por la firewall o serán rechazados. La firewall emplea los métodos siguientes:

- **Filtrado de paquetes** Se revisa el contenido de cada paquete que entra o sale de la red para asegurar que se cumplen las reglas definidas por el usuario. Aunque el filtrado de paquetes es eficaz, está sujeto a la *falsificación de IP*, en que un hacker se disfraza como un usuario legítimo al incorporar una dirección IP legítima que es aceptable para la firewall en un mensaje ilegítimo. Para evitar que su red sea utilizada para lanzar algo conocido como *ataques de zombies*, su firewall siempre debe estar configurada para rechazar paquetes del exterior que tengan direcciones IP de retorno que no sean legítimas para la red empresarial. Un ataque de zombies ocurre cuando un intruso instala un programa falsificado en uno de sus servidores, que cada tiempo especificado, despierta y comienza a enviar cientos o miles de paquetes por minutos a un sistema de destino, que suele ser el sitio Web de un empresa contra la cual el atacante siente rencor, en un intento por saturar el sistema atacado, con el fin de inutilizarlo. A este tipo de ataque se le denomina *negación del servicio* (DoS, Denial of Service).
- **Puerta de enlace de aplicación** Las diversas aplicaciones de red (HTTP, FTP, Telnet, etc.) emplean puertos predeterminados distintos. Por ejemplo, HTTP utiliza, como opción predeterminada, el puerto 80. Los puertos que no se utilizan deben desactivarse. Configure *siempre* la firewall para que *sólo* abra los puertos que son *absolutamente necesarios* para sus negocios normales.
- **Puerta de enlace en el nivel de circuitos** Por razones de eficiencia, esta característica aplica mecanismos de seguridad cuando se establece una conexión; después de establecida, permite que los paquetes fluyan con libertad por la conexión. Por lo general, una firewall debe configurarse para que *sólo* se puedan establecer conexiones desde *su interior*; deben rechazarse los intentos realizados desde el exterior de la firewall para establecer conexiones con los recursos del interior (excepto los específicamente autorizados).

- **Servidor simulado** Las firewall traducen todas las direcciones IP utilizadas en la red protegida a direcciones diferentes cuando pasan los paquetes, y suelen asignar a cada una un puerto diferente, para clasificar la respuesta a esos paquetes y devolverlas a quien las originó. Esta característica, conocida como *traducción de dirección de red* (NAT, Network Address Translation), oculta la red interna al mundo exterior.

Proporcionar conexiones seguras a los empleados que trabajan fuera del lugar

Estos trabajadores representan un riesgo especial. Si están conectados a un servicio de banda ancha de Internet, como DSL o cable, en esencia están en una red de área local (LAN, Local Area Network) con muchos otros usuarios de ese servicio específico. Por lo tanto, si estos empleados sólo deben conectar sus computadoras personales directamente al DSL o cable sin otras precauciones, cualquier dispositivo compartido que tengan (discos duros, impresoras, etc.) automáticamente es compartido por todos sus *vecinos* de la misma LAN. Todo lo que debe saber un intruso es cómo hacer clic en Conexiones de red y después en Toda la red, y todos los sistemas no protegidos de la LAN estarán ahí, listos para tomarlos. A menudo, los atacantes sólo están a una contraseña de distancia para consultar todo en un sistema de destino. Dos precauciones evitan el problema.

Debe colocarse un dispositivo de seguridad, que suele ser una combinación de enrutador/interruptor de red/firewall entre el DSL o módem de cable y cualquier computadora utilizada en el hogar. Un beneficio adicional aquí es que el usuario puede conectar varias computadoras al dispositivo de alta velocidad mientras paga por una sola dirección IP al proveedor de servicios de Internet (ISP, Internet Service Provider) (no olvide que algunos ISP prohíben esta práctica). El dispositivo "traduce" automáticamente cualquier dirección IP dentro de la red del hogar a la dirección IP pública asignada por el ISP para la conexión de banda amplia, mediante diferentes puertos para diferenciar las distintas conexiones. El autor utiliza ese dispositivo en el servicio de Internet por cable de su casa y ha visto intentos de hackers por explorar los puertos y detectar los recursos de la red. Una *exploración de puerto* es una técnica que suelen utilizar los hackers: lanzan un programa especial que prueba todos los puertos imaginables en una dirección IP y registra los que están activos para que pueda tratar de utilizarlos con el fin de irrumpir en el sistema de destino. Los intentos de intrusión ocurren con una frecuencia *alarmante*, en ocasiones varias veces en una sola hora. Si instala una red no protegida en su casa, es probable que sea invadida a las pocas *horas* de ser activada. Observe que Microsoft Windows XP y Vista incluyen una firewall de software configurable. No obstante, casi todos los expertos en seguridad prefieren una firewall externa en un dispositivo de hardware dedicado, porque ofrece una mejor protección.

Además, es posible emplear una técnica de red segura conocida como *red privada virtual* (VPN, Virtual Private Network) al conectarse desde Internet a la red empresarial. Este método cifra todos los paquetes de datos y aplica otras medidas para corroborar que los paquetes no le sean útiles a cualquier persona no autorizada que los intercepte, y que no puedan ser alterados y retransmitidos por los hackers. Por lo general esta técnica se implementa mediante un

software especial de un vendedor comercial junto con un dispositivo pequeño que el usuario remoto utiliza para generar una contraseña única cada vez que se conecta de manera remota a la red empresarial. Sin el dispositivo (y sin el número de identificación personal que lo acompaña), el probable hacker no tiene oportunidad de penetrar la red empresarial mediante la VPN.

Aseguramiento de todos los puntos de acceso a una red inalámbrica

Los *puntos de accesos inalámbricos* son dispositivos de red que reciben señales de radio desde los dispositivos de un equipo de cómputo equipado con adaptadores inalámbricos de red, que los conectan a la red con cables de una oficina. Casi todas las redes inalámbricas se apegan a una versión del protocolo estándar de red conocido como 802.11. Los puntos de acceso inalámbrico se han vuelto económicos (menos de 100 dólares) y, por lo tanto, frecuentes, porque a las personas les agrada desplazarse por su casa y oficina con libertad, sin tener que arrastrar un cable de red. Sin embargo, los puntos de acceso inalámbricos requieren atención especial, porque un intruso puede acceder a una red inalámbrica desde fuera de sus dominios sin tener que pasar por los enrutadores y la firewall cuidadosamente instalados para evitar esa intrusión. Abundan las historias de horror en las publicaciones de TI sobre un usuario desprevenido que instala un punto de acceso inalámbrico no autorizado en una oficina, lo inserta en la conexión de red más cercana y proporciona a todas las personas que están a una distancia de 25 a 50 metros un acceso abierto a la red. Como opción predeterminada, muchos de estos dispositivos no tienen habilitada *absolutamente* ningún cifrado u otros controles de acceso, con lo que permiten el acceso a cualquier persona con una computadora con funciones inalámbricas en las oficinas cercanas, en el estacionamiento o incluso en un edificio al otro lado de la calle. Lo peor de todo es que, una vez que se conecta el intruso, puede acceder a la intranet, completamente *en el interior* de todas las firewall y otros controles que se han implementado cuidadosamente para proteger la red de los intrusos.

Si cree que esto no puede ocurrirle, a continuación se presentan algunos ejemplos de la vida real:

- En un viaje reciente a un consultorio médico, la laptop del autor, equipada con un adaptador de red inalámbrico 802.11g, se conectó *automáticamente* a una red inalámbrica de un consultorio vecino. No me puse a ver lo que habría podido conseguir en cuanto a equipos de cómputo, discos compartidos, archivos, etc., pero el personal desconocía por completo que alguien podía conectarse a su red inalámbrica. No comprendía que las paredes no son una barrera para las redes inalámbricas. Por cierto, un vistazo rápido a la pantalla del adaptador inalámbrico mostró otras dos redes vulnerables que se podían consultar desde la misma sala de espera. Una de ellas incluso tenía el nombre de red predeterminado que viene con el punto de acceso inalámbrico, de modo que era probable que la contraseña para el enrutador también fuera la de fábrica. Un intruso hubiera podido reconfigurar toda su red antes de que ellos supieran qué había ocurrido.

- Hace poco, en un paseo por Market Street, en San Francisco, el adaptador inalámbrico de la misma laptop detectó un promedio de tres redes inalámbricas en cada calle; una cantidad sorprendente de ellas estaba completamente abierta a quien quisiera conectarse.
- Una gerente de TI informó que después de descubrir que la red de su compañía había sido consultada subrepticiamente desde un punto de acceso inalámbrico no autorizado, trató de detectarlo y fracasó tras varios intentos. Por último, contrató a un consultor que poseía un dispositivo para rastrear la señal falsificada. (Tal vez no lo crea, pero un tubo de patatas fritas recubierto con papel aluminio es una estupenda antena direccional para "detectar" puntos de acceso inalámbricos.) El consultor lo encontró oculto en el techo de una sala de conferencias. La persona que lo había instalado sabía que era contra las reglas, pero no quería molestarse en usar una conexión física para su laptop. No es necesario decir que esa persona perdió su empleo, pero quién sabe qué obtuvieron los intrusos antes de que desapareciera el punto de acceso no autorizado.

Establecer una política de seguridad inalámbrica La política de seguridad de su organización debe atender las conexiones inalámbricas, disponer que sólo los administradores capacitados se encarguen de instalarlas y configurar los estándares para un funcionamiento adecuado.

Cifrado obligatorio Los estándares deben obligar a que se habilite el cifrado en todos los puntos de acceso inalámbricos. Todos los puntos de acceso que se venden en el mercado tienen incorporada una opción de cifrado, y se requieren sólo algunos minutos para habilitar esta característica e introducir una frase que debe proporcionar cualquier dispositivo que trate de conectarse con el fin de tener acceso a la red.

Limitar el acceso mediante una lista de direcciones MAC En la actualidad, todos los dispositivos de red que se fabrican tienem una dirección de control de acceso de medios (MAC, Media Access Control) asignada por el fabricante. Casi todos los puntos de acceso inalámbrico permiten la entrada de una lista de direcciones MAC que limita el acceso a la red *sólo* a los dispositivos que aparecen en la lista. O bien, la lista de direcciones MAC puede contener una lista de dispositivos a los que *no* se permite conectarse.

Seguridad en el nivel del sistema

Una vez que la red está lo más asegurada posible, la siguiente área de atención es el sistema que ejecutará el DBMS. Un servidor de la base de datos deficientemente asegurado ofrece muchas rutas no revisadas para uso de los intrusos. Éstas son algunas medidas que vale la pena considerar:

Instalar un mínimo de software del sistema operativo Instale sólo el mínimo de componentes de software requeridos para realizar el trabajo, sobre todo en un servidor de producción. Evite las opciones de instalación predeterminadas o "típicas" y aplique la opción de instalación "personalizada" para elegir sólo los componentes necesarios. Por ejemplo, en

los servidores Unix para producción, debe tener la costumbre de eliminar la utilería "Make" y los compiladores del lenguaje C después de que concluya una instalación. Los hackers tienen dificultades para instalar cosas cuando las herramientas necesarias para realizar instalaciones de software no existen en el servidor.

Emplear un mínimo de servicios del sistema operativo Apague o elimine los servicios del sistema operativo que no sean necesarios. En particular, no deben ejecutarse servicios de comunicaciones como FTP, a menos que se requieran de manera específica. En los sistemas Windows, es buena idea deshabilitar Tipo de inicio para los servicios no requeridos. Esto imposibilita el inicio de esos servicios, a menos que tenga privilegios de administrador.

Instalar un mínimo de software DBMS Cuantas menos funciones del DBMS tenga instaladas, menos expuesto estará a tener problemas como vulnerabilidades por desbordamiento de búfer. El DBA debe colaborar con quienes desarrollan aplicaciones para crear una lista consolidada de las funciones necesarias del DBMS. Una vez que la tenga, emplee la opción de instalación personalizada del DBMS y aplique sólo instalaciones mínimas.

Aplicar los parches de seguridad de manera oportuna Establezca un programa en que se revisen las alertas de seguridad en el momento en que se anuncien y se apliquen de manera oportuna las medidas preventivas, entre ellas parches y reparaciones. Los parches deben ser revisados minuciosamente en un entorno de desarrollo durante un periodo finito, antes de su aplicación en un ambiente de producción.

Cambiar todas las contraseñas predeterminadas Las contraseñas predeterminadas deben modificarse a frases o palabras que sean difíciles de adivinar o descubrir mediante *fuerza bruta*, un método que prueba repetidamente las posibilidades hasta que consigue el acceso.

Seguridad de clientes y aplicaciones de base de datos

Un *cliente de base de datos* es cualquier sistema computacional que se registra directamente en el servidor de base de datos. Por lo tanto, el servidor de aplicaciones casi siempre es un cliente de la base de datos, junto con la estación de trabajo cliente de cualquier persona en la organización que tenga privilegios de registro en la base de datos. El DBMS suele requerir la instalación de software cliente en estos sistemas para facilitar la comunicación entre el cliente de la base de datos y el DBMS empleando cualquier mecanismo de comunicaciones especializado requerido por el DBMS.

Credenciales para inicio de sesión

Cada usuario de una base de datos que se conecta a ella debe proporcionar credenciales adecuadas para establecer la conexión. Éstas suelen ser una identificación de usuario y una

contraseña. Tenga cuidado de establecer credenciales que no queden en riesgo con facilidad. Éstas son algunas cosas que deben tomarse en cuenta:

- Las credenciales no deben ser compartidas por varios usuarios de la base de datos.
- Debe elegir contraseñas que no sean fáciles de adivinar. Una política de seguridad debe establecer los estándares mínimos para la seguridad de las contraseñas, entre ellas una longitud mínima, la mezcla de letras mayúsculas y minúsculas, números y caracteres especiales, y evitar palabras que se encuentren en un diccionario.
- Las contraseñas deben cambiarse con regularidad, como cada 30 o 45 días. Los expertos en seguridad no se ponen de acuerdo sobre la eficacia de los cambios periódicos de contraseñas, pero casi todos los auditores de TI insisten en esta práctica.
- Cualquier contraseña revelada debe modificarse de inmediato.
- Las contraseñas nunca deben anotarse, y deben cifrarse cuando se guardan de manera electrónica.

Cifrado de datos

El *cifrado* es la traducción de datos a un código secreto que no puede ser leído sin el uso de una contraseña o clave secretas. Los datos no cifrados se consideran *texto simple*, mientras que los cifrados son *texto cifrado*.

Algunos esquemas para cifrado emplean una *clave simétrica*, lo que significa que se utiliza una sola clave para cifrar el texto simple y para descifrar el texto cifrado. A esta forma se le considera menos segura en comparación con el uso de *claves asimétricas*, en donde se aplican dos claves: una clave *pública* y una *privada*. Lo que cifra la clave pública, lo descifra la clave privada, y viceversa. Los nombres provienen del uso esperado de las claves: la clave pública se entrega a todas las personas que hacen negocios con la empresa, y la clave privada es confidencial e interna para la empresa.

Éstos son algunos lineamientos que debe seguir en relación con el cifrado:

- Las claves de cifrado deben tener un mínimo de 128 bytes de longitud. Cuanto más extensa sea la clave, más segura se considerará (dentro de lo razonable). Sin embargo, las claves más extensas retardan el proceso de descifrado, de modo que debe buscarse una solución intermedia.
- La pérdida de una clave de cifrado debe considerarse con la misma seriedad que la pérdida de los datos que suele cifrar.
- Los datos confidenciales deben cifrarse cuando se guardan de manera permanente. El personal de la empresa que posee los datos, no el DBA, debe determinar cuáles datos se consideran confidenciales. No obstante, en general, cualquier dato personal (como números de Seguro Social y fechas de nacimiento) que sirvan para robar identidades deben considerarse confidenciales.

- Todos los datos que no son de conocimiento del público deben cifrarse cuando se transporten de manera electrónica a través de conexiones de red que no estén cifradas. Por ejemplo, si una empresa envía un archivo de pedido de compra a un socio comercial mediante FTP, el archivo debe cifrarse. Nada garantiza que personas malintencionadas no vigilen las redes públicas.
- El correo electrónico no se considera seguro, de modo que cualquier información delicada que se va enviar por correo electrónico debe estar en un archivo cifrado de datos adjuntos y no en el cuerpo principal del mensaje. Como opción, algunos sistemas de correo electrónico permiten mensajes cifrados y firmados.

Otras consideraciones sobre el cliente

Los clientes de una base de datos requieren un escrutinio especial de las precauciones de seguridad, porque si quedan en riesgo, ofrecen una ruta fácil para que un intruso acceda a los datos de la base de datos. Éstas son algunas consideraciones adicionales sobre los clientes:

Nivel de seguridad del navegador Web Los navegadores Web modernos permiten establecer un nivel de seguridad. Para Microsoft Internet Explorer, las especificaciones de seguridad son controladas mediante la ficha Seguridad, en el panel Opciones de Internet, al que se accede mediante la opción Herramientas, en la barra de herramientas principal. El nivel de seguridad debe fijarse lo más alto posible, pero cuidando que todavía permita el uso normal de las aplicaciones de la base de datos. Dos consideraciones se relacionan con el navegador Web:

Las *cookies* permiten que el navegador Web guarde información textual en el cliente, la que más adelante puede ser recuperada por el navegador y enviada al servidor que la solicitó. Las cookies no son muy seguras y se pueden usar para espiar a los usuarios del sistema cliente. Además, nada garantiza que personas y software no autorizados no tengan acceso a la información que se encuentra en las cookies. La política de seguridad de una organización debe atender este problema y establecer un estándar claro para el uso de cookies, que es una de las opciones controladas por el nivel de seguridad del navegador Web. Asimismo, no es inteligente diseñar sistemas de aplicaciones que requieran cookies, porque no son permitidas por todos los navegadores Web ni por todos los usuarios. En Microsoft Internet Explorer, las opciones relacionadas con cookies son controladas mediante la ficha Privacidad, en el panel Opciones de Internet.

Los *lenguajes para generar secuencias de comandos,* como VBScript, JavaScript y JScript, proporcionan características atractivas para apoyar la interacción del usuario con una página Web. Sin embargo, pueden ser utilizadas (y se han utilizado) para inyectar código malintencionado en los sistemas, de modo que debe tener cuidado de permitir que esos lenguajes se utilicen en el cliente. En especial, VBScript ha llamado la atención por el uso erróneo que se le ha dado para transportar virus en los archivos de datos adjuntos de correo electrónico.

Uso mínimo de software adicional No se debe instalar software que no sea necesario para el funcionamiento normal del cliente. Las políticas de seguridad deben prohibir que los empleados instalen software no autorizado.

Rastreo de virus Todos los sistemas computacionales que ejecutan sistemas operativos susceptibles a los virus deben tener instalado software de rastreo de virus adecuado. Los rastreadores de virus que actualizan su perfil de virus de manera automática y regular ofrecen la protección más eficaz.

Exposiciones a aplicaciones de prueba Es necesario probar con minuciosidad las aplicaciones Web mediante un cliente configurado del mismo modo en que se configurarán las estaciones de trabajo clientes de los usuarios de negocios reales. Debe aplicar los siguientes trucos de hacker para verificar que no ocurran exposiciones:

- **Inyección de SQL** Se introducen comandos de SQL en los campos de datos normales de las páginas Web, del modo en que el servidor de aplicaciones o el servidor Web los entregará a la base de datos para procesamiento. Los programas de aplicaciones deben incluir precauciones contra esos ataques, como emplear procedimientos guardados para todas las actualizaciones o probar que se rechace lo escrito en los campos que contengan caracteres de control como punto y coma, signo de unión (&) y diagonal inversa, que se utilizan para formar las secuencias de escape necesarias para una inyección de SQL.
- **Falsificación de URL** Alguien escribe en forma manual sobre el URL en el navegador Web, de modo que se revelan datos no autorizados. Son muy susceptibles a este método los diseños en que las identificaciones de sesión se asignan en forma secuencial por el servidor de aplicaciones y después son devueltas al navegador Web como un argumento en el URL. Si puede adivinar la identificación de sesión de otro usuario, encontrará la sesión de ese usuario con sólo escribir sobre su identificación de sesión en el URL.
- **Desbordamientos de búfer** Deben probarse con minuciosidad las exposiciones publicadas, como los desbordamientos de búfer, una vez instalado el parche de un vendedor, para comprobar que realmente se corrigió el problema. Un *desbordamiento de búfer* es una condición en que un proceso intenta guardar datos que rebasan los límites de un búfer de longitud fija. El resultado es que los datos adicionales se escriben en lugares adyacentes de la memoria. Los datos excedentes pueden incluir código malintencionado que luego sirve para poner en riesgo los controles de seguridad.

Seguridad del acceso a una base de datos

Una vez que confía en que los clientes, los servidores y la red están seguros, debe concentrarse en el acceso a una base de datos. La meta aquí es determinar con precisión los datos que necesita cada usuario de la base de datos para realizar negocios, y qué se permite hacer al usuario con esos datos (es decir, seleccionar, insertar, actualizar o eliminar). Cada usuario

de una base de datos debe recibir *exactamente* los privilegios que requiere; nada más y nada menos. Recuerde que un programa de aplicaciones con acceso a la base de datos es un usuario de la base de datos, igual que un empleado que la consulta directamente. En términos de seguridad de una base de datos, todos los usuarios deben ser tratados igual (es decir, se deben aplicar las mismas normas a todos), ya sea que el usuario de la base de datos sea software o "personaware". En esta sección, se analizan las opciones y los desafíos relacionados con el aseguramiento del acceso a la base de datos y a los datos.

Arquitecturas de seguridad de una base de datos

Para los DBA que dan soporte a bases de datos de varios vendedores, uno de los desafíos es que, con excepción de Microsoft SQL Server y Sybase Adaptive Server Enterprise (ASE), no existen dos bases de datos que tengan la misma arquitectura, para su seguridad. Y, por supuesto, éste es un efecto secundario de que en general las arquitecturas para bases de datos sean distintas. La única razón de que Microsoft SQL Server y Sybase ASE tengan arquitecturas similares es que la primera derivó de la segunda.

Debido a que Microsoft SQL Server/Sybase ASE y Oracle están entre las bases de datos más populares, a continuación se presenta un vistazo rápido de la manera en que cada una implementa la seguridad.

Seguridad de una base de datos en Microsoft SQL Server y Sybase ASE

Con Microsoft SQL Server y Sybase ASE, una vez que el software de DBMS se instala en el servidor, se crea un servidor de bases de datos. Por supuesto, "servidor" es un término confuso, porque se suele llamar "servidor" al hardware. En este caso, el término *servidor SQL* es una copia del software de DBMS que se ejecuta en la memoria como un grupo de procesos (por lo general, instalados como *servicios* en los entornos Windows) con información de control relacionada que se guarda en una base de datos especial en el *servidor de la base de datos*. Se usará el término *servidor SQL* para designar el software de DBMS, y el término *servidor de la base de datos* para referirse a la plataforma en que se ejecuta la base de datos. En esta arquitectura, cada servidor SQL administra varias bases de datos, y cada una representa un agrupamiento lógico de datos, determinado por el diseñador de la base de datos. En la figura 10-1 se muestra una vista simplificada de la arquitectura de seguridad para Microsoft SQL Server y Sybase ASE.

Inicio de sesión A un inicio de sesión también se le denomina *inicio de sesión de usuario*, y es una cuenta de usuario en el servidor SQL. No es igual que cualquier cuenta del sistema operativo que pueda tener el usuario en el servidor de la base de datos. Sin embargo, en los servidores de base de datos que ejecutan Microsoft Windows, el inicio de sesión puede emplear una autentificación de Windows, lo que significa que el sistema operativo Windows guarda las credenciales (el nombre de inicio de sesión y la contraseña) y autentifica a los usuarios cuando intentan conectarse al servidor SQL. Una ventaja obvia de la autentificación de Windows es que el acceso de un usuario a los diversos servidores de SQL en la empresa puede ser administrado

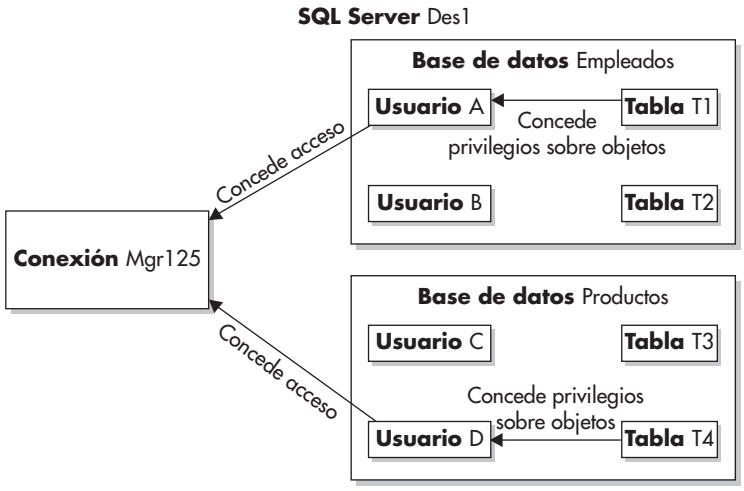


Figura 10-1 La seguridad en Microsoft SQL Server y Sybase ASE.

de manera central mediante la cuenta de Windows, en lugar de administrarse de manera local en cada servidor SQL. Tome en cuenta que, una vez definido un inicio de sesión en el servidor SQL, el usuario de la base de datos puede conectarse al servidor, pero un inicio de sesión por sí solo no proporciona al usuario acceso a información de la base de datos. Sin embargo, existe un cuenta de inicio de sesión maestra llamada *sa* (administrador del sistema, System Administrator) que, de manera similar a root en Unix y al Administrador en Microsoft Windows, tiene privilegios completos para todo lo que pasa en el entorno de SQL Server. En la figura 10-1 se presenta sólo un inicio de sesión de usuario, llamada Mgr125.

Base de datos Una base de datos es un conjunto lógico de objetos de base de datos (tablas, vistas, índices, etc.) definidos por el diseñador de una base de datos. En la figura 10-1 se muestran dos bases de datos: Empleados y Productos. Debe comprender que se permite que una conexión sólo se conecte con una base de datos después de que el administrador le ha concedido este privilegio. (Consulte el tema "Usuario", que se presenta a continuación.) Además de las bases de datos que contienen los datos del sistema, se crean algunas especiales cuando se crea el servidor SQL (que no se muestra en la figura 10-1) y son utilizadas por el DBMS para administrar el servidor SQL. A continuación se presentan algunas:

- **master (maestra)** La base de datos maestra contiene información en el nivel del sistema, las especificaciones para inicialización y configuración, las cuentas de inicio de sesión, la lista de las bases de datos configuradas en el servidor SQL y la ubicación de los archivos de datos de la base de datos principal.
- **tempdb** La base de datos tempdb contiene las tablas temporales y los procedimientos almacenados temporales.

- **model** La base de datos model contiene una plantilla para todas las otras bases de datos creadas en el sistema.
- **msdb** Sólo en bases de datos de Microsoft SQL Server, msdb contiene información que sirve para programar trabajos y alertas.

Usuarios Cada base de datos tiene un grupo de usuarios asignados. Cada usuario de base de datos se asigna a un inicio de sesión, de modo que cada usuario es una "pseudocuenta", que es un alias de una cuenta de inicio de sesión de SQL Server. No es necesario que las cuentas de usuarios tengan el mismo nombre de usuario que sus cuentas de inicio de sesión correspondientes. Cuando un administrador concede el acceso a una base de datos a una cuenta de inicio de sesión específica, el DBMS crea la cuenta de usuario correspondiente a la de inicio de sesión. En la figura 10-1, el inicio de sesión Mgr125 corresponde a Usuario A, en la base de datos Empleados, y a Usuario D en la base de datos Productos. Estos privilegios permiten un inicio de sesión en las bases de datos, pero no conceden al usuario ningún privilegio sobre los objetos de esas bases de datos. La manera en que ocurre esto se cubrirá en el tema siguiente.

Privilegios Es posible conceder a cada cuenta de usuario en una base de datos cualquier cantidad de privilegios (también llamados *permisos*). Los *privilegios del sistema* son privilegios generales aplicados en el nivel de la base de datos. Microsoft SQL Server los divide en *privilegios de servidor*, que incluye permisos de encendido, apagado y creación de copias de seguridad del servidor SQL, y *privilegios de instrucciones*, que incluyen permisos para crear una base de datos y una tabla. Los *privilegios de objetos* permiten acciones específicas sobre un objeto determinado, como selecciones y actualizaciones sobre la tabla T1. La figura 10-1 contiene flechas que muestran la concesión de privilegios de objetos sobre Tabla T1 a Usuario A en la base de datos Empleados y sobre Tabla T4 a Usuario D en la base de datos Productos. Estos privilegios funcionan de manera muy similar en todas las bases de datos relacionales, gracias a las normas ANSI, y por lo tanto se cubren en las secciones "Privilegios de sistema" y "Privilegios de objetos", más adelante en este capítulo.

Seguridad de una base de datos en Oracle

La arquitectura de seguridad de Oracle, presentada en la figura 10-2, es notablemente diferente en comparación a la de Microsoft SQL Server y Sybase ASE. Las diferencias entre ambas se destacan conforme se presenta cada componente:

Instancia Se trata de una copia del software de DBMS de Oracle que se ejecuta en la memoria. Cada instancia administra sólo *una* base de datos.

Base de datos Éste es el conjunto de archivos administrados por una sola instancia de Oracle. En conjunto, la instancia y la base de datos de Oracle forman lo que Microsoft SQL Server y Sybase ASE llaman el *servidor SQL*. La figura 10-2 representa la base de datos Des1.

Usuario Cada cuenta de una base de datos se denomina *usuario*. Igual que con Microsoft SQL Server y Sybase ASE, la cuenta de usuario puede ser autentificada de manera externa (es

decir, por el sistema operativo) o interna (por el DBMS). A cada usuario se le asigna automáticamente un *esquema* (definición siguiente), si este usuario es el *propietario* de ese esquema, lo que significa que el usuario automáticamente tiene privilegios completos sobre cualquier objeto del esquema. De manera automática se crean los usuarios predefinidos siguientes cuando se genera la base de datos (no mostrada en la figura 10-2):

- El usuario SYS es el propietario de la instancia de Oracle y contiene los objetos que utiliza Oracle para administrar la instancia. Este usuario es equivalente al usuario sa en Microsoft SQL Server y Sybase ASE.
- El usuario SYSTEM es el propietario de la base de datos de Oracle y contiene objetos que emplea Oracle para administrar la base de datos. Este esquema de usuario es similar a la base de datos master en Microsoft SQL Server y Sybase ASE.
- Muchas opciones de base de datos de Oracle crean sus propias cuentas de usuario cuando se instalan.

Esquema El esquema es el conjunto de objetos de una base de datos que pertenecen a un usuario específico. El esquema de Oracle equivale a lo que Microsoft SQL Server y Sybase ASE llaman *base de datos*. En la figura 10-2 se exponen los esquemas Empleados, Productos y Mgr125, que son propiedad de los usuarios Empleados, Productos y Mgr125, respectivamente. Los nombres de esquema y de usuario son *siempre* idénticos en Oracle. Mgr125 es una solución temporal para un desafío especial propio de la arquitectura de seguridad de Oracle, que se analiza en la siguiente sección, "Cuentas de propietario de esquema".

Privilegios Al igual que en Microsoft SQL Server y Sybase ASE, existen privilegios de sistema y de objetos. Éstos se cubren en su sección correspondiente, un poco más adelante.

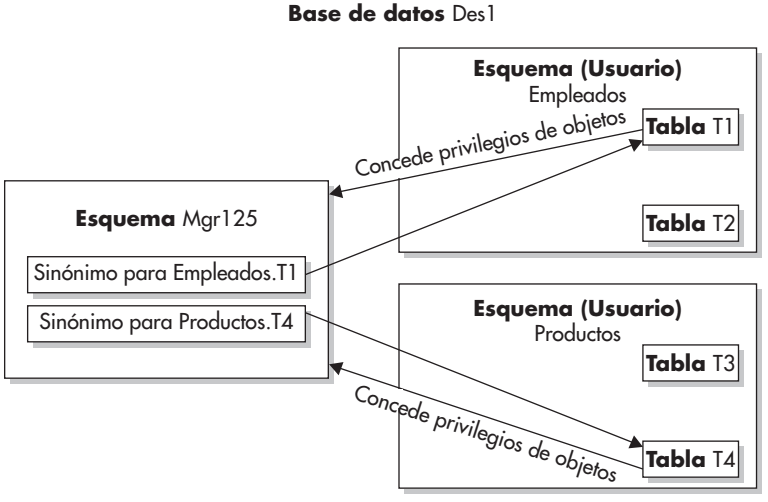


Figura 10-2 Seguridad de base de datos en Oracle.

Cuentas de propietario de esquema

En todas las bases de datos, debe tratar de no otorgar a los usuarios más privilegios de los que necesitan para hacer su trabajo. Esto no sólo evita que los errores cometidos por las personas (entre ellos los incluidos en los programas de aplicaciones y las consultas de base de datos que redactan) se conviertan en desastres de datos, sino también refuerza la honestidad de estas personas.

En Microsoft SQL Server y Sybase ASE, no debe permitirse a los usuarios de una base de datos conectarse como usuario sa. Debe crear inicios de sesión en la base de datos que tengan los privilegios mínimos requeridos. Es lamentable que esto no se aplique a menudo, y que las aplicaciones se conecten como sa o a una base de datos con una cuenta de usuario que tenga la denominación DBO (propietario de la base de datos, DataBase Owner) o DBA (administrador de la base de datos, DataBase Administrator). Las *denominaciones* son conjuntos de privilegios y se analizan en una sección venidera. Ya sea que ocurra debido a una falta de comprensión o por pereza, esta práctica representa un *enorme* riesgo de seguridad que debe prohibirse mediante políticas.

En la figura 10-2, observe que el usuario Mgr125 no posee tablas, pero los usuarios Empleados y Productos le concedieron ciertos privilegios. Esto es para solucionar temporalmente un desafío fundamental con la arquitectura de seguridad de Oracle. Si se permite a un usuario de la base de datos conectarse a ella mediante un usuario como Empleados o Productos, el usuario automáticamente tendría privilegios completos hacia cada objeto del esquema, entre ellos inserción, eliminación y actualización de cualquier tabla, y también podría crear y alterar tablas sin restricciones. En lo fundamental, esto crearía el mismo problema que cuando se permite la utilización del usuario sa o las denominaciones DBO y DBA en Microsoft SQL Server y Sybase ASE. El usuario Mgr125 imita el comportamiento del inicio de sesión con el mismo nombre, tal como se aprecia en la figura 10-1. Con los privilegios correctos del sistema, se evita que el usuario Mgr125 pueda crear en Oracle tablas propias.

Es probable que haya observado los sinónimos para Mgr125 en la figura 10-2. Un *sinónimo* es simplemente un alias o sobrenombre asignado un objeto de la base de datos. Los sinónimos son convenientes para los usuarios, de modo que los nombres no tengan que ser calificados con su nombre de esquema. Para seleccionar directamente de las tablas T1 en el esquema Empleados, el usuario Mgr125 tendría que hacer referencia al nombre de la tabla como Empleados.T1 en la instrucción de SQL. Esto no sólo es inconveniente, sino que puede provocar problemas aparentemente interminables, si alguna vez alguien decide cambiar el nombre del usuario Empleados. Al crear un sinónimo llamado T1 en el esquema Mgr125 que apunta a Empleados.T1, ahora el usuario puede hacer referencia a la tabla tan sólo como T1. Por cierto, recuerde que todos los nombres de usuarios y de objetos en Oracle distinguen entre mayúsculas y minúsculas, de modo que esos tipos de letras se combinan aquí sólo como ejemplo. La sintaxis para crear este sinónimo es la siguiente:

```
CREATE SYNONIM T1 FOR EMPLEADOS.T1;
```

Privilegios de sistema

Los privilegios de sistema son permisos generales para efectuar funciones de administración del servidor y las bases de datos. Cada vendedor de una base de datos permite cientos de permisos y casi todos son privilegios de sistema. Al igual que con los privilegios de objetos, los de sistema se conceden mediante la instrucción GRANT y se rescinden mediante la instrucción REVOKE de SQL. En las secciones siguientes se presentan algunos de los privilegios que se emplean con mayor frecuencia. Encontrará los detalles completos en la documentación proporcionada por su vendedor.

Ejemplos de privilegios de sistema de Microsoft SQL Server (de servidor y de instrucciones)

Éstos son algunos privilegios de sistema que suelen emplearse en Microsoft SQL Server:

- **SHUT DOWN** Proporciona la facultad de usar un comando para apagar el servidor.
- **CREATE DATABASE** Proporciona la capacidad de crear nuevas bases de datos en el servidor SQL.
- **BACKUP DATABASE** Brinda la facultad de ejecutar copias de seguridad de las bases de datos que están en el servidor SQL.

Ejemplos de privilegios de sistema en Oracle

Éstos son algunos privilegios de sistema que se emplean a menudo en Oracle:

- **CREATE SESSION** Proporciona la facultad de conectarse a la base de datos.
- **CREATE TABLE** Proporciona la capacidad de crear tablas en su propio esquema. Existen privilegios similares para otros tipos de objetos, como índices, sinónimos, procedimientos, etc.
- **CREATE ANY TABLE** Brinda la facultad de crear tablas en el esquema de *cualquier* usuario. Existen privilegios similares para otros tipos de objetos, como índices, sinónimos, procedimientos, etcétera.
- **CREATE USER** Proporciona la facultad de crear usuarios nuevos en la base de datos.

Privilegios de objetos

Los privilegios de objetos se conceden a los usuarios con la instrucción GRANT y se revocan con la instrucción REVOKE de SQL. Al usuario (el inicio de sesión) que recibe los privilegios se le denomina *concesionario*. Estas instrucciones también se analizaron en el capítulo 4. La instrucción GRANT puede incluir una cláusula WITH GRANT OPTION que permite al destinatario conceder el privilegio a otras personas. Si después se revoca el privilegio, ocurre una revocación en cascada en caso de que este usuario, a su vez, haya concedido permiso a

otros. *No* es recomendable el uso de la cláusula WITH GRANT OPTION porque es muy fácil perder el control acerca de quién tiene cuáles privilegios.

Ésta es la sintaxis general de las instrucciones **GRANT** y **REVOKE**, junto con algunos ejemplos:

```
GRANT <lista de privilegios> ON <objeto> TO <lista de personas a las que
se otorga>
   (WITH GRANT OPTION);
GRANT SELECT, UPDATE, INSERT ON T1 TO Mgr125;
GRANT SELECT ON T2 TO Usuario1, Usuario2, Usuario3;

REVOKE <lista de privilegios> ON <objeto> FROM <lista de personas a las
que se otorga>
 (WITH REVOKE OPTION);
REVOKE SELECT, UPDATE, INSERT ON T1 FROM Mgr125;
REVOKE SELECT ON T2 FROM Usuario1, Usuario2, Usuario3;
```

Funciones

Una *función* (o rol) es un conjunto designado de privilegios que, a su vez, puede concederse a uno o más usuarios. Casi todos los sistemas RDBMS tienen funciones predefinidas incluidas en el sistema, y los usuarios de la base de datos con un privilegio CREATE ROLE pueden crear privilegios propios. Las funciones tienen las ventajas siguientes:

- *Las funciones pueden existir antes que las cuentas de usuarios*. Por ejemplo, puede crear una función que contenga todos los privilegios requeridos para trabajar en un entorno de desarrollo específico. Cuando una nueva contratación se integra al equipo del proyecto, una instrucción GRANT proporciona a la nueva cuenta de usuario todos los permisos requeridos.
- *Las funciones alivian al administrador de mucho tedio*. Se pueden conceder (o revocar) muchos privilegios con un solo comando, cuando se emplea una función.
- *Las funciones sobreviven cuando se descartan las cuentas de usuario*. Si el DBA debe descartar o volver a crear una cuenta de usuario, puede requerir mucho trabajo volver a instalar todos los privilegios, lo que se simplifica si todos los privilegios se integran en una función.

Para los administradores, una función común es DBA, que transfiere muchos privilegios poderosos (en Oracle, más de 125 distintos). Es obvio que un conjunto de privilegios tan poderoso debe concederse después de una cuidadosa reflexión.

Vistas

Uno de los problemas de seguridad comunes que se deben atender es cómo permitir a los usuarios de una base de datos el acceso a algunas y filas columnas de una tabla y al mismo tiempo impedir el acceso a otras filas y columnas. Las vistas son un medio excelente para

Pregunta al experto

P: **No he visto ninguna instrucción SQL para crear cuentas nuevas. ¿Existen?**

R: Sí y no. Unas implementaciones de SQL, como las de Oracle, incluyen una instrucción CREATE USER. Sin embargo, otras, como las de Microsoft SQL Server y Sybase ASE, no lo hacen; en cambio, dependen de procedimientos almacenados que proporciona el vendedor y de herramientas GUI para la creación de cuentas de usuario. La norma AMSI/ISO de SQL no ofrece una sintaxis estándar para crear cuentas de usuarios, de modo que cada vendedor está en libertad de implementar la función como le plazca.

conseguir esto. Éstos son algunos beneficios de emplear vistas para alcanzar los objetivos de seguridad:

- *Es posible omitir de la vista las columnas que no necesita el usuario de una base de datos*. Si se supone que se concede al usuario acceso a la vista en lugar de a las tablas relacionadas, este método evita por completo que el usuario vea la información de las columnas omitidas en la vista.
- *En una vista, se puede incluir una cláusula WHERE para limitar las filas devueltas*. Se pueden incluir combinaciones que coincidan con otras tablas como un modo de limitar las filas. Por ejemplo, la vista puede limitar las filas de la tabla Productos a los productos cuya ID de división coincida con la división en que trabaja el empleado.
- *Se pueden emplear combinaciones de tablas de búsqueda con el fin de reemplazar valores codificados en una tabla con sus descripciones correspondientes*. Una tabla de búsqueda suele contener una lista de valores codificados (por ejemplo, códigos de departamentos, de transacciones y de estado) y sus descripciones, y sirve para "buscar" las descripciones de los códigos. Aunque éste es un tema menor, si un empleado intenta irrumpir en los registros de una base de datos durante un intento de fraude, le costará mucho trabajo si no puede ver los códigos utilizados para clasificar las transacciones. Además, a los empleados que intentan cumplir con su trabajo les resulta más fácil leer y comprender las descripciones de un código que los valores codificados correspondientes.

Monitoreo y auditoría de la seguridad

Las políticas y los controles de seguridad suelen ser insuficientes. También debe instalarse un sistema de monitoreo para detectar huecos en la seguridad y aplicar medidas correctivas. Existen en el mercado numerosas herramientas para detectar brechas de seguridad y son capaces de monitorear un servidor y detectar los cambios no autorizados a los archivos guardados

en el sistema. Asimismo, todos los productos RDBMS importantes tienen contemplado configurar la auditoría para que, sin que nadie lo perciba, se registren acciones específicas en la base de datos, y para que esos registros se incluyan en tablas de auditoría con las que después se generan informes. Consulte en la documentación de su RDBMS una descripción completa de estas características de auditoría.

También es buena idea hacer que un auditor independiente revise las políticas y procedimientos de seguridad de su organización cuando que se redacten y, más adelante, a intervalos periódicos. Además, es una medida inteligente hacer que sus auditores o un consultor que se especialice en seguridad de los sistemas de información, efectúe una auditoría en el lugar, incluida una prueba en busca de vulnerabilidades en el sitio que no hayan sido atendidas. Las intrusiones en el sistema, entre las que se incluye el fraude, cuestan mucho más que una auditoría del sistema, y ésta evita cualquier situación embarazosa ante los empleados y los clientes.

Pruebe esto 10-1 Privilegios de los objetos de una base de datos

En este ejercicio probará las instrucciones de SQL que conceden y revocan privilegios en la base de datos, entre ellos algunas pruebas para comprobar que los privilegios se conceden de manera correcta.

Paso a paso

1. Se necesitan dos cuentas de usuarios para este ejercicio: una que será la propietaria del objeto de base de datos y otra a la que se concederán privilegios en ese objeto de base de datos. Cree una cuenta llamada Datos1 y otra llamada Usuario1. Emplee autentificación por parte de la base de datos en lugar de la del sistema operativo. Si trabaja en SQL Server o Sybase ASE, también necesitará crear una base de datos, hacer que la cuenta Datos1 sea la propietaria de la base de datos, conceder a la cuenta Usuario1 acceso a la base de datos, y hacer que la nueva base de datos sea la predeterminada para ambas cuentas, cuando se conecten. Cada producto de RDBMS permite la creación de una cuenta de usuario de manera única, de modo que si no está familiarizado con esta función en su RDBMS, consulte su documentación.
2. Conceda a las cuentas Datos1 y Usuario1 cualquier privilegio de sistema requerido para conectarse a la base de datos y crear objetos de ella. En Oracle, debe conceder a las cuentas las funciones CONNECT y RESOURCE. En SQL Server y Sybase ASE, las acciones que efectuó en el paso 1 deben ser suficientes.
3. Conéctese a la base de datos mediante la cuenta Usuario1.

(*continúa*)

4. Los objetos deben existir antes de que se les concedan privilegios. Se utilizará una tabla Departamento sencilla que contenga los códigos y los nombres de los departamentos. Cree la tabla DEPARTMENTO al ejecutar la instrucción de SQL siguiente. En SQL Server y Sybase ASE, compruebe que se haya generado en la base de datos la tabla definida en el paso 1 de este ejercicio.

```
CREATE TABLE DEPARTAMENTOO
   (CÓDIGO_DEPARTAMENTO   CHAR(3),
   NOMBRE_DEPARTAMENTO    VARCHAR(50));
```

5. Utilice la instrucción de SQL siguiente para conceder los privilegios SELECT e INSERT en la tabla DEPARTMENTO a Usuario1:

```
GRANT SELECT, INSERT ON DEPARTAMENTO TO USUARIO1;
```

6. Conéctese a la base de datos como Usuario1.

7. Aplique la instrucción siguiente con el fin de insertar una fila para el departamento 001 en la tabla:

```
INSERT INTO DATOS1.DEPARTAMENTO
VALUES ('001','Ejecutivo');
```

NOTA

En este paso y del 8 al 10, para SQL Server y Sybase ASE, el nombre de la tabla no debe calificarse con DATOS1.

8. Recupere la fila que acaba de insertar mediante la instrucción siguiente:

```
SELECT * FROM DATOS1.DEPARTAMENTO
 WHERE CÓDIGO_DEPARTAMENTO = '001';
```

9. Intente eliminar la fila que acaba de insertar con la instrucción siguiente. La eliminación debe fallar porque una cuenta Usuario1 no tiene privilegios de eliminación sobre el objeto.

```
DELETE FROM DATOS1.DEPARTAMENTO
 WHERE CÓDIGO_DEPARTAMENTO = '001';
```

10. Trate de descartar la tabla mediante la instrucción siguiente. La instrucción debe fallar porque Usuario1 no tiene privilegios para descartar el objeto.

```
DROP TABLE DATOS1.DEPARTAMENTO;
```

11. Conéctese con la cuenta Datos1 (la cuenta propietaria de la tabla DEPARTAMENTO).

12. Descarte la tabla con la instrucción siguiente (observe que esta vez no tiene que estar calificada con DATOS1 porque está conectado como esa cuenta):

```
DROP TABLE DEPARTAMENTO;
```

13. Para finalizar la tarea de limpieza, descarte las cuentas de usuario Datos1 y Usuario1, y cualquier base de datos que haya creado para este ejercicio.

Resumen de Pruebe esto

En este ejercicio Pruebe esto, creó dos cuentas de usuario. Después generó una tabla en una de ellas y le concedió ciertos privilegios sobre la otra cuenta. A continuación, probó diversas instrucciones SQL sobre la tabla para demostrar que la falta de privilegios adecuados impidió que funcionaran algunas de las instrucciones. Por último, eliminó la tabla y las cuentas de usuarios que creó para dejar los datos como estaban cuando comenzó.

✓ Autoexamen Capítulo 10

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. A un conjunto de privilegios que se conceden a varios usuarios se le denomina _____.

2. Los privilegios se rescinden mediante el comando _____ de SQL.

3. Para los servidores de una base de datos conectados a una red, la aplicación exclusiva de seguridad física es _____.

4. Los empleados que se conectan a una red empresarial desde su hogar u otro lugar remoto deben tener _____ entre la computadora y su módem de cable o DSL.

5. Cuando las credenciales de inicio de sesión se guardan en la computadora, siempre deben _____.

6. La seguridad de red

- **A** Sólo puede manejarse mediante enrutadores.
- **B** Sólo puede manejarse mediante una firewall.
- **C** Debe incluir consideraciones para los empleados que trabajan en otro lugar.
- **D** Debe ser obligatoria para todos los sistemas computacionales conectados a cualquier red.

7. La protección con firewall puede incluir

- **A** Filtrado de paquetes.
- **B** Selección de paquetes mediante una tabla de enrutamiento.
- **C** Traducción de dirección de red.
- **D** Delimitación de los puertos que pueden emplearse para acceso.
- **E** Falsificación de IP.

8. Las redes inalámbricas deben asegurarse porque

- **A** Existen en el mercado puntos de acceso inalámbricos económicos.
- **B** Cualquier persona con un adaptador de red inalámbrica se puede conectar a una red no protegida.
- **C** Los empleados pueden utilizar la red inalámbrica para comunicarse en secreto con hackers.
- **D** Las ondas de radio penetran las paredes de las oficinas vecinas.
- **E** Las ondas de radio pueden llegar a la vía pública fuera de un edificio.

9. Algunos componentes de la seguridad de un punto de acceso inalámbrico son

- **A** Traducción de dirección de red.
- **B** La política de seguridad de una organización.
- **C** Cifrado.
- **D** Redes privadas virtuales.
- **E** Listas de direcciones MAC.

10. Entre las precauciones para la seguridad a nivel del sistema están

- **A** La instalación de un mínimo de componentes de software necesarios.
- **B** Conceder sólo los privilegios sobre tablas necesarias para los usuarios.
- **C** Aplicar los parches de seguridad de manera oportuna.
- **D** Modificar todas las contraseñas predeterminadas.
- **E** Emplear contraseñas sencillas y fáciles de recordar.

11. El cifrado

- **A** Debe utilizarse para todos los datos confidenciales.
- **B** Siempre deben usar claves de cuando menos 28 bits de longitud.
- **C** Debe aplicarse a los datos confidenciales enviados en una red.
- **D** Puede emplear claves simétricas o asimétricas.
- **E** Nunca debe usarse para credenciales de inicio de sesión.

12. ¿Cuáles son consideraciones de seguridad del cliente?

- **A** Listas de direcciones MAC.
- **B** Nivel de seguridad del navegador Web.

C Conceder sólo los privilegios sobre las tablas de una base de datos que sean absolutamente necesarios.

D Utilizar un rastreador de virus.

E Probar los riesgos de una aplicación.

13. En Microsoft SQL Server, un inicio de sesión de usuario

A Se puede conectar a cualquier cantidad de bases de datos.

B Automáticamente tiene privilegios de acceso a una base de datos.

C Puede utilizar autentificación de Windows.

D Se puede autentificar mediante Microsoft SQL Server.

E Posee un esquema de base de datos.

14. En Microsoft SQL Server, una base de datos

A Se vuelve una propiedad mediante un inicio de sesión.

B Puede tener uno o más usuarios asignados a ella.

C Puede contener datos del sistema (por ejemplo, datos maestros) o datos de usuarios (aplicaciones).

D Puede tener privilegios concedidos.

E Es un conjunto lógico de objetos de base de datos.

15. En Oracle, una cuenta de usuario

A Se puede conectar a cualquier cantidad de bases de datos.

B Automáticamente tiene privilegios de base de datos.

C Puede emplear autentificación por parte del sistema operativo.

D Se puede autentificar mediante el DBMS de Oracle.

E Es propietaria de un esquema de base de datos.

16. En Oracle, una base de datos

A Es propiedad de un usuario.

B Puede tener una o más cuentas de usuarios definidas en ella.

C Puede contener datos del sistema (por ejemplo, un esquema del sistema) y datos de usuarios (aplicaciones).

D Es igual que un esquema.

E Es administrada por una instancia de Oracle.

17. Los privilegios de sistema

- **A** Son concedidos de manera similar en Oracle, Sybase ASE y Microsoft SQL Server.
- **B** Son específicos para un objeto de una base de datos.
- **C** Permiten al concesionario efectuar ciertas funciones administrativas en el servidor, como apagarlo.
- **D** Son rescindidos mediante la instrucción REMOVE de SQL.
- **E** Varían entre las bases de datos de diferentes vendedores.

18. Los privilegios de objetos

- **A** Son concedidos de manera similar en Oracle, Sybase ASE y Microsoft SQL Server.
- **B** Son específicos para un objeto de una base de datos.
- **C** Permiten al concesionario efectuar ciertas funciones administrativas en el servidor, como apagarlo.
- **D** Son rescindidos mediante la instrucción REMOVE de SQL.
- **E** Son concedidos mediante la instrucción GRANT de SQL.

19. El uso de **WITH GRANT OPTION** cuando se conceden privilegios de objetos

- **A** Permite al concesionario conceder privilegios a otras personas.
- **B** Otorga al concesionario privilegios de DBA sobre toda la base de datos.
- **C** Puede provocar problemas de seguridad.
- **D** Generará una eliminación en cascada, si después se revocan los privilegios.
- **E** Es una práctica muy recomendada porque es muy conveniente utilizarla.

20. Las vistas ayudan a la implementación de una política de seguridad al

- **A** Limitar las columnas de una tabla a las que tiene acceso un usuario.
- **B** Limitar las bases de datos a las que tiene acceso un usuario.
- **C** Limitar las filas de una tabla a las que tiene acceso un usuario.
- **D** Almacenar los resultados de una auditoría de la base de datos.
- **E** Vigilar las intrusiones en una base de datos.

Capítulo 11

Implementación de las bases de datos

Habilidades y conceptos clave

- Procesamiento mediante un cursor
- Administración de transacciones
- Afinación del rendimiento
- Control de cambios

En este capítulo se trata el desarrollo de aplicaciones que emplean el sistema de base de datos. Entre ellas están el procesamiento mediante un cursor, la administración de transacciones, la afinación del desempeño y el control de cambios.

Procesamiento mediante un cursor

Antes de analizar el *procesamiento de transacciones*, que incluye una revisión de los mecanismos de bloqueo requeridos para dar soporte a actualizaciones concurrentes en la base de datos, es necesario explorar los modos en que los programas de aplicaciones manejan las consultas en la base de datos. Al conjunto de filas devueltas por la ejecución de una consulta a la base de datos se le denomina *conjunto de resultados*. Cuando selecciona datos de la base de datos, los lenguajes de programación de aplicaciones, como C y Java, presentan un dilema cuando el conjunto de resultados contiene varias filas de datos. Estos lenguajes de programación están diseñados para manejar un registro a la vez (en el caso de Java, una instancia de un objeto a la vez), de modo que ocurre una desigualdad, que debe atenderse.

Para superar la desigualdad, casi todas las bases de datos relacionales permiten el concepto de un *cursor*, que es simplemente un apuntador a una sola fila del conjunto de resultados. En Oracle, el soporte de cursores está incluido en la extensión de SQL llamada PL/SQL (lenguaje procedimental/SQL, Procedural Language/SQL), y ocurre lo mismo con Transact-SQL para Sybase ASE y Microsoft SQL Server. Los ejemplos de este capítulo emplean Oracle, de modo que algunos tal vez requieran modificaciones menores antes de que funcionen en otros productos RDBMS. El uso de un cursor se parece al de un archivo simple tradicional en que el cursor debe definirse y abrirse antes de utilizarse, puede leerse mediante una búsqueda de filas en un bucle de programación y debe cerrarse cuando el programa ya no lo necesita.

A continuación se presenta un ejemplo de una declaración de cursor. Para mayor claridad, todas las palabras clave se muestran en mayúsculas y los nombres de objetos de la base de datos en minúsculas. En Oracle, esto no representa ninguna diferencia porque los nombres de

todos los objetos de la base de datos no distinguen mayúsculas y minúsculas. Sin embargo, su experiencia puede ser diferente con otros productos RDBMS.

```
DECLARE CURSOR clientes_ny AS
   SELECT número_de_cliente, nombre, dirección, ciudad, código_postal
     FROM cliente
    WHERE estado = 'NY';
```

NOTA

Las instrucciones de manejo de un cursor presentadas en esta sección están diseñadas para usarse en los programas de aplicaciones. En general, no se pueden ejecutar mediante un cliente interactivo de SQL.

Es probable que reconozca la tabla Cliente del capítulo 8. Si ignora la primera línea, la instrucción parece una consulta de SQL común: selecciona algunas columnas de una tabla y, en este caso, tiene una cláusula WHERE que limita las filas devueltas a las que contienen el estado de Nueva York (NY). Esto es muy atractivo, porque significa que es posible probar la consulta mediante una herramienta interactiva de cliente de SQL antes de pegarla en un programa y convertirla en una declaración de cursor. La cláusula DECLARE CURSOR define el cursor, que ha sido denominado clientes_ny. Las declaraciones de cursor no son instrucciones ejecutables, lo que significa que cuando son procesadas por el RDBMS, no hacen nada excepto instalar una definición a la que después se puede hacer referencia. Se revisa la sintaxis de la declaración y otros detalles internos, pero la base de datos no necesita acceder a ninguna tabla hasta que se abre el cursor.

El cursor debe abrirse antes de utilizarse. En este ejemplo, es probable que el RDBMS no tenga que recuperar filas cuando se abra el cursor, pero, por eficiencia, puede decidir que se recupere cierta cantidad de filas y se coloquen en un búfer. Un *búfer* es sólo un área de la memoria de la computadora utilizada para contener datos de manera temporal. Es mucho más eficiente crear un búfer para que contenga cierta cantidad de filas buscadas con anticipación que acudir a los archivos de la base de datos por cada fila individual, porque las computadoras acceden a la memoria mucho más rápido que a los archivos de un sistema de archivos. Sin embargo, en algunos casos el RDBMS *debe* buscar todas las filas que coinciden con una consulta y clasificarlas antes de que pueda devolverse la primera fila al programa de aplicaciones. Es probable que haya adivinado que éstas son consultas que contienen una instrucción ORDER BY para clasificar las filas devueltas. Si no existe un índice en las columnas que se emplean para definir una secuencia, entonces el RDBMS debe encontrarlas y clasificarlas antes de saber cuál fila devolver correctamente como la *primera* (la primera que se clasifica primero en la secuencia solicitada).

Aunque ocurren muchas cosas cuando se abre un cursor, la instrucción misma es muy sencilla. Ésta es una instrucción OPEN CURSOR para este ejemplo:

```
OPEN CURSOR clientes_ny;
```

Cada vez que el programa requiere una fila nueva del conjunto de resultados, en realidad se usa un comando **FETCH** contra el cursor. Esto es muy similar a leer el siguiente registro de un archivo en un antiguo sistema de archivo simple. Recuerde que el cursor es sólo un apuntador al conjunto de resultados. Cada vez que se emite una instrucción **FETCH**, la fila a la que se apunta en ese momento es devuelta al programa solicitante (es decir, el programa que usó **FETCH**) y el cursor avanza una fila para apuntar a la siguiente fila que va a ser devuelta. Si ya no existen filas en el conjunto de resultados, se devuelve un código al programa solicitante para indicarlo. Otro detalle que maneja **FETCH** es ubicar las columnas devueltas en variables del lenguaje de programación (llamadas *variables de lenguaje de host* o simplemente *variables de host*). Esto se hace con la cláusula INTO, y naturalmente la sintaxis de los nombres de variables cambia de un lenguaje de programación a otro. En este ejemplo se emplean nombres muy sencillos para mantenerse alejado de problemas con el lenguaje de programación, pero en la vida real necesita que, en lo posible, los nombres sean muy descriptivos. También es una buena práctica de programación crear nombres que *no* sean exactamente iguales a los de columnas de la base de datos, para evitar confusión cuando alguien más lea el programa. Por esta razón, en este ejemplo los nombres de variables contienen el prefijo **v_** (para indicar que es una *variable*). Ésta es la manera de buscar el cursor clientes_ny:

```
FETCH clientes_ny
 INTO v_número_de_cliente, v_nombre, v_dirección, v_ciudad,
      v_código_postal;
```

Observe que la instrucción FETCH sólo hace referencia al nombre del cursor y a las variables de host. La declaración del cursor relaciona el cursor con la tabla y las columnas a las que se hace referencia. Como ya se mencionó, siempre debe cerrar el cursor cuando el programa ya no lo necesita, porque esto libera los recursos que ha utilizado el cursor, entre ellos memoria para búferes. La instrucción CLOSE es tan simple como la instrucción OPEN:

```
CLOSE clientes_ny;
```

El tema del procesamiento de un cursor se ha presentado antes del análisis de la administración de transacciones porque los cursores desempeñan una función fundamental en ciertos eventos de transacciones.

Administración de transacciones

Para dar soporte a los usuarios de una base de datos, el DBMS debe incluir provisiones para manejar las transacciones realizadas por los sistemas de aplicaciones que emplean la base de datos.

¿Qué es una transacción?

Una *transacción* es una serie separada de acciones que debe procesarse por completo o no procesarse en absoluto. Algunos llaman a una transacción una *unidad de trabajo*, como un

modo de destacar todavía más su naturaleza tipo todo o nada. Las transacciones tienen propiedades que se recuerdan con facilidad por medio de las siglas *ACID* (Atomicidad, Consistencia, Independencia, Durabilidad):

- **Atomicidad** Una transacción debe mantenerse completa. Es decir, debe tener éxito o fracasar por completo. Cuando tiene éxito, el sistema debe preservar todos los cambios realizados por la transacción. Si una transacción falla, deben deshacerse por completo todos los cambios efectuados. En los sistemas de base de datos, se utiliza la frase *recuperación al último estado bueno conocido* para el proceso que revierte los cambios realizados mediante una transacción fallida y el término *confirmación* para el proceso que vuelve permanentes los cambios en las transacciones.
- **Consistencia** Una transacción debe transformar la base de datos de un estado consistente a otro. Por ejemplo, una transacción que crea una factura para un pedido lo transforma de *enviado* a *facturado*, lo que incluye todos los cambios adecuados en la base de datos.
- **Independencia** Cada transacción debe realizar su trabajo independientemente de cualquier otra transacción que ocurra al mismo tiempo.
- **Durabilidad** Los cambios realizados mediante transacciones completadas deben ser permanentes, incluso después de un apagado o una falla posterior de la base de datos u otro componente fundamental del sistema. En la terminología de objetos, se aplica el término *persistencia* para los datos guardados de manera permanente. Aquí, el concepto de permanencia puede parecer confuso, porque nada jamás parece mantenerse intacto durante mucho tiempo en una base de datos para procesamiento de transacciones en línea (OLTP, Online Transaction Processing). Sólo recuerde que *permanente* significa que el cambio no desaparecerá cuando la base de datos se apague o falle; *no* significa que los datos están en un estado permanente que nunca puede volverse a modificar.

Soporte DBMS para transacciones

Aparte de los sistemas de base de datos para computadoras personales, casi todos los DBMS proporcionan soporte a transacciones. Esto incluye provisiones en SQL para identificar el inicio y el final de cada transacción, junto con una opción para registrar todos los cambios realizados mediante transacciones, de modo que se pueda efectuar una recuperación cuando sea necesario. Como puede suponer, las normas se quedaron a la zaga de la necesidad del soporte a transacciones, de modo que éste varía un poco entre vendedores de RDBMS. Como ejemplo, a continuación se revisa el soporte de transacciones en Microsoft SQL Server y Oracle, seguido por un análisis de los registros de transacciones.

Soporte a transacciones en Microsoft SQL Server

Microsoft SQL Server permite transacciones en tres modos: confirmación automática, explícito e implícito. Los tres modos están disponibles cuando se conecta directamente a la base de

datos mediante una herramienta cliente para este propósito. Sin embargo, si planea utilizar un controlador ODBC o JDBC, debe consultar en la documentación del controlador la información sobre el soporte a transacciones que ofrece. A continuación se presenta una descripción de los tres modos:

- **Modo de confirmación automática** En el modo de confirmación automática, cada instrucción SQL se consigna automáticamente cuando concluye. En esencia, esto convierte a cada instrucción en una transacción separada. Cada conexión a Microsoft SQL Server emplea la confirmación automática hasta que se inicia una transacción explícita o se establece el modo de transacción implícita. En otras palabras, la confirmación automática es el modo predeterminado para cada conexión de SQL Server.
- **Modo explícito** En el modo explícito, cada transacción se inicia con una instrucción BEGIN TRANSACTION y concluye con una COMMIT TRANSACTION (para una conclusión correcta) o una ROLLBACK TRANSACTION (para una conclusión fallida). Este modo se emplea con más frecuencia en los programas de aplicaciones, procedimientos almacenados, desencadenadores y secuencias de comandos. Ésta es la sintaxis general de las tres instrucciones de SQL:

```
BEGIN TRAN[SACTION] [nombre_tran | @variable_nombre_tran]

COMMIT [TRAN[SACTION) [nombre_tran | @variable_nombre_tran]]

ROLLBACK [TRAN[SACTION] [nombre_tran | @variable_nombre_tran |
          nombre_punto_de_interrupción | @variable_ nombre_punto_
          de_interrupción]]
```

- **Modo implícito** El modo de transacciones implícito se activa y desactiva con el comando **SET IMPLICIT_TRANSACTIONS {ON | OFF}**. Cuando el modo implícito está activo, se inicia una transacción nueva cada vez que se ejecuta una instrucción de una lista específica de SQL, en que están DELETE, INSERT y UPDATE, entre otras. Una vez que se inicia de manera implícita una transacción, continúa hasta que es consignada o recuperada. Si el usuario de la base de datos se desconecta antes de emitir una instrucción de conclusión de transacción, la transacción se recupera automáticamente.

Microsoft SQL Server conserva todas las transacciones y las modificaciones realizadas en el *registro de transacciones*. Éste conserva la imagen anterior y posterior de cada modificación hecha en la base de datos mediante una transacción. Esto facilita cualquier recuperación al último estado correcto, porque se emplea la imagen anterior para revertir los cambios realizados en la base de datos mediante una transacción. La confirmación de una transacción no está completa hasta que se ha escrito una anotación de confirmación en el registro de transacciones. Debido a que los cambios en una base de datos no siempre se escriben de inmediato en el disco, en ocasiones el registro de transacciones es el único medio de recuperación cuando ocurre una falla en el sistema.

Soporte a transacciones en Oracle

Oracle sólo permite dos modos de transacciones: confirmación automática e implícita. Igual que con Microsoft SQL Server, el soporte varía cuando se utilizan controladores ODBC y JDBC, de modo que en esos casos debe consultar la documentación del vendedor del controlador. A continuación se presentan descripciones de estos dos modos en Oracle:

- **Modo de confirmación automática** Igual que con Microsoft SQL Server, cada instrucción SQL se consigna automáticamente cuando concluye. El modo de confirmación automática se activa y desactiva mediante el comando **SET AUTOCOMMIT**, tal como se muestra aquí, y de manera predeterminada está desactivado:

```
SET AUTOCOMMIT ON
SET AUTOCOMMIT OFF
```

- **Modo implícito** Una transacción se inicia implícitamente cuando el usuario de la base de datos se conecta a ella (es decir, cuando comienza una sesión nueva en la base de datos). Éste es el modo de transacciones predeterminado en Oracle. Cuando concluye una transacción con una confirmación o una recuperación al último estado correcto, se inicia de manera automática una transacción nueva. A diferencia de Microsoft SQL Server, no se permiten las transacciones anidadas (transacciones dentro de transacciones). Una transacción termina con una confirmación cuando ocurre cualquiera de las siguientes opciones: el usuario de la base de datos emite la instrucción COMMIT de SQL, la sesión de la base de datos termina normalmente (es decir, el usuario emite un comando **EXIT** o **DISCONNECT**), o el usuario de la base de datos emite una instrucción del lenguaje de definición de datos (DDL, Data Definition Language) de SQL (una instrucción CREATE, DROP o ALTER). Una transacción finaliza con una recuperación cuando ocurre algo de lo siguiente: el usuario de la base de datos emite una instrucción ROLLBACK de SQL, o la sesión de la base de datos termina de manera anormal (esto es, la conexión del cliente se cancela o la base de datos se detiene o se apaga mediante una de las opciones de apagado que abortan las conexiones de un cliente en lugar de esperar a que concluyan).

Pruebe esto 11-1 Soporte a transacciones en SQL

En este ejercicio, explorará las instrucciones para soporte a transacciones en su RDBMS.

Paso a paso

1. Utilice la misma tabla Departamento empleada en Pruebe esto 10-1. Si ya creó una, descártela y vuelva a crearla, porque los resultados de su consulta en este ejercicio serán predecibles. Ejecute la instrucción siguiente (la instrucción DROP es innecesaria si la tabla no existe):

(*continúa*)

```
DROP TABLE DEPARTAMENTO;

CREATE TABLE DEPARTAMENTO
  (CÓDIGO_DEPARTAMENTO  CHAR(3),
  NOMBRE_DEPARTAMENTO   VARCHAR(50));
```

2. Establezca la base de datos en el modo de transacciones implícito. Para Oracle, es el modo predeterminado, siempre y cuando no haya activado el modo de confirmación automática. Consulte en la documentación de su RDBMS cómo se hace esto. Si emplea SQL Server, aplique la instrucción siguiente:

```
SET IMPLICIT_TRANSACTIONS ON
```

3. Inserte una fila en la tabla con la instrucción que aparece a continuación, pero no consigne el cambio:

```
INSERT INTO DEPARTAMENTO
VALUES ('001','Ejecutivo');
```

4. Ejecute una instrucción SELECT para confirmar que la fila existe:

```
SELECT * FROM DEPARTAMENTO;
```

5. Si sabe cómo conectarse a la base de datos por segunda vez en una sesión de cliente diferente, hágalo y ejecute en ella la consulta de selección del paso 4. No podrá ver la fila porque son datos no consignados y, por lo tanto, sólo está disponible en la sesión que la creó. Dependiendo del modo en que su DBMS maneja el bloqueo, puede parecer como si esta consulta estuviera congelada mientras el cliente SQL espera que el DBMS devuelva la fila (en particular, en SQL Server). El bloqueo se cubre en la sección siguiente de este capítulo.

6. Ejecute una instrucción ROLLBACK del modo siguiente:

```
ROLLBACK;
```

7. Vuelva a aplicar la instrucción SELECT del paso 4. Observe que ahora la fila ha desaparecido.

8. Ejecute de nuevo la instrucción INSERT del paso 3, seguida por una confirmación:

```
INSERT INTO DEPARTAMENTO
VALUES ('001','Ejecutivo');

COMMIT;
```

9. Aplique otra vez la instrucción SELECT del paso 4 para confirmar que la fila está ahí.

10. Ejecute una instrucción ROLLBACK como lo hizo en el paso 6. En SQL Server, puede recibir un error que le indique que ninguna transacción está en curso (la confirmación anterior concluyó su transacción implícita).

```
ROLLBACK;
```

11. Trate de aplicar la instrucción SELECT una vez más. Observe que la fila todavía está ahí. Una instrucción ROLLBACK no afecta los datos que ya se han consignado en la base de datos.

12. Elimine la tabla Departamento para regresar su base de datos (o esquema en Oracle) adonde comenzó. En Oracle, las instrucciones de DDL nunca son parte de las transacciones, pero lo son en SQL Server, de modo que necesitará ejecutar una instrucción COMMIT después de la instrucción DROP en SQL Server.

```
DROP TABLE DEPARTAMENTO;
COMMIT;
```

Resumen de Pruebe esto

En este ejercicio Pruebe esto, empleó el modo implícito de transacciones junto con instrucciones INSERT, SELECT, COMMIT Y ROLLBACK para comprobar el soporte a transacciones en SQL.

Bloqueo y bloqueo mutuo de transacciones

Aunque compartir datos de manera simultánea entre muchos usuarios de una base de datos aporta beneficios significativos, una desventaja seria es que puede provocar que se pierdan las actualizaciones. Por suerte, los vendedores de bases de datos han preparado soluciones para el problema. En esta sección se presenta el problema de una actualización concurrente y diversas soluciones.

El problema de una actualización concurrente

En la figura 11-1 se ilustra el problema que se presenta cuando se permite a varias sesiones de una base de datos actualizar de manera concurrente los mismos datos. Recuerde que se creará una sesión cada vez que un usuario se conecta a una base de datos, lo que incluye que el mismo usuario se conecte a la base de datos varias veces. El problema de una actualización concurrente ocurre con más frecuencia entre dos usuarios diferentes de una base de datos que no están enterados de que realizan actualizaciones en conflicto sobre los mismos datos. Sin embargo, los usuarios de una base de datos con varias conexiones pueden bloquearse entre sí si aplican actualizaciones mediante más de una de sus sesiones en la base de datos.

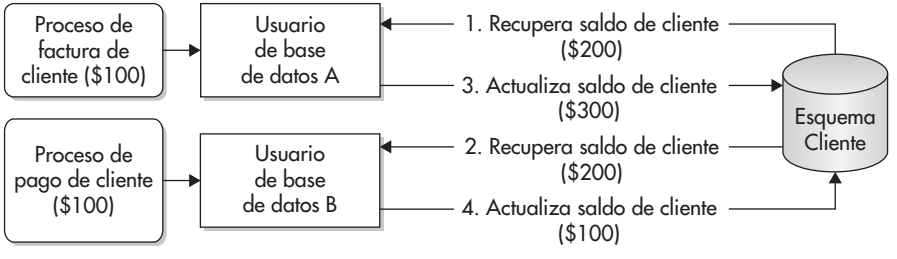


Figura 11-1 El problema de una actualización concurrente.

La situación presentada incluye una empresa imaginaria que vende productos y crea una factura para cada pedido enviado, similar a Industrias Acme en los ejemplos de normalización de los capítulos anteriores. En la figura 11-1 se presenta al usuario A, oficinista del Departamento de Envíos, quien prepara una factura para un cliente, lo cual requiere actualizar los datos del cliente por medio de sumar su saldo adeudado. Al mismo tiempo, el usuario B, oficinista del Departamento de Cuentas por Cobrar, procesa un pago del mismo cliente, lo que requiere actualizar el saldo adeudado por el cliente al restar la cantidad que éste pagó. Ésta es la secuencia exacta de eventos, tal como se aprecia en la figura 11-1:

1. El usuario A consulta la base de datos y recupera el saldo adeudado por el cliente, que es $200.
2. Algunos segundos después, el usuario B consulta la base de datos y recupera el mismo saldo del cliente, que es todavía de $200.
3. Pocos segundos más tarde, el usuario A aplica su actualización y suma una factura de $100 al saldo adeudado, lo que genera un saldo nuevo de $300 en la base de datos.
4. Por último, el usuario B aplica su actualización, y resta un pago de $100 al saldo adeudado que recuperó de la base de datos ($200), lo que genera un saldo nuevo de $100. No está enterado de la actualización realizada por el usuario A y, por lo tanto, establece el saldo adeudado (incorrectamente) en $100.

El saldo adeudado por este cliente debe ser $200, pero la actualización realizada por el usuario B se ha escrito sobre la actualización hecha por el usuario A. La empresa tiene un faltante de $100 que se convertirá en ingresos perdidos o requerirá bastante tiempo del personal para descubrir el error y corregirlo. Como puede ver, permitir actualizaciones concurrentes en la base de datos sin algún tipo de control puede provocar que se pierdan las actualizaciones. Casi todos los vendedores de bases de datos implementan una estrategia de bloqueo para evitar actualizaciones concurrentes en los mismos datos exactos.

Mecanismos de bloqueo

Un *bloqueo* es un control colocado en la base de datos para reservar los datos, de modo que sólo podrán ser actualizados en una sesión a la vez. Cuando se bloquean los datos, ninguna otra sesión de la base de datos puede actualizarlos hasta que se libera el bloqueo, lo que suele conseguirse con una instrucción COMMIT o ROLLBACK de SQL. Algunos DBMS también bloquean los intentos de leer los datos bloqueados. A cualquier sesión que intente actualizar datos bloqueados se le impondrá un estado de *espera en bloqueo*, y la sesión se detendrá hasta que se libere el bloqueo. En algunos productos de base de datos, como DB2 de IBM, expirará el tiempo de una sesión y aparecerá un mensaje de error en lugar de concluir la actualización solicitada. Otros, como Oracle, dejarán una sesión en estado de espera en bloqueo durante un periodo indefinido.

Para este momento debe ser evidente que no existe una variación significativa en la manera en que manejan los bloqueos los productos de base de datos de los diferentes vendedores. A continuación se presenta un resumen general, con la recomendación de que consulte en la documentación de su vendedor de base de datos el modo en que da soporte a los bloqueos. Éstos pueden colocarse en diversos niveles (a lo que suele llamarse *granularidad de bloqueo*), y algunos productos de base de datos, como Sybase ASE, Microsoft SQL Server y DB2 de IBM, permiten varios niveles con una *escala de bloqueo* automática, que aumenta los bloqueos en niveles más altos conforme una sesión de la base de datos coloca más y más bloqueos sobre los mismos objetos de una base de datos. El bloqueo y desbloqueo de pequeñas cantidades de datos ocupa una cantidad importante de recursos generales, por lo que mover la escala de los bloqueos a niveles más altos mejora sustancialmente el desempeño. Los niveles de bloqueo comunes son los siguientes:

- **Base de datos** Se bloquea toda la base de datos para que sólo una sesión pueda aplicar actualizaciones. Es obvio que ésta es una situación extrema que no debe ocurrir con frecuencia, pero puede ser útil cuando se efectúa un mantenimiento importante, como actualizar a una nueva versión del software de la base de datos. Oracle permite este nivel indirectamente cuando se abre la base de datos en modo exclusivo, lo que limita la base de datos a una sesión de usuario.
- **Archivo** Se bloquea un archivo completo de la base de datos. Recuerde que un archivo puede contener parte de una tabla, una tabla completa o partes de muchas tablas. Este nivel tiene menos preferencias en las bases de datos modernas, porque los datos bloqueados pueden ser muy diversos.
- **Tabla** Se bloquea una tabla completa. Este nivel es útil cuando se efectúa un cambio en el nivel de la tabla, como volver a cargar todos los datos en la tabla, actualizar cada fila o modificar la tabla para agregar o eliminar columnas. Oracle llama a este nivel un *bloqueo de DDL,* y se utiliza cuando se aplican instrucciones DDL (CREATE, DROP y ALTER) contra una tabla u otros objetos de la base de datos.
- **Bloque o página** Se bloquea todo un bloque o una página completa dentro de un archivo de la base de datos. Un *bloque* es la unidad más pequeña de datos que el sistema operativo puede leer de un archivo, o escribir en él. En casi todas las computadoras personales, el tamaño de un bloque es el *tamaño del sector*. Algunos sistemas operativos emplean páginas en lugar de bloques. Una *página* es un bloque virtual de tamaño fijo, que suele tener 2K a 4K, y se utiliza para simplificar el procesamiento cuando varios dispositivos de almacenamiento permiten diferentes tamaños de bloque. El sistema operativo puede leer y escribir páginas y permitir que los controladores del hardware traduzcan las páginas a los bloques apropiados. Al igual que con el bloqueo de archivos, el bloqueo de bloques (páginas) tiene menos preferencias en los sistemas de base de datos modernos debido a la diversidad de los datos que pueden inscribirse en el mismo bloque en el archivo.
- **Fila** Se bloquea una fila de una tabla. Éste es el nivel de bloqueo más común y casi todos los sistemas de base de datos modernos lo permiten.

- **Columna** Se bloquean una o más columnas dentro de una fila de la tabla. Este método parece fantástico en teoría, pero no es muy práctico debido a los recursos requeridos para aplicar y liberar bloqueos en este nivel de granularidad. Existe muy poco soporte para este bloqueo en los sistemas modernos de base de datos comercial.

Siempre se colocan bloqueos cuando se actualizan o eliminan datos. Casi todos los RDBMS permiten también el uso de una cláusula FOR UPDATE OF en una instrucción SELECT para que se coloquen bloqueos cuando el usuario de la base de datos declara la *intención* de actualizar algo. Algunos bloqueos pueden ser *exclusivos de lectura*, lo que evita que otras sesiones lean siquiera los datos bloqueados. Muchos RDBMS tienen parámetros de sesión que se configuran para controlar el comportamiento de un bloqueo. Uno de los comportamientos de un bloqueo por considerar es si todas las filas buscadas mediante un cursor se bloquean hasta la siguiente instrucción COMMIT o ROLLBACK, o si se liberan las filas leídas cuando se busque la fila siguiente. Consulte los detalles en la documentación de su vendedor de base de datos.

El principal problema con los mecanismos de bloqueo es que los provocan *contiendas*, lo que significa que la colocación de bloqueos para evitar pérdidas de datos a partir de actualizaciones concurrentes tiene el efecto secundario de provocar que las sesiones concurrentes compitan por el derecho de aplicar actualizaciones. En el mejor de los casos, las contiendas por bloqueos hacen más lentos los procesos de un usuario cuando las sesiones esperan debido a bloqueos. En el peor de los casos, las solicitudes de bloqueo que contienden paralizan las sesiones indefinidamente, como verá en la sección siguiente.

Bloqueo mutuos

Un *bloqueo mutuo* es una situación en que dos o más sesiones de una base de datos tienen bloqueados ciertos datos y después cada una solicita un bloqueo sobre los datos que la otra sesión ha bloqueado. La figura 11-2 ilustra esta situación.

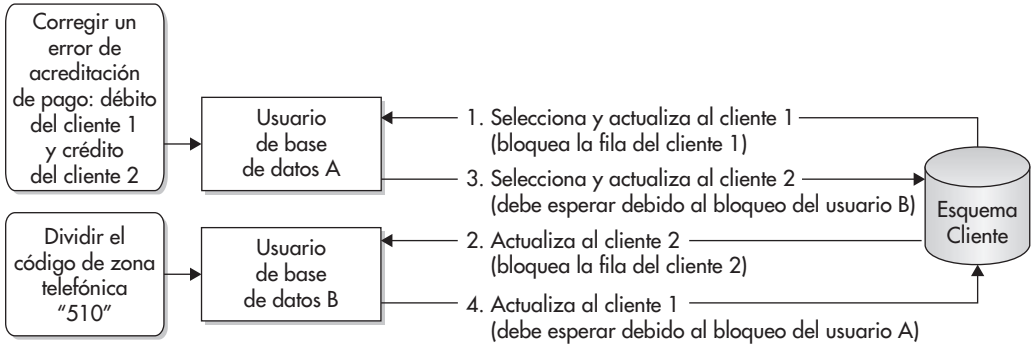


Figura 11-2 El bloqueo mutuo.

Una vez más, este ejemplo utiliza dos usuarios de una empresa imaginaria, llamados A y B. El usuario A es un representante del Departamento de Servicio al Cliente que intenta corregir un pago acreditado a una cuenta de cliente de manera incorrecta. Necesita restar (débito) el pago al cliente 1 y sumarlo (crédito) al cliente 2. El usuario B es un especialista en bases de datos del Departamento de TI y ha escrito una instrucción SQL para actualizar algunos de los números telefónicos de clientes que tienen cierto código de zona telefónica, como respuesta a una reciente división de zonas por parte de la empresa telefónica. La instrucción tiene una cláusula WHERE que limita la actualización a los clientes que tienen un número telefónico con ciertos prefijos en el código de zona 510 y actualiza esos números telefónicos con el nuevo código de zona. El usuario B usa su instrucción UPDATE de SQL mientras el usuario A trabaja en su problema de acreditar un pago. Los clientes 1 y 2 tienen números telefónicos que necesitan actualizarse. La secuencia de eventos (todo esto ocurre con una diferencia de segundos), tal como se exhibe en la figura 11-2, ocurre de este modo:

1. El usuario A selecciona los datos del cliente 1 y aplica una actualización al saldo adeudado. No aplica una confirmación todavía porque esto es sólo una parte de la transacción que debe ocurrir. Se aplica un bloqueo a la fila del cliente 1 a causa de la actualización.
2. La instrucción usada por el usuario B actualiza el número telefónico del cliente 2. La instrucción SQL completa debe ejecutarse como una sola transacción, de modo que en este punto no ocurre una confirmación y, por lo tanto, el usuario B mantiene un bloqueo sobre la fila del cliente 2.
3. El usuario A selecciona el saldo del cliente 2 y luego emite una actualización sobre el saldo adeudado (la misma cantidad que se acreditó al cliente 1). La solicitud debe esperar porque el usuario B mantiene un bloqueo sobre la fila que se va a actualizar.
4. La instrucción emitida por el usuario B ahora intenta actualizar el número telefónico para el cliente 1. La transacción debe esperar porque el usuario A mantiene un bloqueo sobre la fila que se va actualizar.

Ahora estas dos sesiones de la base de datos están en un bloqueo mutuo. El usuario A no puede continuar debido a un bloqueo que mantiene el usuario B, y viceversa. En teoría, estas dos sesiones de la base de datos estarán detenidas para siempre. Por suerte, los DBMS modernos contienen provisiones para manejar esta situación. Un método evita los bloqueo mutuos. Pocos DBMS tienen esta capacidad debido a la considerable cantidad de recursos generales que requiere este método y la imposibilidad virtual de predecir qué hará a continuación un usuario de una base de datos interactiva. Sin embargo, la teoría es revisar la posibilidad de que cada solicitud de bloqueo provoque una contienda y no permitir que ocurra el bloqueo si existe la posibilidad de un bloqueo mutuo. El método más común es la detección de bloqueo mutuo, que aborta una de las dos solicitudes que lo provocaron. Esto se consigue al limitar el tiempo de espera debido a un bloqueo y cancelarlo después de un intervalo preestablecido o al inspeccionar en forma periódica todos los bloqueos para hallar dos sesiones que se bloquean

mutuamente. En cualquier caso, una de las solicitudes debe darse por concluida y deben recuperarse los cambios de la transacción para permitir el avance de la otra solicitud.

Afinación del rendimiento

Cualquier DBA experimentado le dirá que la afinación del rendimiento de una base de datos es una tarea eterna. Siempre es necesario ajustar algo para que funcione con más rapidez, eficacia, o ambas. La clave del éxito consiste en administrar su tiempo y las expectativas de los usuarios de la base de datos, y establecer los requisitos de rendimiento para una aplicación incluso antes de que sea escrita. Suelen funcionar mejor las instrucciones sencillas como "cada actualización de la base de datos debe concluir en 4 segundos". Una vez hecho eso, la afinación del rendimiento se vuelve una sencilla cuestión de buscar cosas que no se apegan al requisito de rendimiento y afinarlas hasta que lo hagan. La ley de la disminución de las ganancias se aplica a la afinación de una base de datos, y es posible aplicar un gran esfuerzo para afinar un proceso de una base de datos que arroja poca o ninguna ganancia. Lo bueno de tener un requisito de rendimiento estándar es que puede detenerse cuando el proceso cumple el requisito y luego pasar al problema siguiente.

Aunque es posible afinar los componentes que no son instrucciones SQL, éstos son tan específicos para un DBMS en particular que es mejor no tratar de cubrirlos aquí. Basta con decir que debe afinar la utilización de la memoria y la CPU, y las entradas/salidas del sistema de archivos, junto con las instrucciones SQL. La afinación de las instrucciones SQL se aborda en la sección siguiente.

Afinación de las consultas a una base de datos

Casi 80% de los problemas de rendimiento de una consulta en una base de datos se resuelve al ajustar una instrucción SQL. Sin embargo, antes de modificarla, debe entender cómo procesa las instrucciones de SQL el DBMS específico que utiliza. Por ejemplo, la colocación de instrucciones SQL dentro de procedimientos guardados produce notables mejorías en el rendimiento en Microsoft SQL Server y Sybase ASE, pero no ocurre lo mismo en Oracle.

Un *plan de ejecución* de consulta es una descripción de cómo un DBMS procesa una consulta en particular, lo que incluye el uso de índices, la lógica de combinación y el costo estimado de los recursos. Es importante aprender a utilizar la utilería "plan de explicación" de su DBMS, si cuenta con una, porque le mostrará exactamente cómo procesa el DBMS la instrucción SQL que pretende afinar. En Oracle, la instrucción EXPLAIN PLAN analiza la instrucción SQL y envía los resultados del análisis a una tabla de planes especial. La tabla de planes debe crearse exactamente como lo especifica Oracle, de modo que para este propósito es mejor emplear la secuencia de comandos que ofrece Oracle. Después de ejecutar la instrucción EXPLAIN PLAN, debe recuperar los resultados de la tabla de planeación mediante una instrucción SELECT. Por suerte, herramientas de Oracle como SQL Developer ofrecen una versión GUI que facilita mucho la afinación de una consulta. La herramienta Consulta,

incluida en Microsoft SQL Server Management Studio (SQL Server 2005 y 2008) contiene los botones Mostrar plan de ejecución estimado e Incluir plan de ejecución real, que muestran de manera gráfica cómo se ejecutará la instrucción SQL. Estas opciones también se acceden desde el menú Consulta. En las versiones anteriores de Microsoft SQL Server, estas opciones (con nombres distintos) se encuentran en la herramienta Analizador de consultas.

A continuación se ofrecen algunas sugerencias generales de afinación para SQL que se aplican a casi todas las implementaciones. Debe consultar una guía de afinación para el DBMS específico que utiliza, porque las técnicas, sugerencias y otras consideraciones varían según el producto.

Evite exploraciones de tablas grandes. En el caso de tablas con más de 1 000 filas, puede ser costosa la exploración de todas sus filas en lugar de utilizar un índice, considerando los recursos requeridos. Y, por supuesto, cuanto más grande sea la tabla, más costosa será una nueva exploración. Las exploraciones de tablas completas ocurren en las situaciones siguientes:

- La consulta no contiene una cláusula WHERE para limitar las filas.
- Ninguna de las columnas a las que se hace referencia en la cláusula WHERE coincide con la columna principal de un índice en la tabla.
- No se han actualizado las estadísticas del índice y la tabla. Casi todos los optimizadores de consultas de un RDBMS emplean estadísticas para evaluar los índices disponibles, y sin estadísticas, la exploración de una tabla puede considerarse más eficiente que el uso de un índice.
- Cuando menos una columna de la cláusula WHERE no coincide con la primera columna de un índice disponible, pero la comparación utilizada hace que resulte obvio el uso de un índice. Entre estos casos están los siguientes:
  - Aplique el operador NOT (por ejemplo, **WHERE NOT CITY = 'NEW YORK'**). En general, es posible utilizar índices para hallar lo que *está* en una tabla, pero no se pueden emplear para encontrar lo que *no* está en ella.
  - Utilice el operador NOT EQUAL (por ejemplo, **WHERE CITY <> 'NEW YORK'**).
  - Incluya un comodín en la primera posición de una cadena de comparación (por ejemplo, **WHERE CITY LIKE '%YORK'**).
  - Emplee una función de SQL en la comparación (por ejemplo, **WHERE UPPER (CITY) = 'NEW YORK'**).

Cree índices que sean selectivos. La *selectividad de un índice* es un cociente del número de valores distintos que tiene una columna, dividido entre la cantidad de filas de una tabla. Por ejemplo, si una tabla tiene 1 000 filas y una columna tiene 800 valores distintos, la selectividad del índice es 0.8, lo que se considera bueno. Sin embargo, una columna Género que sólo tiene dos valores distintos (M y F) tiene una selectividad muy baja (en este caso, 0.002). Los índices únicos siempre tienen una selectividad de 1.0, que es la mejor posible. Con algu-

nos RDBMS como DB2, los índices únicos son tan superiores que los DBA suelen agregar columnas en otras circunstancias innecesarias a un índice, sólo para hacerlo único. Sin embargo, recuerde siempre que los índices ocupan espacio de almacenamiento y deben recibir mantenimiento, de modo que nunca son un "bocado gratuito".

Evalúe con atención las técnicas de combinación. Casi todos los RDBMS ofrecen varios métodos para combinar tablas, y el optimizador de consultas en el RDBMS elige el que parece mejor con base en las estadísticas de la tabla. En general, la creación de índices en columnas de clave externa da al optimizador más opciones entre las cuales elegir, lo que siempre es bueno. Ejecute un plan de explicación y consulte la documentación de su RDBMS cuando afine combinaciones.

Ponga atención a las vistas. Debido a que las vistas son consultas guardadas de SQL, pueden presentar problemas de rendimiento igual que cualquier otra consulta.

Afine las consultas secundarias de acuerdo con las recomendaciones del vendedor de su RDBMS.

Limite el empleo de tablas remotas. Las tablas conectadas a enlaces de base de datos de manera remota nunca funcionan tan bien como las tablas locales.

Las tablas muy grandes requieren atención especial. Cuando el tamaño de las tablas aumenta a millones de filas, cualquier consulta puede ser una pesadilla para el rendimiento. Evalúe con atención cada consulta, y considere particionar la tabla para mejorar el rendimiento de la consulta. La partición de una tabla se explica en el capítulo 8. Su RDBMS puede ofrecer otras funciones especiales para tablas muy grandes que mejoren el rendimiento de una consulta.

Pregunta al experto

P: **A menudo no identifico si en la base de datos se utilizaron mayúsculas para nombres propios como las ciudades. Mencionó que el empleo de una función como UPPER en el predicado (por ejemplo,** WHERE UPPER (CIUDAD) = 'NUEVA YORK') **hace obvio el uso de un índice en esa columna. ¿Existen soluciones temporales para esto?**

R: Se me ocurren varias. En primer lugar, si utiliza un DBMS que permite comparaciones que no distinguen mayúsculas, como SQL Server, Sybase ASE o Microsoft Access, la función no es necesaria porque no importa si emplea mayúsculas en el predicado **WHERE**. En segundo lugar, si el DBMS permite lo que se conoce como un *índice basado en una función*, puede crear un índice en una expresión como **UPPER (CIUDAD)** y después los predicados que empleen la misma función en la misma columna pueden usar el índice. Oracle permite esta característica. En tercer lugar, puede guardar los datos en dos columnas: una tal como la introduce el usuario, y la otra convertida a mayúsculas o minúsculas para búsquedas. Si bien ésta no es una gran idea en una base de datos para procesamiento de transacciones, es una técnica común en los almacenes de datos y los mercados de datos, en donde los datos redundantes no suelen conducir a ningún problema de consistencia de los datos. (Estos tipos de base de datos se analizan con detalle en el capítulo 12.)

Afinación de las instrucciones DML

Las instrucciones en el lenguaje de manipulación de datos (DML, Data Manipulation Language) suelen producir menos problemas de rendimiento que las de una consulta. No obstante, pueden ocurrir problemas.

Las instrucciones INSERT tienen dos consideraciones importantes:

- **Asegurar un espacio libre adecuado en los espacios de tabla para contener filas nuevas** Los espacios de tabla que tienen poco espacio presentan problemas cuando el DBMS busca espacio libre para contener las filas que se insertan. Además, las inserciones no suelen poner las filas de la tabla en la secuencia de la clave principal porque el espacio libre no suele estar disponible exactamente en los lugares correctos. Además, la reorganización de la tabla, que en esencia es un proceso de descargar las filas en un archivo simple, volver a crear la tabla y volver a cargarla, mejora el rendimiento de las inserciones y las consultas.
- **Mantenimiento de índices** Cada vez que se inserta una fila en una tabla, debe insertarse una entrada correspondiente en cada índice creado en la tabla (no obstante, nunca debe asignar un índice a un valor nulo). Cuantos más índices existan, más recursos generales requerirá cada inserción. Se suele afinar el espacio libre de los índices igual que el espacio libre de las tablas.

Las instrucciones UPDATE tienen las consideraciones siguientes:

- **Mantenimiento de índices** Si se actualizan las columnas que contienen índices, también deben actualizarse las entradas de índices correspondientes. En general, la actualización de valores de clave principal tiene implicaciones particularmente inconvenientes en el rendimiento (tanto que algunos RDBMS la prohíben).
- **Expansión de las filas** Cuando se actualizan las columnas de un modo que aumenta significativamente el tamaño de una fila, ésta ya no cabe en su ubicación original y es posible que no haya suficiente espacio libre alrededor de ella para que crezca (puede haber otras filas a un lado de la que se actualiza). Cuando ocurre esto, la fila debe moverse a un lugar del archivo de datos donde quepa o se divida, y donde la parte ampliada quede en una nueva ubicación, conectada a la ubicación original mediante un apuntador. Cuando ocurren, estas situaciones no sólo son costosas, sino que también deterioran el rendimiento de las consultas posteriores que incluyen esas filas. Las reorganizaciones de las tablas resuelven el problema, pero es mejor evitarlas mediante el diseño de la aplicación de modo que el tamaño de las filas no tienda a crecer después de que se insertan.

Es probable que las instrucciones DELETE sean las que presenten menos problemas de rendimiento. Sin embargo, una tabla que participa como primaria en una relación que se define con una opción ON DELETE CASCADE puede funcionar de manera deficiente si existen muchas filas secundarias por eliminar.

Control de cambios

El *control de cambios* (también conocido como *administración de cambios*) es el proceso utilizado para administrar los cambios que ocurren después de que se implementa un sistema. Un proceso de control de cambios ofrece los beneficios siguientes:

- Ayuda a comprender cuándo es aceptable hacer cambios y cuándo no.
- Ofrece un registro de todos los cambios que se han realizado para ayudar a la detección y solución cuando ocurren problemas.
- Puede administrar versiones de los componentes del software para que sea posible descartar sin problemas una versión defectuosa.

El cambio es inevitable. No sólo cambian los requisitos de negocios, sino que en algún momento se incorporan versiones nuevas del software de la base de datos y del sistema operativo, así como dispositivos de hardware novedosos. Los expertos en tecnología deben diseñar un método de control de cambios conveniente para la organización, y la administración debe aprobarlo como una norma. Cualquier cosa que implique menos que eso conduce al caos cuando se hacen cambios sin una coordinación y comunicaciones adecuadas. Aunque la terminología varía entre los métodos estándar, todos tienen características comunes:

- **Numeración de versión** Se asignan números de versión a los componentes de un sistema de aplicaciones, que suelen comenzar con 1 y avanzar en secuencia cada vez que se modifica el componente. El número de versión suele incluir una fecha de revisión y el identificador de la persona que realiza el cambio.
- **Numeración de lanzamiento (construcción)** Un *lanzamiento* es un punto en el tiempo en que se promueven todos los componentes de un sistema de aplicaciones (incluidos los componentes de la base de datos) al entorno siguiente (por ejemplo, pasa de desarrollo a prueba del sistema) como un conjunto que se puede probar e implementar al mismo tiempo. Algunas organizaciones prefieren el término *construcción*. Los entornos de base de datos se analizan en el capítulo 5. Cuando se forman lanzamientos, es importante asignar un nombre a cada componente incluido con el número de lanzamiento (o construcción). Esto le permite saber cuál versión de cada componente fue incluida en un lanzamiento en particular.
- **Priorización** Es posible asignar prioridades a los cambios para poder programarlos de acuerdo con ellas.
- **Control de solicitudes de cambios** Las solicitudes de cambios se pueden colocar en el sistema de control de cambios, enrutarse mediante canales de aprobación y marcarse con el número de liberación aplicable cuando se concluye el cambio.

- **Reserva y liberación** Cuando un desarrollador o DBA está preparado para aplicar cambios a un componente, debe ser capaz de reservarlo, lo cual evita que otras personas realicen cambios que podrían entrar en conflicto con el mismo componente al mismo tiempo. Cuando concluye el trabajo, el desarrollador o DBA libera el componente, lo que en esencia retira el resguardo.

Es posible desplegar varios productos de software comercial y freeware para ayudar al control de cambios. Sin embargo, es importante que establezca el proceso *antes* de elegir las herramientas. De este modo, la organización puede establecer el mejor proceso para sus necesidades y encontrar la herramienta que se ajuste mejor al proceso, en lugar de tratar de que el proceso se ajuste a la herramienta.

Desde la perspectiva de una base de datos, el DBA debe desarrollar instrucciones de DDL para implementar todos los componentes de la base de datos como un sistema de aplicaciones, y una secuencia de comandos que sirva para invocar todos los cambios, lo que incluye todas las conversiones requeridas. Esta secuencia de comandos de implementación y todo el DDL deben comprobarse en el sistema de control de cambios y administrarse igual que todos los otros componentes del software del sistema.

✓ Autoexamen Capítulo 11

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. Un cursor es _____.

2. Un conjunto de resultados es _____.

3. La *I* en las siglas ACID significa _____.

4. Antes que sea posible buscar filas con un cursor, el cursor debe

- **A** Declararse.
- **B** Consignarse.
- **C** Abrirse.
- **D** Cerrarse.
- **E** Purgarse.

5. Una transacción

- **A** Puede procesarse y consignarse de manera parcial.
- **B** No puede procesarse y consignarse de manera parcial.

C Cambia la base de datos de un estado consistente a otro.

D A veces se le denomina *unidad de trabajo*.

E Posee las propiedades descritas mediante las siglas ACID.

6. Microsoft SQL Server permite los modos de transacciones siguientes:

A Confirmación automática.

B Automático.

C Durable.

D Explícito.

E Implícito.

7. Oracle permite los modos de transacciones siguientes:

A Confirmación automática.

B Automático.

C Durable.

D Explícito.

E Implícito.

8. Las instrucciones SQL (comandos) que concluyen una transacción son

A **SET AUTOCOMMIT**.

B **BEGIN TRANSACTION** (en SQL Server).

C **COMMIT**.

D **ROLLBACK**.

E **SAVEPOINT**.

9. El problema de actualización concurrente

A Es una consecuencia de compartir datos de manera simultánea.

B No puede ocurrir cuando **AUTOCOMMIT** se establece en **ON**.

C Es la razón de que deba permitirse el bloqueo de transacciones.

D Ocurre cuando dos usuarios de una base de datos emiten instrucciones SELECT en conflicto.

E Ocurre cuando dos usuarios de una base de datos hacen actualizaciones en conflicto sobre los mismos datos.

10. Un bloqueo

- **A** Es un control aplicado a los datos para reservarlos, de modo que el usuario pueda actualizarlos.
- **B** Se suele liberar cuando ocurre una instrucción COMMIT o ROLLBACK.
- **C** En DB2 y otros productos RDBMS, tiene establecido un tiempo de expiración.
- **D** Puede provocar contiendas cuando otros usuarios intentan actualizar los datos bloqueados.
- **E** Puede tener varios niveles y un protocolo de escala en algunos productos RDBMS.

11. Un bloqueo mutuo

- **A** Es un bloqueo cuyo tiempo ha expirado y, por lo tanto, ya no es necesario.
- **B** Ocurre cuando dos usuarios de una base de datos solicitan un bloqueo sobre datos que están bloqueados por el otro.
- **C** En teoría puede poner a dos o más usuarios en un interminable estado de espera debido a bloqueo.
- **D** En algunos RDBMS, se resuelve mediante detección de bloqueos mutuos.
- **E** En algunos RDBMS, se resuelve mediante un tiempo de expiración de bloqueo.

12. La afinación del rendimiento

- **A** Es un proceso que nunca concluye.
- **B** Debe utilizarse en cada consulta hasta que no pueda realizarse ninguna mejora.
- **C** Debe aplicarse sólo a consultas que no se apegan a los requisitos de rendimiento.
- **D** Implica no sólo afinar el SQL, sino también la CPU, las entradas/salida del sistema de archivos, y el uso de la memoria.
- **E** Debe basarse en los requisitos.

13. La afinación de una consulta SQL

- **A** Se efectúa del mismo modo en todos los sistemas de base de datos relacional.
- **B** Suele implicar el uso de una opción de plan de explicación.
- **C** Siempre conlleva la colocación de instrucciones SQL en un procedimiento almacenado.
- **D** Sólo se aplica a las instrucciones SELECT de SQL.
- **E** Requiere un conocimiento detallado del RDBMS en que se va a ejecutar la consulta.

14. ¿Cuáles son sugerencias generales para afinar el SQL?

- **A** Evitar exploraciones de tablas grandes.
- **B** Emplear un índice cada vez que sea posible.
- **C** Incluir una cláusula ORDER BY cada vez que sea posible.
- **D** Utilizar una cláusula WHERE para filtrar las filas cada vez que sea posible.
- **E** Usar vistas cada vez que sea posible.

15. Las prácticas de SQL que vuelven obvio el uso de un índice son

- **A** El uso de la cláusula WHERE.
- **B** La inclusión de un operador NOT.
- **C** La aplicación de combinaciones de tablas.
- **D** La utilización del operador NOT EQUAL.
- **E** El uso de comodines en la primera columna de las cadenas de comparación mediante LIKE.

16. Los índices funcionan bien para filtrar filas cuando

- **A** Son muy selectivos.
- **B** El cociente de selectividad es muy alto.
- **C** El cociente de selectividad es muy bajo.
- **D** Son únicos.
- **E** No son únicos.

17. Las principales consideraciones relacionadas con rendimiento de las instrucciones INSERT son

- **A** La expansión de filas.
- **B** El mantenimiento de índices.
- **C** El uso del espacio libre.
- **D** La afinación de las consultas secundarias.
- **E** Las tablas muy grandes relacionadas.

18. Las principales consideraciones relacionadas con el rendimiento de las instrucciones UPDATE son

- **A** La expansión de filas.
- **B** El mantenimiento de índices.
- **C** El uso del espacio libre.
- **D** La afinación de consultas secundarias.
- **E** Las tablas muy grandes relacionadas.

19. Un proceso de control de cambios

- **A** Evita que los errores de programación sean aplicados en el entorno de producción.
- **B** También se denomina *administración de cambios*.
- **C** Ayuda a comprender cuándo pueden instalarse cambios.
- **D** Proporciona un registro de todos los cambios realizados.
- **E** Permite que se descarten versiones de software defectuosas.

20. ¿Cuáles son características comunes de los procesos de control de cambios?

- **A** Soporte a transacciones.
- **B** Numeración de versión.
- **C** Prevención de bloqueo mutuo.
- **D** Numeración de lanzamiento.
- **E** Priorización.

Capítulo 12

Bases de datos para procesamiento analítico en línea

Habilidades y conceptos clave

- Almacenes de datos
- Mercados de datos
- Minería de datos

A partir de la década de 1980, las empresas reconocieron la necesidad de conservar datos históricos y utilizarlos para análisis con el fin de apoyar la toma de decisiones. Pronto fue evidente que los datos organizados para las transacciones empresariales cotidianas no eran tan útiles para el análisis. En realidad, guardar cantidades importantes del historial en una base de datos *operativa* (una base de datos diseñada para permitir las transacciones diarias de una organización) podía tener serios efectos nocivos en el desempeño. William H. (Bill) Inmon fue un pionero que desarrolló el concepto conocido como *almacenamiento de datos*, en que los datos históricos se retiran periódicamente de una base de datos operativa y se trasladan a una base de datos específicamente diseñada para análisis. La dedicada promoción que hizo Inmon del concepto le ganó el título de "padre del almacenamiento de datos".

La popularidad del método de almacén de datos aumentó con cada historia de éxito. Además de Inmon, otras personas hicieron contribuciones importantes, de manera destacada Ralph Kimball, quien desarrolló arquitecturas de base de datos especializadas para los almacenes de datos (lo que se cubre en la sección "Arquitectura de un almacén de datos" más adelante en el capítulo). E. F. (Ted) Codd incorporó su respaldo al método del almacén de datos y acuñó dos términos importantes en 1993:

- **Procesamiento de transacciones en línea (OLTP, Online Transaction Processing)** Sistemas diseñados para manejar los altos volúmenes de transacciones que se efectúan durante las actividades cotidianas de una organización.
- **Procesamiento analítico en línea (OLAP, Online Analytical Processing)** Análisis de datos (generalmente históricos) para identificar las tendencias que apoyan la toma de decisiones estratégicas en relación con el negocio.

Hasta este momento, los capítulos de este libro se han referido casi exclusivamente a bases de datos OLTP. Por su parte, este capítulo se dedica exclusivamente a los conceptos de una base de datos OLAP.

Almacenes de datos

De acuerdo con la definición de Inmon, un *almacén de datos* es un conjunto de datos orientado a temas, integrado, que varía con el tiempo y no es volátil, diseñado para apoyar la toma de decisiones administrativas. Éstas son algunas propiedades importantes de los almacenes de datos:

- Están organizados alrededor de las principales áreas temáticas de una organización, como ventas, clientes, proveedores y productos. Por otra parte, los sistemas OLTP suelen organizarse alrededor de los sucesos principales, como nómina, recepción de pedidos, facturación, etcétera.
- Se integran a partir de numerosos orígenes de datos (OLTP) operativas.
- No se actualizan en tiempo real, sino de manera periódica, con base en un programa establecido. Los datos se extraen de fuentes operativas con la frecuencia necesaria, que puede ser diaria, semanal, mensual y trimestral.

Son importantes los posibles beneficios de un almacén de datos bien construido, y entre ellos están los siguientes:

- Aportan una ventaja competitiva.
- Quienes toman decisiones corporativas obtienen mayor productividad.
- Alto potencial de ganancias sobre la inversión, a medida que la organización detecta los mejores modos para mejorar la eficiencia, la rentabilidad, o ambas.

No obstante, existen algunos desafíos importantes para crear un almacén de datos en el nivel de toda una empresa, entre ellos los siguientes:

- Se llegan a subestimar los recursos requeridos para cargar los datos.
- Existen problemas ocultos de integridad en los datos de origen.
- Se omiten datos sólo para descubrir después que eran necesarios.
- Las exigencias de los usuarios finales son cada vez mayores (cada función nueva genera ideas para aún más funciones).
- Es necesario consolidar datos de orígenes dispares.
- Las demandas de recursos son elevadas (enormes cantidades de almacenamiento; consultas que procesan millones de filas).
- Se debe definir la propiedad de los datos.
- Es difícil determinar lo que el negocio en realidad quiere o necesita analizar.
- Son proyectos “de gran trascendencia” que parecen interminables.

Comparación entre sistemas OLTP y sistemas de almacén de datos

Los sistemas de almacén de datos y los OLTP son fundamentalmente diferentes. En seguida se presenta una comparación:

Sistemas OLTP	Sistemas de almacén de datos
Contienen datos actuales.	Contienen datos históricos.
Guardan datos actuales.	Guardan datos detallados, junto con datos poco y muy resumidos.
Los datos son dinámicos.	Los datos son estáticos, excepto por adiciones periódicas.
Las consultas a la base de datos son de corta duración y acceden a relativamente pocas filas de datos.	Las consultas a la base de datos son de larga duración y acceden a muchas filas de datos.
Volumen de transacciones elevado.	Volumen de transacciones de mediano a bajo.
Procesamiento repetitivo; esquema de uso predecible.	Procesamiento ad hoc y no estructurado; esquema de uso impredecible.
Orientados a las transacciones; permiten operaciones cotidianas.	Orientados al análisis; permiten toma de decisiones estratégicas.
Orientados a procesos.	Orientados a temas.
Atienden a una cantidad grande de usuarios concurrentes.	Atienden a una cantidad relativamente baja de usuarios directivos (personas que toman decisiones).

Arquitectura de un almacén de datos

Dos escuelas de pensamiento reinan como el mejor modo para organizar los datos OLTP en un almacén de datos: el *método de la tabla de resumen* y el *método del esquema de estrella*. En las secciones siguientes se analiza cada uno, junto con sus beneficios y desventajas.

Arquitectura de tabla de resumen

Inmon desarrolló la arquitectura de almacén de datos con tabla de resumen. Este método de almacenamiento de datos requiere que los datos no sólo se guarden en forma detallada, sino también en tablas de resumen para que los procesos de análisis no tengan que resumir los mismos datos una y otra vez. Ésta es una violación obvia de los principios de la normalización, pero debido a que los datos son históricos (y, por lo tanto, no se espera que cambien una vez guardados), sencillamente no existen las anomalías de datos (inserción, actualización y eliminación) que impulsan la necesidad de una normalización. En la figura 12-1 se muestra la arquitectura de almacén de datos con tabla de resumen.

Los datos de uno o más orígenes operativos (bases de datos o sistemas de archivo simple) se trasladan periódicamente a la base de datos del almacén. Una clave importante para el éxito consiste en determinar el nivel de detalle adecuado que debe incorporarse en la base

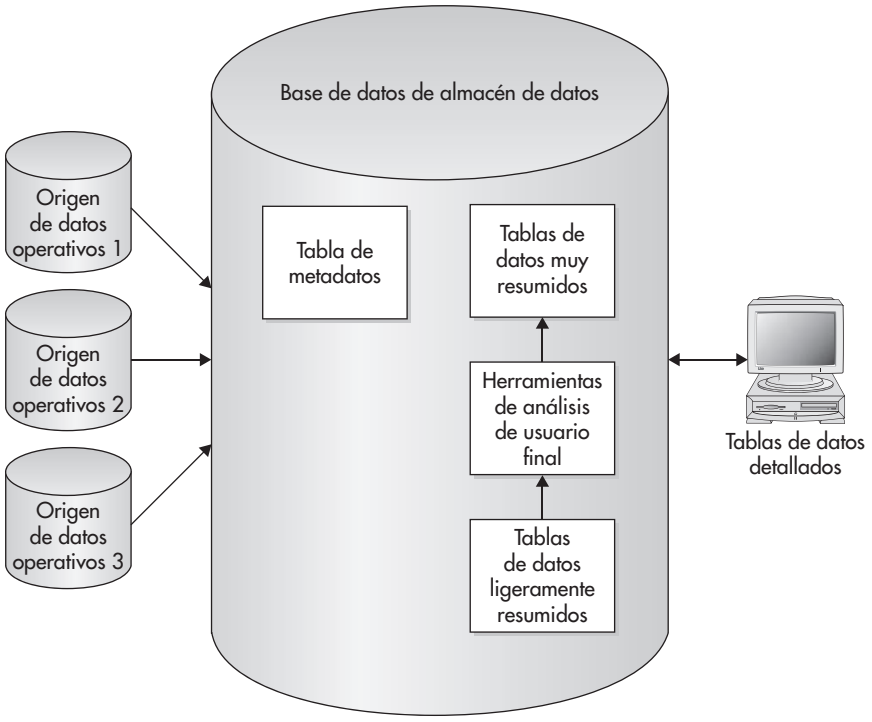


Figura 12-1 Arquitectura de almacén de datos con tabla de resumen.

de datos y prever los niveles de resumen necesarios. En el ejemplo de Industrias Acme, si el tema del almacén de datos son las ventas, tal vez sea necesario conservar cada factura, o sólo las que superan cierto importe; o tal vez sólo las que contengan ciertos productos. Si no se comprenden los requisitos, es improbable que tenga éxito el proyecto de un almacén de datos. El porcentaje de fracasos de proyectos de almacén de datos es mayor que el de casi todos los otros tipos de proyectos de TI, y la causa más común es una definición deficiente de los requisitos.

En cuanto a la creación de resúmenes, es posible resumir las transacciones por mes en una tabla y por producto en otra. En el siguiente nivel, es posible resumir los meses por trimestre en una tabla y los productos por departamento en otra. Un *usuario final* (la persona que emplea las herramientas de análisis para obtener resultados de la base de datos OLAP) puede analizar las ventas por trimestre y observar que un trimestre específico no parece normal. El usuario puede ampliar el trimestre que le interesa y examinar los meses que contiene. Al proceso se le conoce como "profundización" a niveles más detallados. Después, el usuario puede elegir un mes específico y estudiar las transacciones detalladas.

Los metadatos (datos acerca de los datos) presentados en la figura 12-1 son muy importantes y, por desgracia, suelen ser el eslabón perdido. Lo ideal es que definan cada elemento de datos del almacén, junto con información suficiente para rastrear su origen hasta los datos básicos en la base de datos operativa. El desafío más grande con los metadatos es que, al carecer de normas, cada vendedor de herramientas de almacén ha guardado los metadatos a su manera. Cuando están en uso varias herramientas de análisis, los metadatos suelen cargarse en cada una de ellas mediante formatos patentados. Para las herramientas de análisis de usuario final (también llamadas herramientas OLAP o de inteligencia empresarial), no sólo se ofrecen herramientas incrustadas en los principales productos de base de datos relacional, cómo SQL Server y Oracle, sino que existen literalmente docenas de productos comerciales especializados, entre ellos Business Objects (ahora propiedad de SAP), Cognos (una empresa de IBM), Actuate, Hyperion (ahora propiedad de Oracle) y muchos más.

Arquitectura de almacén de datos de esquema de estrella

Kimball desarrolló una estructura de base de datos especializada, conocida como *esquema de estrella*, para guardar los datos de un almacén. Su contribución al almacenamiento de datos OLAP es significativa. Red Brick, el primer DBMS dedicado exclusivamente al almacenamiento de datos OLAP, utilizaba el esquema de estrella. Además, Red Brick ofrecía extensiones de SQL específicamente para análisis de datos, que incluían promedios móviles, comparación entre años, participación de mercado y clasificación. Informix adquirió la tecnología de Red Brick, y más tarde IBM adquirió Informix, de modo que ahora IBM comercializa la tecnología Red Brick como parte de su solución de almacén de datos. En la figura 12-2 se muestra la arquitectura básica de un almacén de datos que utiliza el esquema de estrella.

El esquema de estrella aplica una sola tabla de datos detallada, llamada *tabla de hechos*, rodeada por tablas de datos de consulta de apoyo llamadas *tablas de dimensiones*, que forman un esquema de estrella. Comparada con la arquitectura de almacén de datos con tabla de resumen, la tabla de hechos reemplaza las tablas de datos detallados, y evidentemente las tablas de dimensiones reemplazan a las tablas de resumen. Aparte de la clave principal, cada atributo de la tabla de hechos debe ser un *hecho* (una medida que se puede resumir) o una clave externa de una tabla de dimensiones. Recuerde que los hechos deben tener la capacidad de sumarse, como cantidades, calificaciones, intervalos de tiempo e importes en moneda. Se construye un nuevo esquema de estrella para cada tabla de hechos adicional. Las tablas de dimensiones tienen una relación uno a varios con la tabla de hechos, y la clave principal de la tabla de dimensiones aparece como una clave ajena en la tabla de hechos. Sin embargo, no es necesario que las tablas de dimensiones estén normalizadas, porque pueden tener una jerarquía completa, como las capas de una organización o diferentes componentes secundarios de tiempo comprimidos en una sola tabla. Las tablas de dimensiones pueden o no contener información de resumen, como totales, pero por lo general no deben contener hechos.

Mediante el ejemplo anterior de ventas de Industrias Acme, la tabla de hechos contendría las facturas de la tabla, y las tablas de dimensiones normales serían tiempo (días, meses, tri-

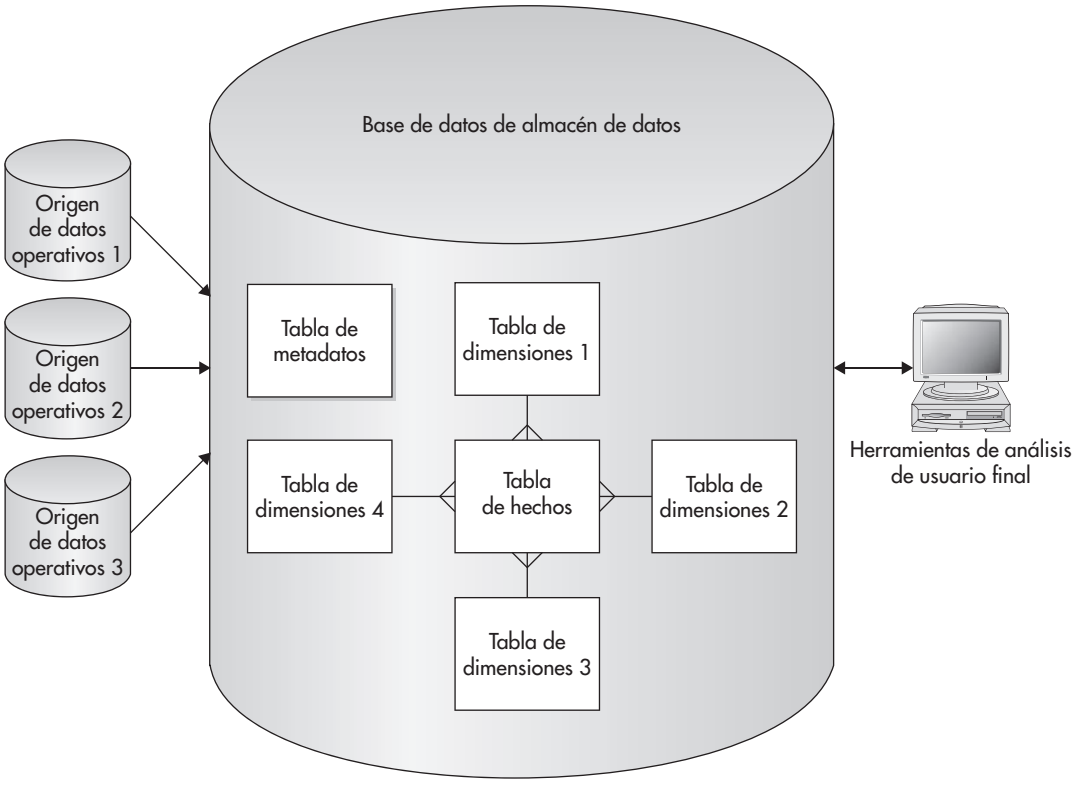


Figura 12-2 Arquitectura de almacén de datos de esquema de estrella.

Pregunta al experto

P: **He escuchado que los esquemas de estrella son muy difíciles de usar cuando un análisis requiere combinar datos de varias tablas de hechos. ¿Existe algún modo de evitar estos problemas?**

R: Sí, por supuesto, pero la solución consiste en diseñar las dimensiones de manera correcta, en lugar de emplear soluciones temporales después de que se implementa el almacén de datos. Por ejemplo, si la dimensión de tiempo en un esquema emplea meses calendario y otra aplica meses fiscales, tal vez sea imposible combinarlas, a menos que de alguna manera se puedan aplicar días individuales. El truco consiste en utilizar lo que Kimball denomina *dimensiones que concuerdan*, que son dimensiones con estructura, atributos, valores de dominio, definiciones y conceptos idénticos. Al aplicar ese principio, cada dimensión de tiempo en la base de datos se definiría de un modo idéntico, tal vez como días calendario, que se transforman fácilmente en semanas, meses y trimestres calendario o fiscales.

mestres y, tal vez, años), productos y unidades organizativas (departamentos, divisiones, etc.). En realidad, las estructuras de tiempo y organizativas aparecen como dimensiones en casi todos los esquemas de estrella. Como puede suponer, la clave del éxito de las bases de datos OLAP de esquema de estrella está en preparar la tabla de hechos correcta y emplear sólo las dimensiones que concuerden. Ésta es una lista de las consideraciones que afectan el diseño de la tabla de hechos:

- El periodo requerido (con cuánta frecuencia se agregarán datos y cuánto historial debe contener la base de datos OLAP).
- El almacenamiento de cada transacción, en comparación con un muestreo estadístico.
- Las columnas de las tablas de datos de origen que no necesariamente son para OLAP.
- Las columnas cuyo tamaño se puede reducir.
- Los mejores modos de utilizar claves inteligentes (naturales) y sustitutas (tontas).
- Partición de la tabla de hechos.

Con el tiempo, surgieron algunas variaciones del esquema de estrella:

- **Esquema de copo de nieve** Una variante en que se permite que las dimensiones tengan medidas propias. El nombre proviene de la semejanza del diagrama entidad-relación con un copo de nieve. Si normaliza por completo las dimensiones de un esquema de estrella, termina con un esquema de copo de nieve. Por ejemplo, la dimensión de tiempo en el primer nivel podría rastrear los días, con una tabla de dimensiones en el siguiente nivel para rastrear las semanas, otra más para rastrear los meses, una más para los trimestres, etc. Es posible emplear diseños similares para rastrear la jerarquía de una organización (departamentos, divisiones y demás).
- **Esquema de copo de estrella** Un diseño híbrido que contiene una mezcla de dimensiones de estrella (no normalizadas) y de copo de nieve (normalizadas).

Bases de datos multidimensionales

Las bases de datos de varias dimensiones evolucionaron a partir de los esquemas de estrella. También se conocen como bases de datos OLAP multidimensionales (MOLAP). Existen en el mercado varios sistemas especializados de este tipo de base de datos, entre ellos Oracle Express, Microsoft SQL Server Analysis Services y Oracle Essbase. Las bases de datos MOLAP se visualizan mejor como cubos, en donde cada dimensión forma un lado del cubo. Para alojar dimensiones adicionales, sencillamente se repite el cubo (o el grupo de cubos) para cada una.

En la figura 12-3 se expone una tabla de hechos de cuatro columnas para Industrias Acme. Las dimensiones son Línea de productos, Departamento de ventas y Trimestre, y serían claves externas de una tabla de dimensiones en un esquema de estrella. Cantidad contiene

Línea de productos	Departamento de ventas	Trimestre	Cantidad
Cascos	Ventas corporativas	1	2 250
Cascos	Ventas corporativas	2	2 107
Cascos	Ventas corporativas	3	5 203
Cascos	Ventas corporativas	4	5 806
Cascos	Ventas por Internet	1	1 607
Cascos	Ventas por Internet	2	18 112
Cascos	Ventas por Internet	3	4 834
Cascos	Ventas por Internet	4	5 150
Resortes	Ventas corporativas	1	16 283
Resortes	Ventas corporativas	2	17 422
Resortes	Ventas corporativas	3	21 288
Resortes	Ventas corporativas	4	32 768
Resortes	Ventas por Internet	1	12
Resortes	Ventas por Internet	2	24
Resortes	Ventas por Internet	3	48
Resortes	Ventas por Internet	4	48
Cohetes	Ventas corporativas	1	65
Cohetes	Ventas corporativas	2	38
Cohetes	Ventas corporativas	3	47
Cohetes	Ventas corporativas	4	52
Cohetes	Ventas por Internet	1	2
Cohetes	Ventas por Internet	2	1
Cohetes	Ventas por Internet	3	6
Cohetes	Ventas por Internet	4	9

Figura 12-3 Tabla de hechos de cuatro columnas de Industrias Acme.

la cantidad de unidades vendidas para cada combinación de Línea de productos, Departamento de ventas y Trimestre.

En la figura 12-4 se presenta el equivalente multidimensional de la tabla mostrada en la figura 12-3. Observe que Departamento de ventas, Línea de productos y Trimestre se convierten en bordes del cubo, y el hecho único Trimestre se guarda en cada cuadro de la cuadrícula. Las dimensiones mostradas se modifican con sólo girar el cubo.

Cuando una dimensión contiene datos que cambian con el tiempo, como un producto que pasa de una familia de productos a otra, se le conoce como *dimensión de cambio lento*. Esto

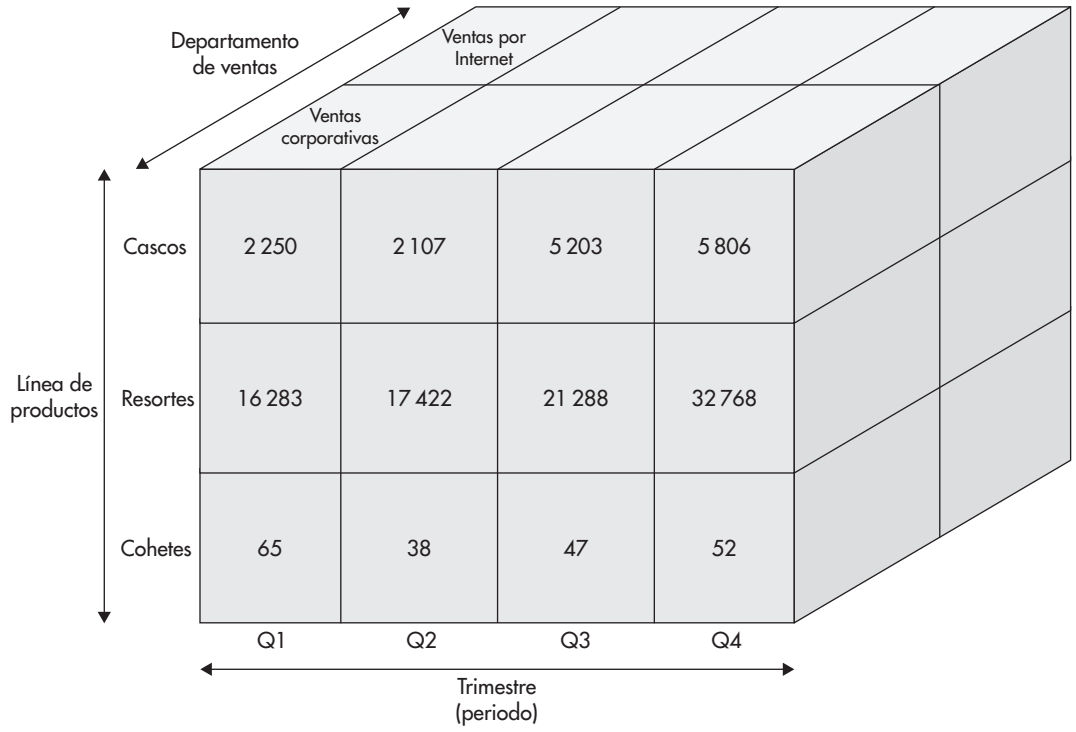


Figura 12-4 Cubo tridimensional para Industrias Acme.

representa un desafío especial cuando se diseñan esquemas de varias dimensiones. En la tabla siguiente se presentan varios métodos de solución, conocidos como *tipos* de dimensiones de cambio lento:

Tipo de método	Descripción
1	Los datos nuevos se escriben sobre los antiguos, de modo que no ocurre un rastreo del historial.
2	Se crea una fila nueva cada vez que cambian los datos en la dimensión, lo que ofrece un historial ilimitado. En cada fila se incluye un número de versión o fechas de aplicación, para registrar la secuencia de los cambios.
3	Se incluyen varias columnas para cada atributo cuyos cambios deben rastrearse, y cada valor nuevo se escribe en la siguiente columna disponible para el atributo. De manera natural, el historial está limitado a la cantidad de columnas incluidas.
4	Los datos actuales se conservan en una tabla, y se emplea una tabla de historial para registrar algunos o todos los valores de datos anteriores.

Encontrará más información sobre las dimensiones de cambio lento en numerosos artículos publicados en Internet.

Mercados de datos

Un *almacén de datos* es un subconjunto de un almacén de datos que da soporte a los requisitos de un departamento o una función empresarial específicos. En parte, los mercados de datos evolucionaron como respuesta a algunos fracasos de proyectos de almacén de datos multimillonarios muy destacados. Cuando una organización tiene poca experiencia en la creación de sistemas y bases de datos OLTP, o cuando los requisitos son muy superficiales, un proyecto a escala menor, como un almacén de datos, es un método mucho menos arriesgado. Éstas son algunas características de los mercados de datos:

- Se concentran en un departamento o proceso empresarial.
- Por lo general, no contienen datos operativos.
- Contienen mucha menos información que un almacén de datos.

Éstas son algunas razones para crear un mercado de datos:

- Los datos se ajustan a un departamento o función empresarial específica.
- Los costos generales son menores que los de un almacén de datos completo.
- El proyecto es menos arriesgado que un proyecto de almacén de datos completo.
- Una cantidad limitada de herramientas de análisis para usuario final (por lo general, sólo una) permite que los datos se ajusten a la herramienta especifica que se utiliza.
- En el caso de mercados de datos departamentales, la base de datos se ubica físicamente cerca del departamento, lo que reduce los retrasos que ocurren en una red.

Se emplean tres estrategias básicas para crear mercados de datos:

- *Crear primero el almacén de datos en el nivel de la empresa, y emplearlo para llenar los mercados de datos*. El problema con este método es que nunca conseguirá crear los mercados de datos si al final el proyecto de almacén de datos es cancelado o se mantiene en espera indefinida.
- *Crear varios mercados de datos y después el almacén de datos, para integrar los mercados al almacén en el nivel empresarial en ese momento*. Ésta es una estrategia de menor riesgo, cuando menos en cuanto a la entrega, porque no depende de la conclusión de un proyecto importante de almacén de datos. Sin embargo, puede costar más debido al trabajo repetido que se necesita para integrar los mercados de datos después del hecho. Además, si se crean varios mercados que contienen datos similares sin un almacén común para integrar todos los datos, la misma consulta puede producir resultados diferentes, dependiendo del mercado de datos utilizado. Por ejemplo, imagine que el Departamento de Finanzas menciona una cantidad de ingresos y el Departamento de Ventas otra, sólo para concluir que ambos citan correctamente sus orígenes de datos.

Pregunta al experto

P: ¿Los mercados de datos se crean mediante tablas de resumen o esquemas de estrella?

R: Los mercados de datos se crean casi exclusivamente mediante esquemas de estrella. Esto es lo más probable, porque casi todas las herramientas de análisis de usuario final populares esperan esquemas de estrella, como las tablas dinámicas permitidas por herramientas de hoja de cálculo como Microsoft Excel.

- *Crear el almacén de datos y los mercados de datos al mismo tiempo*. Esto se ve estupendo en el papel, pero cuando considera que el ya complejo y enorme proyecto de almacén de datos ahora incorpora los mercados de datos, se aprecia por primera vez la enormidad de la tarea. En realidad, esta estrategia prácticamente *garantiza* que el proyecto de almacén de datos será interminable por los siglos de los siglos.

Minería de datos

La *minería de datos* es el proceso de extraer información válida, pormenorizada y funcional, previamente desconocida, de bases de datos grandes, y de utilizarla para tomar decisiones empresariales cruciales. El mayor beneficio es que se pueden descubrir en los datos relaciones que nunca sospechó. La advertencia es que esto suele requerir volúmenes de datos muy grandes para producir resultados precisos. Casi todas las herramientas comerciales de inteligencia empresarial (BI, Business Intelligence)/OLAP incluyen algunas características de minería de datos.

Una de las historias más mencionadas de éxito inicial de la minería de datos se relaciona con un empleado de NCR Corporation, quien preparó un estudio para Osco Drugs, de American Stores, en 1992. El estudio observó una correlación entre las ventas de cerveza y las de pañales entre 5 y 7 P.M., lo que significaba que los dos artículos se encontraban juntos en una sola compra con más frecuencia de lo que sugeriría el azar. Esta relación fue mencionada después en un discurso, y la historia de "la cerveza y los pañales" rápidamente se convirtió en una especie de leyenda urbana en el círculo de los almacenes de datos. Incontables conferencistas han referido la historia de los papás jóvenes enviados por pañales que, al mismo tiempo, tomaron un paquete de latas de cerveza, y a menudo la han extendido más allá de los hechos. Sin embargo, esta historia sigue siendo un excelente ejemplo de cuán inesperados pueden ser los resultados de la minería de datos.

Una vez que descubre una relación, la organización debe decidir la mejor acción que debe aplicar para capitalizar la nueva información. En el ejemplo de "la cerveza y los pañales", la compañía pudo colocar estratégicamente el exhibidor de los pañales cerca de los refrigeradores con cervezas para que ocurriera esa rápida venta por impulso, o tal vez colocar distribuidores de cupones para la cerveza cerca del exhibidor de pañales, y ubicar estratégicamente

la cerveza y los pañales en esquinas opuestas de la tienda con la esperanza de conseguir más compras por impulso cuando el comprador tomaba un artículo y atravesaba la tienda por el otro. Para aprovechar la información recién encontrada, la organización debe ser lo suficientemente ágil para aplicar ciertas acciones, de modo que la minería de datos por sí sola no es de ningún modo una solución mágica.

Pruebe esto 12-1 Diseño de tablas de hechos y de dimensiones para esquemasde estrella

En este ejercicio diseñará un hecho para el esquema de estrella de la tabla LIBROS del esquema de la Compañía de Libros de Computación del ejercicio Prueba esto 6-2, junto con sus tablas de dimensiones asociadas. Para una consulta fácil, éstas son las tablas OLTP normalizadas que debe considerar:

```
LIBRO: ISBN (CP), TÍTULO DE LIBRO, CÓDIGO DE TEMA, ID DE EDITOR,
       CÓDIGO DE EDICIÓN, COSTO DE EDICIÓN, PRECIO DE VENTA,
       CANTIDAD EN EXISTENCIA, CANTIDAD PEDIDA,
       CANTIDAD RECOMENDADA, ISBN DE EDICIÓN ANTERIOR

TEMA: CÓDIGO DE TEMA (CP), DESCRIPCIÓN

AUTOR: ID DE AUTOR (CP), NOMBRE DE AUTOR

AUTOR DE LIBRO: ID DE AUTOR (CP), ISBN (CP)

EDITOR: ID DE EDITOR (CP), NOMBRE DE EDITOR, DIRECCIÓN DE CALLE,
        CIUDAD, ESTADO, CÓDIGO POSTAL, MONTO POR PAGAR
```

Paso a paso

1. Diseñe la tabla de hechos:

- **a.** Identifique los hechos que incluirá en su tabla de hechos. Para la tabla LIBRO, los únicos atributos que pueden ser hechos son COSTO DE EDICIÓN, PRECIO DE VENTA, CANTIDAD EN EXISTENCIAS, CANTIDAD PEDIDA Y CANTIDAD RECOMENDADA.
- **b.** Entre los atributos restantes en la tabla LIBRO, identifique los que son claves externas para las tablas de dimensiones. Éstos son CÓDIGO DE TEMA e ID DE EDITOR.
- **c.** Los atributos restantes son TÍTULO DE LIBRO e ISBN DE EDICIÓN ANTERIOR. ¿Qué puede hacer con ellos? Una opción es simplemente eliminarlos de su esquema de estrella, otra es convertirlos en una dimensión, llamada TÍTULO DE LIBRO. La

(*continúa*)

tabla de hechos puede combinarse con la dimensión mediante el ISBN cuando quiera incluir el título o el ISBN de una edición anterior en los resultados de la consulta.

d. Presente en una lista el contenido de la tabla de hechos.

2. Diseñe las tablas de dimensiones:

a. A partir del paso 1.c, diseñe una tabla de dimensiones que contenga TÍTULO DE LIBRO e ISBN DE EDICIÓN ANTERIOR.

b. TEMA se convierte en una dimensión tal como está.

c. AUTOR y AUTOR LIBRO plantean un pequeño desafío porque forman una jerarquía. Sin embargo, si los contrae en una sola tabla, forman una dimensión que presenta una lista de cada autor para cada libro. La tabla de dimensiones incluirá una violación a la segunda forma normal (NOMBRE DE AUTOR dependerá sólo de ID DE AUTOR), pero no necesita preocuparse por esto en los esquemas de estrella. En realidad, si no fuera por la posibilidad de dos autores diferentes que tengan el mismo nombre, podría eliminar por completo ID DE AUTOR de la dimensión.

d. EDITOR se ve bastante sencilla, pero existe un problema menor con MONTO POR PAGAR. Es un hecho, y los hechos no caben en las tablas de dimensiones. De modo que debe eliminarlo de este esquema de estrella. Tal vez sea útil cuando la tabla de hechos se refiera a las compras de una editorial o algo así, pero no tiene relación con este inventario de libros.

e. Presente una lista con el contenido de cada tabla.

Resumen de Pruebe esto

En este ejercicio diseñó una tabla de hechos y varias tablas de dimensiones. Mi solución aparece en el apéndice B.

✓ Autoexamen Capítulo 12

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. Las bases de datos OLTP están diseñadas para manejar volúmenes _____ de transacciones.

2. Las consultas OLAP suelen acceder a cantidades _____ de datos.

3. Comparados con los sistemas OLTP, los sistemas de almacén de datos tienden a tener consultas _____.

4. El pionero de los almacenes de datos fue _____.

5. Al proceso de pasar de datos más resumidos a datos más detallados se le conoce como _____.

6. El esquema de copo de nieve permite que las dimensiones tengan _____.

7. El esquema de copo de estrella es un híbrido que contiene dimensiones _____ y _____.

8. Un almacén de datos

A Está orientado a temas.

B Se integra a partir de varios orígenes de datos.

C Varía con el tiempo.

D Se actualiza en tiempo real.

E Se organiza alrededor de un departamento o una función empresarial.

9. Algunos desafíos del método de almacén de datos son

A La actualización de los datos operativos del almacén de datos.

B La subestimación de los recursos requeridos.

C La disminución de las demandas de los usuarios.

D Son proyectos grandes y complejos.

E Las demandas de recursos son altas.

10. La arquitectura de tabla de resumen

A Fue desarrollada originalmente por Bill Inmon.

B Incluye una tabla de hechos.

C Incluye tablas de dimensiones.

D Incluye tablas poco y muy resumidas.

E Debe incluir metadatos.

11. El esquema de estrella

A Fue desarrollado por Ralph Kimball.

B Incluye una tabla de dimensiones y una o más tablas de hechos.

C Siempre tiene tablas de dimensiones completamente normalizadas.

D Era una función clave del DBMS Red Brick.

E Incorpora varios niveles de tablas de dimensiones.

12. Algunos factores que deben considerarse cuando se diseña una tabla de hechos son

A Agregar columnas a la tabla de hechos.

B Reducir el tamaño de las columnas entre las tablas de origen y de hechos.

C Particionar la tabla de hechos.

D Con cuánta frecuencia debe actualizarse.

E Cuánto historial debe contener.

13. Las bases de datos multidimensionales

A Emplean una tabla de hechos completamente normalizada.

B Se visualizan mejor como cubos.

C Tienen tablas de dimensiones completamente normalizadas.

D En ocasiones se les denomina bases de datos MOLAP.

E Alojan más de tres dimensiones al repetir cubos para cada dimensión adicional.

14. Un mercado de datos

A Es un subconjunto de un almacén de datos.

B Es una tienda que vende datos a personas y empresas.

C Da soporte a los requisitos de un departamento o una función empresarial específicos.

D Puede ser un buen punto inicial para organizaciones sin experiencia en almacenes de datos.

E Puede ser un buen punto inicial cuando los requisitos son imprecisos.

15. Algunas razones para crear un mercado de datos son

A Es más pormenorizado que un almacén de datos.

B Es un proyecto con menores posibilidades de riesgo.

C Los datos se ajustan a un departamento o una función empresarial específicos.

D Contiene más datos que un almacén de datos.

E El proyecto tiene un costo general más bajo que un proyecto de almacén de datos.

16. Crear primero un almacén de datos, seguido por mercados de datos

- **A** Retrasará el despliegue de un mercado de datos si se vuelve lento el proyecto de almacén de datos.
- **B** Tiene un riesgo menor que tratar de crearlos juntos.
- **C** Tiene el menor riesgo de las tres estrategias posibles.
- **D** Tiene el mayor riesgo de las tres estrategias posibles.
- **E** Tal vez necesite mucha repetición de trabajo.

17. Crear primero uno o más mercados de datos, seguidos por el almacén de datos

- **A** Retrasará la entrega del almacén de datos si se vuelven lentos los proyectos de mercado de datos.
- **B** Tiene el potencial para entregar más rápido algunas funciones OLAP.
- **C** Tiene el menor riesgo de las tres estrategias posibles.
- **D** Tiene el mayor riesgo de las tres estrategias posibles.
- **E** Tal vez necesite mucha repetición de trabajo.

18. Crear el almacén de datos y los mercados de datos al mismo tiempo

- **A** Genera el proyecto único más grande de todas las estrategias posibles.
- **B** Tiene el potencial para tardar el máximo de tiempo para entregar cualquiera de las funciones OLAP.
- **C** Tiene el menor riesgo de las tres estrategias posibles.
- **D** Tiene el mayor riesgo de las tres estrategias posibles.
- **E** Tal vez necesite mucha repetición de trabajo.

19. La minería de datos

- **A** Crea un almacén de datos a escala menor.
- **B** Extrae relaciones de datos previamente desconocidas del almacén de datos.
- **C** Puede tener éxito con cantidades de datos pequeñas.
- **D** Es más útil cuando la organización es lo bastante ágil para aplicar una acción con base en la información.
- **E** Suele requerir grandes volúmenes de datos para producir resultados precisos.

20. Algunas propiedades de los sistemas de almacén de datos son

A Contienen información histórica en lugar de actual.

B Incluyen consultas de ejecución prolongada que procesan muchas filas de datos.

C Permiten las operaciones cotidianas.

D Están orientados a procesos.

E Su volumen de transacciones va de mediano a bajo.

Capítulo 13

Integración de documentos y objetos XML en bases de datos

Habilidades y conceptos clave

- Conozca los fundamentos de XML.
- Conozca SQL/XML.
- Aplicaciones orientadas a objetos.
- Bases de datos de objetos-relacionales.

Junto con el enorme crecimiento del uso de las bases de datos, sobre todo las relacionales, ha aumentado marcadamente la necesidad de guardar tipos de datos más complejos. En especial esto se aplica a las bases de datos que dan soporte a sitios Web que presentan imágenes y documentos formados, al igual que clips de audio y video. Además, conforme ha aumentado el uso de lenguajes de programación orientados a objetos, como C++ y Java, también ha crecido la necesidad de guardar los objetos que manipulan estos lenguajes. (Los objetos fueron presentados brevemente en el capítulo 1.) En este capítulo, se analizarán varias maneras de integrar ese contenido en las bases de datos.

Conozca los fundamentos de XML

El lenguaje extensible de marcado (eXtensible Markup Language, XML) es un lenguaje de marcado de propósito general que sirve para describir datos en un formato conveniente para despliegue en páginas Web y para intercambio de datos entre diferentes partes. En 2003, se agregó (como la parte 14, llamada SQL/XML) a la norma SQL de ANSI/ISO, la especificación para guardar datos XML en bases de datos de SQL (relacionales). En 2006, la parte 14 se amplió aún más.

NOTA

SQL/XML no es lo mismo que SQLXML de Microsoft, que es una tecnología patentada que se utiliza en SQL Server. Como imaginará, estos nombres desafortunadamente similares han causado mucha confusión. Microsoft participó en los procesos de las normas para SQL/XML, pero después optó por no implementarlas.

Para comprender SQL/XML, primero debe entender los fundamentos de XML. Si bien una explicación completa de XML está mucho más allá del alcance de este libro, se ofrece un compendio breve. Encontrará mucha información adicional en una búsqueda en Internet.

Tal vez ya esté familiarizado con HTML, el lenguaje de marcado utilizado para definir páginas Web. De ser así, la sintaxis de XML le parecerá similar. Esto se debe a que ambos están basados en el lenguaje estándar de marcado generalizado (SGML, Standard Generali-

zad Markup Language), que a su vez se basó en el lenguaje de marcado generalizado (GML, Generalizad Markup Language), desarrollado por IBM en la década de 1960. Un *lenguaje de marcado* es un conjunto de anotaciones, a las que suele llamárseles *etiquetas*, que son utilizadas para describir cómo se va a estructurar, formar o colocar el texto. El texto etiquetado está diseñado para que las personas lo lean. Una de las diferencias fundamentales entre HTML y XML es que el primero ofrece un conjunto predefinido de etiquetas, mientras que el segundo permite al autor crear sus propias etiquetas.

A continuación se presenta un documento de XML de ejemplo que contiene los resultados de una consulta de SQL. En la figura 13-1 se presenta una tabla DEPARTAMENTO que contiene dos departamentos y una tabla CURSO que contiene cinco cursos educativos ofrecidos por esos departamentos. Como aprendió en el capítulo 4, las dos tablas se combinan fácilmente mediante una instrucción SELECT de SQL, como ésta:

```
SELECT a.NOMBRE_DEPTO, b.TITULO_DE_CURSO, b.ID_DE_CURSO
  FROM DEPARTAMENTO a JOIN CURSO b
       ON a.ID_DEPTO = b.ID_DEPTO
 ORDER BY a.NOMBRE_DEPTO, b.TITULO_DE_CURSO,
```

Observe que se utilizó la cláusula ORDER BY para especificar el orden de las filas en el grupo de resultados. Los resultados de la consulta deben verse así:

```
NOMBRE_DEPTO                 TITULO_DE_CURSO                 ID_DE_CURSO
---------------------------  ------------------------------  -----------
Negocios                     Introducción a la contabilidad  101
Negocios                     Conceptos de mercadotecnia      102
Tecnología de la información Programación en C I             401
Tecnología de la información Programación en C II            402
Tecnología de la información Intro. sistemas computacionales 400
```

Los resultados de la consulta están bien organizados para despliegue o impresión, pero no están en una forma que sea fácil de mostrar en una página Web o de transferir a otra aplica-

DEPARTAMENTO

ID_DEPTO	NOMBRE_DE_DEPARTAMENTO
NEG	Negocios
TI	Tecnologia de la informacion

CURSO

ID_DE_CURSO	TITULO DE CURSO	ID_DEPTO
101	Introduccion a la contabilidad	NEG
102	Conceptos de mercadotecnia	NEG
400	Intro. sistemas computacionales	TI
401	Programacion en C I	TI
402	Programacion en C II	TI

Figura 13-1 Las tablas DEPARTAMENTO y CURSO.

ción de computadora para un procesamiento adicional. Un modo de facilitar esto consiste en convertir a XML los resultados de la consulta, como se presenta aquí:

```
<departamentos>
   <departamento name="Negocios">
      <cursos>
         <curso title="Introduccion a la contabilidad"><id>101</id>
         </curso>
         <curso title="Conceptos de mercadotecnia">
            <id>102</id></curso>
      </cursos>
   </departamento>
   <departamento name="Tecnologia de la informacion">
      <cursos>
         <curso title="Programacion en C I"><id>401</id></curso>
         <curso title="Programacion en C II"><id>402</id></curso>
         <curso title="Intro. sistemas computacionales">
            <id>400</id></curso>
      </cursos>
   </departamento>
   <!-- Pronto existirán departamentos adicionales -->
</departamentos>
```

Como puede ver en la lista de código, las etiquetas están entre paréntesis angulares, y cada etiqueta inicial tiene una etiqueta final correspondiente que es idéntica, excepto que se incluye una diagonal (/). (El HTML emplea una convención idéntica, pero es mucho más permisivo si hace algo como omitir una etiqueta final.) Por ejemplo, la etiqueta **<departmentos>** inicia la lista de departamentos académicos, la que concluye con la etiqueta final **</departmentos>**. Con la lista de departamentos, la información para cada departamento individual comienza con la etiqueta **<departmento>**, que incluye un valor para el atributo de nombre, y concluye con la etiqueta **</departmento>**. Se acostumbra (y se considera una buena práctica) asignar nombre a una lista usando el plural del nombre de etiqueta utilizado para cada concepto de la lista. Se agregan comentarios mediante un etiqueta especial que comienza con **<!--** y concluye con **-->**, como se aprecia en la penúltima línea del ejemplo. (Un último tema es que el uso de acentos y caracteres especiales requiere un tratamiento especial que está más allá del alcance de este libro, por lo que se opta por omitirlos.)

Los conceptos y los valores de datos, como los que se guardarían en una columna de una tabla relacional, se pueden codificar de dos maneras: como pares de nombre y valor. La primera es utilizar un *atributo* de XML, al nombrarlo dentro de otra etiqueta, seguido por el signo de igual y el valor del atributo entre comillas dobles, como se hizo con los atributos del nombre y el título. La segunda es emplear un *elemento* de XML, al crear un etiqueta separada para el concepto de los datos, con el valor de los datos en medio de las etiquetas inicial y final, como se hizo con el atributo **id** dentro de la etiqueta **curso**. La pregunta relacionada con cuál forma utilizar ha sido tema de gran debate entre los desarrolladores de XML. No obstan-

te, el consenso general es emplear los elementos cuando el concepto de los datos puede después dividirse en elementos adicionales, como separar el nombre de una persona en nombre y apellido, o dividir en una lista de elementos un elemento de datos único que contiene una lista separada por comas de nombres de cursos propedéuticos. Una consideración adicional es si prefiere permitir que el procesador de XML ignore los espacios en blanco no significativos, como lo haría con los atributos, pero no con los elementos.

Tal vez haya observado que, a diferencia del grupo de resultados de SQL, XML puede mostrar la jerarquía de los datos. En este caso, la lista de cursos ofrecidos por cada departamento se anida dentro de la información sobre el departamento. Se han incorporado sangrías en las instrucciones XML para hacer más obvio el anidamiento. Y aunque poner sangrías en las etiquetas anidadas es una buena práctica, no tiene un efecto secundario, porque los espacios en blanco entre las etiquetas son ignorados cuando se procesa el XML.

La codificación XML puede ser muy tediosa. Por suerte, existen herramientas que ayudan a convertir entre XML y el texto simple, y funciones de SQL/XML (que se cubren más adelante el capítulo) para convertir los datos de una base de datos relacional a XML. Durante

Pregunta al experto

P: **¿Existe una norma para el lenguaje XML mismo?**

R: Si bien en la actualidad ISO lista una norma para XML, ISO 8879 proporciona una norma para SGML, y el XML está basado en SGML. Y, lo que es más importante, el World Wide Web Consortium (W3C) publica especificaciones de XML que forman la norma que suele aceptarse en toda la industria de la tecnología de la información.

P: **Mencionó que XML es una manera conveniente para que diferentes partes intercambien información. ¿Eso significa que dos compañías pueden intercambiar datos libremente sin tener que crear un complicado software de interfaz, siempre y cuando ambas empleen XML?**

R: Bueno, no exactamente. El XML sólo proporciona una manera estándar para formar los datos. Para que una compañía interprete correctamente los datos de XML que le ha enviado otra empresa, la receptora debe conocer los nombres y las definiciones de las etiquetas que la empresa remitente formó para ellos, sobre todo los elementos y los atributos que contienen los datos. Por suerte, varias normas de la industria pueden ser útiles. Por ejemplo, HR/XML proporciona una norma para intercambiar datos de recursos humanos (HERRAMIENTA, Human Resources), de modo que una compañía puede, por ejemplo, enviar los datos de sus empleados a un vendedor que les proporciona un seguro médico. En algunas industrias, XML comienza a reemplazar una norma más antigua conocida como intercambio electrónico de datos (EDI, Electronic Data Interchange).

un tiempo, se volvieron populares las bases de datos especializadas para guardar y recuperar XML, pero los principales vendedores de bases de datos relacionales agregaron funciones para permitir que el XML nativo se guardara directamente en sus bases de datos. Al mismo tiempo, la norma de SQL se expandió con el fin de incluir provisiones para los datos XML, tal como se analiza en las siguientes secciones del capítulo.

Conozca SQL/XML

Como se mencionó, XML suele utilizarse para representar los datos en páginas Web, y esos datos suelen provenir de bases de datos relacionales. Sin embargo, como ha visto, los dos modelos en uso son muy diferentes, porque los datos relacionales se guardan en tablas en que no tienen importancia una jerarquía ni una secuencia, mientras que XML se basa en árboles jerárquicos en que el orden se considera importante. Se suele emplear el término *bosque* para referirse a un conjunto de estructuras de árbol de XML. El XML se emplea para páginas Web porque su estructura se parece mucho a la que se usaría para mostrar los mismos datos en HTML. En realidad, muchas páginas Web son una mezcla de HTML para las funciones estáticas y de XML para los datos dinámicos. Tal vez sea esta implementación tan difundida lo que ha provocado que muchos de los principales vendedores, entre ellos Oracle, Microsoft e IBM, permitan extensiones XML. Sin embargo, sólo Oracle y DB2 UDB de IBM permiten los comandos de SQL/XML cubiertos en este tema: la extensión XML de Microsoft SQL Server es marcadamente diferente y no se ha incluido en este libro porque está patentada.

SQL/XML se divide en tres partes principales: el tipo de datos XML, las funciones de SQL/XML y las reglas de conversión de SQL/XML. Se desarrollan cada una de ellas como temas principales del resto del capítulo.

Tipo de datos XML

El tipo de datos XML es manejado de la misma manera general que todos los otros tipos de datos analizados en el capítulo 2. Guardar datos en el formato de XML directamente en la base de datos no es el único modo de usar SQL y XML juntos. Sin embargo, es una manera muy sencilla de iniciar, porque se trata de una extensión lógica de las primeras implementaciones, en que los desarrolladores de SQL sólo guardaban el texto XML en una columna definida con un tipo de datos de caracteres generales, como CHARACTER VARYING (VARCHAR), es decir, caracteres diversos. Es mucho mejor indicar al DBMS que la columna contiene XML, y el modo específico en que se codifica XML, para que el DBMS pueda proporcionar funciones adicionales adaptadas para el formato XML.

La especificación para el tipo de datos XML tiene este formato general:

XML (<modificador de tipo> {(<modificador de tipo secundario>)})

Se requiere el modificador de tipo y debe estar entre paréntesis, tal como se muestra, mientras que el modificador de tipo secundario es opcional, y en realidad no es permitido por todos

los modificadores de tipo. La norma no es específica acerca de cómo una implementación del SQL en particular debe tratar los diversos tipos, pero se especifican algunas convenciones y reglas de sintaxis. Éstos son los modificadores de tipo válidos:

- **DOCUMENT** El tipo DOCUMENT está diseñado para guardar documentos de texto formateados mediante XML. En general, se espera que los valores de datos estén formados por caracteres que puedan leer las personas, como letras, números y símbolos, tal como aparecerían en un documento de texto no estructurado.
- **CONTENT** El tipo CONTENT está contemplado para datos más complejos, entre los que se incluyen datos binarios como imágenes y segmentos de sonido.
- **SEQUENCE** El tipo SEQUENCE está diseñado para documentos de XQuery, a los que suele denominárseles secuencia de XQuery. XQuery es un tema avanzado que está más allá del alcance de este libro.

El modificador de tipo secundario, que se utiliza sólo con los modificadores de tipo primario DOCUMENT y CONTENT, puede tener uno de estos valores:

- **UNTYPED** Los datos de XML no son de un tipo específico.
- **ANY** Los datos de XML son de cualquiera de los tipos permitidos por la implementación de SQL.
- **XMLSCHEMA** El tipo XMLSCHEMA hace referencia a un esquema registrado de XML, del qque se ha informado al servidor de la base de datos. Los tres más comunes se presentan en la tabla siguiente:

Prefijo común	Identificador uniforme de recursos (URI) de nombre de espacio objetivo
Xs	www.w3.org/2001/XMLSchema
Xsi	www.w3.org/2001/XMLSchema-instance
Sqlxml	standards.iso.org/iso/9075/2003/sqlxml

Para implementaciones de SQL que no permiten el modificador de tipo secundario, se supone que ANY es un valor predeterminado.

NOTA

Debido a que SQL/XML es una norma relativamente nueva, varía el soporte para cada implementación de vendedor. Oracle permite un tipo de datos XMLType en lugar del tipo XML, pero se aplica en el nivel de tabla, de modo que la tabla completa se guarda como XML. DB2 UDB de IBM permite un tipo XML, pero sin los modificadores de tipos. Como ya se mencionó, Microsoft SQL Server permite XML y un tipo de datos XML, pero de una manera un tanto distinta de la norma de SQL/XML. En cuanto a MySQL versión 5.0, no ofrece soporte a XML, pero se espera que sea incluido en un lanzamiento futuro.

Suponga que se requiere agregar el programa de estudios del curso a la tabla de cursos que se muestra en una página Web. Si un programa de estudios puede provenir de varias fuentes diferentes y, por lo tanto, tiene formatos distintos, dependiendo de la fuente, XML puede ser una buena manera para guardar los datos en la tabla de cursos. En el ejemplo siguiente, se agregó la columna a la definición de la tabla CURSO que aparece en la figura 13-1:

```
CREATE TABLE          CURSO
( ID_DE_CURSO         INT,
  TITULO_DE_CURSO     VARCHAR(60),
  ID_DEPTO            CHAR(3)
  PROGRAMA_DE_CURSO   XML(DOCUMENT(UNTYPED) );
```

NOTA

Aunque la norma de SQL de ISO/ANSI especifica un tipo de datos XML en la forma que se presenta aquí, ninguna implementación importante de SQL parece permitir esta sintaxis. Sin embargo, la norma es bastante nueva, de modo que se espera que esta sintaxis sea permitida en un futuro cercano.

Funciones de SQL/XML

Una función de SQL/XML (también llamada función de valor de XML) es simplemente una función que devuelve un valor de tipo XML. Por ejemplo, es posible escribir una consulta que seleccione datos que no son de XML (es decir, datos guardados en tipos de datos diferentes a XML) y formar los resultados de la consulta en un XML adaptado para incluirse en un documento de XML que se pueda mostrar en una página Web o transmitirse a otra persona. En otras palabras, SQL/XML no siempre forma documentos completos; en ocasiones se deben incorporar elementos adicionales para convertir el XML devuelto por el DBMS en un documento completo. En la tabla 13-1 se presentan las funciones básicas de SQL/XML.

Existen más combinaciones que las presentadas aquí, y todas estas funciones de SQL/XML se pueden emplear en combinaciones para formar consultas muy poderosas (aparte de complicadas). Asimismo, las funciones disponibles varían entre las implementaciones de SQL. Observe un ejemplo sencillo para aclarar cómo se utilizan estas funciones. En este ejemplo se presentan los cursos para el departamento Negocios mediante las tablas DEPARTAMENTO y CURSO mostradas en la figura 13-1. Ésta es la instrucción SQL, que emplea las funciones XMLELEMENT y XMLFOREST:

```
SELECT XMLELEMENT ("DepartamentoCurso",
       XMLFOREST (a.NOMBRE_DEPTO as departamento, a.ID_DEPTO,
                  b.ID_DE_CURSO, b.TITULO_DE_CURSO))
  FROM DEPARTAMENTO a JOIN CURSO b
       ON a.ID_DEPTO = b.ID_DEPTO
 WHERE a.ID_DEPTO = 'Neg'
 ORDER BY b.ID_CURSO,
```

Función	Valor devuelto
XMLAGG	Un valor único de XML que contiene un bosque de XML formado al combinar (totalizar) un conjunto de filas en que cada una contiene un valor único de XML.
XMLATTRIBUTES	Un atributo en la forma nombre=valor dentro de un XMLELEMENT.
XMLCOMMENT	Un comentario en XML.
XMLCONCAT	Una lista unidad de valores de XML, que crea un valor único que contiene un bosque de XML.
XMLDOCUMENT	Un valor de XML que contiene un solo nodo de documento.
XMLELEMENT	Un elemento de XML, que puede ser secundario de un nodo de documento, con el nombre especificado en el parámetro de nombre.
XMLFOREST	Un elemento de XML que contiene una secuencia de elementos XML formada a partir de las columnas de una tabla, y que usa el nombre de cada columna como el nombre correspondiente del elemento.
XMLPARSE	Un valor de XML formado al examinar la sintaxis de la cadena suministrada sin validarla.
XMLPI	Un valor de XML que contiene una instrucción de procesamiento de XML.
XMLQUERY	El resultado de una expresión de XQuery (XQuery es un lenguaje secundario utilizado para buscar el XML guardado en la base de datos; está más allá del alcance de este libro).
XMLTEXT	Un valor de XML que contiene un solo nodo de texto XML, que puede ser un elemento secundarios de un nodo de documento.
XMLVALIDATE	Una secuencia de XML que es resultado de validar un valor de XML.

Tabla 13-1 Funciones de SQL/XML.

Los resultados devueltos deben verse así:

```
<DepartamentoCurso>
  <Departamento>Negocios</Departamento>
  <ID_DEPTO>NEG</ID_DEPTO>
  <ID_CURSO>101</ID_CURSO>
  <TITULO_DE_CURSO>Introduccion a la contabilidad</TITULO_DE_CURSO>
</DepartamentoCurso>
<DepartamentoCurso>
<Departamento>Negocios</Departamento>
  <ID_DEPTO>NEG</ID_DEPTO>
  <ID_CURSO>102</ID_CURSO>
  <TITULO_DE_CURSO>Conceptos de mercadotecnia</TITULO_DE_CURSO>
</DepartamentoCurso>
```

Observe que los nombres de elementos XML se toman de los nombres de columnas, en mayúsculas separadas con guiones bajos, tal como se acostumbra en SQL. Sin embargo, si utiliza un alias de columna, como se hizo con la columna NOMBRE_DEPTO, puede modificar los nombres de columna a cualquier cosa que prefiera. Recuerde que el grupo de resul-

tados no es necesariamente un documento completo (un desarrollador de XML diría que el XML tal vez no esté "bien formado"). Para convertir el XML del último ejemplo en un documento completo, cuando menos se requiere un elemento de raíz, junto con su correspondiente etiqueta final. Si agregara el elemento **<DepartamentoCursos>** al inicio de los resultados y **</DepartamentoCursos>** al final de los resultados, se obtendría un documento bien formado.

Regla de conversión de SQL/XML

Hasta este momento no se ha analizado cómo los valores del SQL se traducen y representan como valores de XML y viceversa. La norma del SQL describe con detalle la manera en que los valores de SQL se pueden convertir en valores de XML, y viceversa. A continuación se presenta un compendio de las reglas de conversión del SQL/XML.

Conversiones de SQL a XML

Las conversiones en este tema se aplican para traducir datos en los tipos de datos de SQL a XML.

Conversión de grupos de caracteres de SQL a Unicode *Unicode* es una norma de la industria que permite a los sistemas de computadoras representar de manera uniforme (codificar) los caracteres de texto expresados en casi todos los lenguajes escritos en el mundo. El XML se suele codificar como caracteres de Unicode para permitir texto en varios lenguajes. Los datos de los caracteres de SQL se guardan en cualquier grupo de caracteres que se especifica cuando se crea la tabla con la base de datos, y mientras casi todas las implementaciones de SQL permiten Unicode, también se pueden emplear muchos otros grupos de caracteres. La norma de SQL requiere que cada carácter de un conjunto de caracteres de SQL tenga una conversión hacia un carácter de Unicode equivalente.

Conversión de grupos de caracteres de SQL a Unicode Es necesario definir una conversión de identificadores de SQL, como los nombres de tablas y columnas, a nombres de XML, porque no todos los identificadores de SQL son nombres de XML aceptables. Los caracteres que no son válidos en los nombres de XML se convierten a una secuencia de dígitos hexadecimales derivada de la codificación de Unicode del carácter, encerrada entre un guión bajo y una x introductorios y un guión bajo final. Por ejemplo, el signo de dos puntos (:) en un identificador de SQL se puede traducir a _x003A_ en un nombre de XML.

Conversión de tipos de datos de SQL a tipos de datos de esquema XML Tal vez ésta sea la más complicada de las formas de conversión. Para cada tipo dominio de SQL, se requiere la implementación de SQL para que proporcione una conversión al tipo de esquema de XML adecuado. Las conversiones detalladas de los tipos de SQL comunes a tipos de datos de esquema de XML se presentan en la norma con detalles minuciosos. Aquí se resumen en la tabla 13-2.

Tipo de SQL	Tipo de esquema XML	Notas
CHARACTER, CHARACTER VARYING, CHARACTER LARGE OBJECT	xs:string	Se usa la faceta xs:lenght de XML para especificar la longitud de las cadenas de longitud fija. (Una faceta es un elemento que sirve para definir una propiedad de otro elemento.)
NUMERIC DECIMAL	xs:decimal	La precisión y escala se especifican mediante las facetas xs:precision y xs:scale de XML.
INTEGER SMALLINT BIGINT	xs:integer	Esta conversión se presenta como definida por la implementación, lo que significa que es opcional.
FLOAT REAL DOUBLE PRECISION	xs:float; xs:double	Para una precisión hasta de 24 dígitos binarios (bits) y un exponente entre -149 y 104 inclusive, se usa xs:float; de lo contrario se aplica xs:double.
BOOLEAN	xs:Boolean	
DATE	xs:date	Se usa la faceta xs:pattern para excluir el uso de un desplazamiento de zona horaria.
TIME WITH TIME ZONE TIME WITHOUT TIME ZONE	xs:time	Se usa la faceta xs:pattern para excluir o especificar el uso de un desplazamiento de zona horaria, según sea adecuado.
TIMESTAMP WITH TIME ZONE; TIMESTAMP WITHOUT TIME ZONE	xs:dateTime	Se usa la faceta xs:pattern para excluir o especificar el uso de un desplazamiento de zona horaria, según sea adecuado.
Tipos de intervalos	xdt:yearMonthDuration, xdt:dayTimeDuration	
Tipo de fila	Tipo complejo de esquema de XML	El documento de XML contiene un elemento para cada campo del tipo de fila de XMLSQL.
Dominio	Tipo de datos de esquema de XML	El tipo de datos de dominio se convierte a XML con una anotación que identifica el nombre del dominio.
Tipo distintivo de SQL	Tipo simple de esquema de XML	
Tipo de colección de SQL	Tipo complejo de esquema de XML	El tipo complejo tiene un solo elemento llamado element.
Tipo de XML	Tipo complejo de esquema de XML	

Tabla 13-2 Conversión de los tipos de datos de SQL a los tipos de esquema XML.

Conversión de valores de tipos de datos de SQL a valores de tipos de datos de esquema XML Para cada tipo o dominio de SQL, con excepción de los tipos estructurados y los de referencia, hay una conversión de los valores del tipo al valor del espacio de esquema XML correspondiente. Los valores nulos se representan mediante una ausencia (pasar por alto el elemento) o al usar la faceta **xsi:nil="true"** para establecer de manera explícita el valor nulo.

Conversión de una tabla de SQL a un documento XML y un documento de esquema XML La norma de SQL define una conversión de una tabla de SQL a uno o dos de estos documentos: un documento de esquema XML que describe la estructura de XML convertido, y un documento de XML o una secuencia de elementos de XML. Esta conversión se aplica sólo a las tablas base y a las tablas ya vistas, y sólo se pueden convertir las columnas visibles para el usuario de la base de datos. La implementación puede ofrecer opciones para lo siguiente:

- Si se convierte la tabla a una secuencia de elementos de XML o como un documento de XML con un solo nombre de raíz derivado del nombre de la tabla.
- El espacio de nombre de destino del esquema XML que se va a convertir.
- Si se van a convertir los valores nulos como elementos ausentes o como elementos marcados con la faceta **xsi:nil="true"**.
- Si se va a convertir la tabla a datos de XML, un documento de esquema XML, o ambos.

Conversión de un esquema de SQL a un documento de XML y un documento de esquema XML La norma de SQL define la conversión entre las tablas de un esquema de SQL y ya sea un documento de XML que representa los datos en las tablas, un documento de esquema XML, o ambos. Sólo se convierten las tablas y las columnas visibles para el usuario de la base de datos. La implementación puede ofrecer opciones para lo siguiente:

- Si se convierte la tabla como una secuencia de elementos de XML o como un documento de XML con un solo nombre de raíz derivado del nombre de la tabla.
- El espacio de nombre de destino del esquema XML que se va a convertir.
- Si se van a convertir los valores nulos como elementos ausentes o marcados con la faceta **xsi:nil="true"**.
- Si se va a convertir el esquema a datos de XML, un documento de esquema XML, o ambos.

Conversión de un catálogo de SQL a un documento de XML y un documento de esquema XML La norma de SQL define la conversión entre las tablas de un catálogo de SQL y ya sea un documento de XML que representa los datos en las tablas del catálogo, un documento de esquema XML, o ambos. Sin embargo, esta parte de la norma no especifica la

sintaxis para invocar dicha conversión porque está diseñada para que la utilicen las aplicaciones o para que otras normas hagan referencia a ella. Sólo se convierten los esquemas visibles para el usuario de SQL. La implementación puede ofrecer opciones para lo siguiente:

- Si se convierte cada tabla como una secuencia de elementos de XML o como un documento de XML con un solo nombre de raíz derivado del nombre de la tabla.
- El espacio de nombre objetivo del esquema y los datos de XML que se van a convertir.
- Si se van a convertir los valores nulos como elementos ausentes o como elementos marcados con la faceta **xsi:nil="true"**.
- Si se va a convertir el catálogo a datos de XML, un documento de esquema XML, o ambos.

Conversión de XML a SQL Este tema contiene dos conversiones de XML a SQL.

Conversión de Unicode a grupos de caracteres de SQL Al igual que con la conversión de los grupos de caracteres de SQL a Unicode, la norma de SQL requiere que exista una conversión de los caracteres de Unicode definida por la implementación a los caracteres en cada grupo de caracteres de SQL permitido por la implementación de SQL.

Conversión de nombres de XML a identificadores de SQL Esto es lo opuesto a la conversión de identificadores de SQL a nombres de XML, en donde los caracteres que fueron convertidos porque no eran válidos en los nombres de XML son devueltos a su forma original. De modo que, si un signo de dos puntos en un identificador de SQL fue convertido a _x003A_ cuando el identificador de SQL se tradujo a XML, se devolvería a un signo de dos puntos cuando se invirtiera el proceso. La norma de SQL recomienda además que la implementación de SQL utilice un algoritmo único para la traducción en ambas direcciones.

Pruebe esto 13-1 Uso de las funciones de SQL/XML

En este ejercicio utilizará las funciones de XML para seleccionar datos con un formato XML del esquema de ejemplo HR de Oracle utilizado en el capítulo 4. Es obvio que si optó por utilizar un RDBMS diferente, su implementación de SQL tiene que proporcionar soporte a XML para que concluya el ejercicio y, como de costumbre, tal vez tenga que modificar el código incluido en este ejercicio para ejecutarlo en su DBMS. Al momento de escribir este libro, Oracle y DB2 UDB eran los únicos otros DBMS que permitían SQL/XML. En el caso de SQL Server, se requiere cierta recodificación para usar la cláusula FOR XML patentada de Microsoft en lugar de las funciones de SQL/XML. Puede descargar el archivo Pruebe_esto_13.txt del sitio Web (consulte los detalles en el apéndice B), que contiene no sólo la instrucción de SQL utilizada en este ejercicio (con una instrucción alterna para usar con

(*continúa*)

SQL Server), sino también las instrucciones requeridas para crear la tabla EMPLEADOS y llenarla con los datos necesarios para este ejercicio (en caso que no utilice el esquema de ejemplo HR de Oracle).

Paso a paso

1. Abra la aplicación cliente de su RDBMS.
2. Si emplea la base de datos de Oracle que ya tiene instalado el esquema de ejemplo HR, haga lo siguiente para crear un esquema con la tabla EMPLEADOS y los datos necesarios para concluir este ejercicio:
   a. Si utiliza Oracle, cree un usuario llamado HR (esto creará un esquema con el mismo nombre). Consulte la documentación de Oracle si no sabe cómo hacerlo. Observe que muchas de las herramientas cliente de GUI, como SQL Developer, ya contienen funciones para crear usuarios nuevos.
   b. Si emplea SQL Server o DB2, prepare una base de datos llamada HR. (En estos productos, una base de datos es el equivalente lógico de un esquema en Oracle.) Consulte la documentación del vendedor si necesita ayuda con este paso.
   c. Conéctese al esquema (o base de datos) que acaba de crear. Muchas de las herramientas GUI ofrecen un sencillo menú desplegable con los esquemas disponibles para este propósito.
   d. Copie y pegue la instrucción CREATE TABLE y las tres instrucciones INSERT del archivo Pruebe_esto_13.txt en su cliente de SQL y ejecútelas como una secuencia de comandos.
3. Si todavía no lo ha hecho, conéctese al esquema HR (Oracle) o base de datos (SQL Server, DB2 y otros).
4. Va a crear una consulta de SQL que utilice tres funciones de SQL/XML para formar el XML que contiene un elemento para cada empleado del departamento 90 de la tabla EMPLEADOS. Cada elemento incluirá la identificación del empleado, seguida por elementos separados que contienen el nombre, el apellido y el número telefónico del empleado. Introduzca y ejecute la instrucción siguiente (o cópiela y péguela del archivo Pruebe_esto_13.txt). Para SQL Server, el archivo Pruebe_esto_13.txt contiene una versión alterna que incluye la cláusula FOR XML patentada de Microsoft.

```
SELECT XMLELEMENT("Empleado",
       XMLATTRIBUTES(ID_DE_EMPLEADO AS ID),
       XMLFOREST(NOMBRE AS "Nombre",
                 APELLIDO AS "Apellido",
                 NUMERO_TELEFONICO AS "Tel."))
```

```
 FROM EMPLEADOS
WHERE ID_DE_DEPARTAMENTO = 90
ORDER BY ID_DE_EMPLEADO;
```

5. El resultado obtenido debe parecerse al que aparece a continuación. Observe que el XML de cada empleado se genera como una línea única del grupo de resultados (los saltos de línea y las sangrías fueron agregados para que los resultados se comprendieran mejor; al mismo tiempo, se omitieron acentos, para evitar temas de conversión que están más allá del alcance de este libro).

```
<Empleado ID=”100”>
   <Nombre>Juan</Nombre>
   <Apellido>Campos</Apellido>
   <Tel.>515.123.4567</Tel.>
</Empleado>
   <Empleado ID=”101”>
   <Nombre>Lilia</Nombre>
   <Apellidos>Corona</Apellidos>
   <Tel.>515.123.4568</Tel.>
</Empleado>
   <Empleado ID=”102”>
   <Nombre>Luis</Nombre>
   <Apellidos>Arana</Apellidos>
   <Tel.>515.123.4569</Tel.>
</Empleado
```

6. Cierre la aplicación cliente.

Resumen de Pruebe esto

En este ejercicio, la instrucción SELECT de SQL utilizó tres funciones de SQL/XML para formar datos de la tabla EMPLEADOS a XML. Se empleó la función XMLELEMENT con el fin de crear un elemento para cada empleado. Se aplicó la función XMLATTRIBUTES para incluir el valor de ID_DE_EMPLEADO con la identificación de nombre como valor dentro del elemento Empleado. Por último, se utilizó la función XMLFOREST con el fin de crear elementos para las columnas NOMBRE, APELLIDO y NUMERO_TELEFONICO.

Aplicaciones orientadas a objetos

En esta sección se supone que ya ha leído y comprendido la sección “El modelo orientado a objetos” del capítulo 1. Tal vez quiera repasarla antes de continuar.

Las aplicaciones orientadas a objetos se escriben en un lenguaje de programación orientado a objetos. Estos lenguajes suelen venir con una estructura predefinida de clases de objetos

y métodos predefinidos, pero es evidente que los desarrolladores pueden crear clases y métodos propios. Algunos vienen con un ambiente de desarrollo completo que incluye no sólo los elementos de lenguaje, sino también una base de datos orientada a objetos integrada. Es importante que comprenda que las aplicaciones orientadas a objetos se pueden crear sin una base de datos orientada a objetos, y este tipo de base de datos puede existir (al menos en teoría) sin una aplicación orientada a objetos para acceder a ella.

Programación orientada a objetos

La programación orientada a objetos emplea *mensajes* como vehículo para la interacción de los objetos. Un *mensaje*, en el contexto orientado a objetos, está formado por el identificador del objeto que va a recibir el mensaje, el nombre del método que se va a invocar mediante el objeto que se recibe y, de manera opcional, uno o más parámetros. Recuerde, del capítulo 1, que un *método* es un segmento de lógica de un programa de aplicaciones que funciona sobre un objeto específico y proporciona una función finita. La noción de que todos los accesos a las variables de un objeto se hacen a través de sus métodos resulta esencial para el paradigma orientado a objetos. Por lo tanto, la programación orientada a objetos incluye la escritura de los métodos que abarquen el *comportamiento* del objeto (es decir, lo que hace) y preparar mensajes dentro de esos métodos cada vez que un objeto deba interactuar con otros. El desarrollo de una aplicación orientada a objetos incluye el diseño de objetos y clases, además de las tareas de programación ya mencionadas.

El paradigma orientado a objetos también permite objetos *complejos*, que están formados por uno o más objetos adicionales. Por lo general, esto se implementa mediante una *referencia* a objeto, en donde un objeto contiene el identificador de uno o más objetos adicionales. Por ejemplo, un objeto Cliente puede contener una lista de objetos Pedido que el cliente ha hecho, y cada objeto Pedido puede contener el identificador del cliente que hizo el pedido. Al identificador único de un objeto se le denomina *identificador de objeto* (OID, Object IDentifier), y un valor se asigna automáticamente a cada objeto conforme se crea y después es invariable (es decir, el valor nunca cambia).

Lenguajes orientados a objetos

A continuación se revisan tres de los lenguajes de programación orientada a objetos más populares: Smalltalk, C++ y Java.

Smalltalk

El sistema orientado a objetos pionero fue Smalltalk, y fue desarrollado en 1972 por Software Concepts Group, en el Xerox Palo Alto Research Center (PARC), dirigido por Alan Kay. Fue Kay quien acuñó el término “orientado a objetos”. Smalltalk incluye un lenguaje, un ambiente de programación, un “sistema de archivos de imágenes” para guardar objetos y métodos (más o menos, una base de datos) y una extensa biblioteca de objetos. Entre las innovaciones de

Smalltalk están un despliegue de mapa de bits, un sistema de ventanas y el uso de un ratón. En un interesante giro de la historia, Xerox financió y fue propietaria del primer ambiente de programación orientado a objetos comercial, el sistema de ventanas original, el ratón y muchas otras innovaciones técnicas de las computadoras. Sin embargo, Xerox nunca determinó cómo comercializarlas, de modo que, con el tiempo, las innovaciones de la empresa cayeron en otras manos y en algún momento fueron "introducidas" al mercado por otras compañías. Aunque ni remotamente tan popular como alguna vez fue, Smalltalk todavía existe, y puede encontrar mucho más al respecto en www.smalltalk.org.

C++

Como lo sugiere su nombre, C++ se basa en el lenguaje de programación C. En realidad, **++** es el operador en C que incrementa en 1 una variable, de modo que C++ literalmente significa "C más 1". Este superconjunto de C fue desarrollado principalmente por Bjarne Stroustrup en AT&T Bell Laboratorios, en 1986. Las clases se implementan como tipos definidos por el usuario; son una *estructura*, según la sintaxis de C. Los métodos se implementan como funciones integrantes de una estructura. Los puristas de los objetos desaprueban C++ y afirman que no es un lenguaje orientado a objetos porque los programadores no pueden ignorar el paradigma de objetos cuando seleccionan y hacen cosas como manipular los datos directamente utilizando comandos de lenguaje C. Por otra parte, los aficionados a C++ consideran esto un beneficio enorme porque les ofrece mucha flexibilidad.

Java

Java es un lenguaje orientado a objetos de propósito general, sencillo y portátil, desarrollado por Sun Microsystems alrededor de 1995. Tomó el mercado por asalto inmediatamente después de su introducción, en gran medida porque permite la programación por Internet en forma de "applets", que son independientes de las plataformas. Otra ventaja de Java es que puede ejecutarse en computadoras muy pequeñas debido al reducido tamaño de su intérprete. A diferencia de Smalltalk y C++, Java es un lenguaje *intérprete*, lo que significa que cada instrucción se evalúa durante la ejecución en lugar de ser compilada con anticipación. Un *compilador* es un programa que convierte un programa computacional del lenguaje de origen que utilizó el programador para escribirlo en un lenguaje de máquina de la computadora en que cual se va ejecutar. Al principio, el intérprete obstaculizaba el desempeño en comparación con los lenguajes compilados, pero algunas innovaciones recientes, como los compiladores *justo a tiempo*, que compilan las instrucciones justo antes de su ejecución, han sido de gran utilidad.

Persistencia de objetos

La *persistencia* es la propiedad orientada a objetos que conserva el estado de un objeto entre ejecuciones de una aplicación y a través del apagado y encendido del equipo de cómputo. En casi todos los casos, se utiliza una base de datos para guardar los objetos de manera permanente, de modo que la persistencia es implementada por la base de datos. Los objetos deben cargarse en

la memoria para que una aplicación acceda a ellos, y cualquier cambio debe guardarse para su almacenamiento persistente cuando ya no se requiera. La carga de objetos en la memoria es un proceso *indirecto*, lo que significa que la aplicación no solicita específicamente que se cargue un objeto: el ambiente de la aplicación colabora con el entorno de la base de datos para cargar los objetos en la memoria automáticamente cuando una aplicación accede a ellos. Este acceso suele estar en forma de un mensaje que se envía al objeto pero, como se analiza en la sección siguiente, también puede ocurrir cuando un objeto contiene una referencia a otro.

En seguida se revisan dos métodos para implementar la persistencia de objetos mediante una base de datos: una base de datos orientada a objetos y otra relacional. En la sección siguiente, se explora un método híbrido que combina características de ambos tipos de base de datos.

Persistencia mediante una base de datos orientada a objetos

En la figura 13-2 se muestra la recuperación de un objeto del almacenamiento persistente en una base de datos orientada a objetos. Para los fines del ejemplo, se han omitido los componentes específicos que ejecutan cada uno de los pasos ilustrados, por lo que se muestra lo que

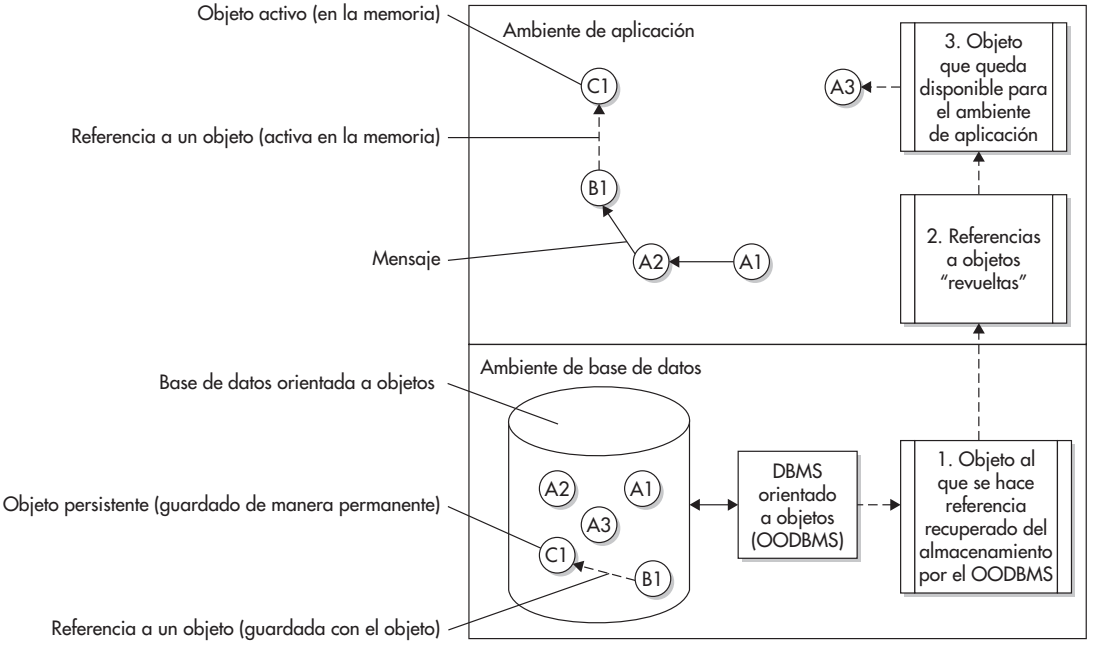


Figura 13-2 La persistencia mediante una base de datos orientada a objetos.

ocurre sin preocuparse de cómo ocurre. En realidad ésta es una excelente manera de considerar las bases de datos orientada a objetos, porque una propiedad común de los sistemas correspondientes es ocultar los detalles de la implementación. Como se aprecia en la figura 13-2, la base de datos contiene copias persistentes de los objetos A1, A2, A3, B1 y C1. Suponga que la primera letra indica la clase a la que pertenecen los objetos. Observe que B1 hace referencia al objeto C1 tal como se ilustra, y se emplea una línea de guiones para conectarlos. Éste es un diseño normal en que un objeto, como un pedido, contiene la identificación de objeto (OID) de un objeto relacionado, como el cliente que hizo el pedido. En una base de datos relacional equivalente, esta relación se implementaría mediante una clave externa en el pedido.

Como se muestra en la figura 13-2, ésta es la secuencia de eventos que ocurren la primera vez que la aplicación hace referencia a un objeto:

1. Se envía a la base de datos orientada a objetos una solicitud para recuperar el objeto, por lo general porque un mensaje en el ambiente de la aplicación hizo referencia al objeto. El DBMS orientado a objetos (Object Oriented DBMS, OODBMS) recupera el objeto del almacenamiento persistente y lo transfiere al ambiente de la aplicación. Si el objeto contiene referencias a otros objetos, el OODBMS también puede recuperar automáticamente esos objetos, dependiendo de la arquitectura del OODBMS.
2. Si un objeto contiene referencias a otros, esas referencias deben transformarse en direcciones en la memoria cuando se cargan los objetos en la misma. A este proceso se le conoce como *revolver* las referencias. (Se desconoce el origen del término *revolver*, pero puede derivarse de los bastones o agitadores que sirven para revolver bebidas.) En el almacenamiento persistente, se utiliza la OID como referencia, porque el OODBMS puede emplear otras estructuras de almacenamiento similares a los índices para localizar los objetos relacionados. Por ejemplo, el objeto B1 contiene la OID del objeto C1, y el OODBMS no tiene dificultad para utilizar la OID con el propósito de localizar el objeto relacionado en el almacenamiento persistente de la base de datos. Sin embargo, la OID no sirve para localizar el objeto relacionado una vez que éste está cargado en la memoria, porque los objetos se cargan en cualquier lugar disponible de la memoria, lo que significa que no hay un modo sencillo para saber el lugar que ocupan. Por lo tanto, la OID se traduce (se revuelve) a la dirección real que ocupa el objeto relacionado en la memoria para permitir un acceso directo de ese objeto. La OID original se conserva dentro del objeto porque será necesaria cuando el objeto se vuelva a guardar en la base de datos.
3. El objeto queda disponible para el entorno de la aplicación. Es decir, se pone en un lugar de la memoria y los mensajes dirigidos al objeto se enrutan a ese lugar. Por lo general, esto también implica registrar el objeto con el entorno de la aplicación, para que se pueda encontrar con facilidad en la memoria la siguiente ocasión que se haga referencia a él.

El proceso inverso de guardar otra vez un objeto en la base de datos orientada a objetos cuando la aplicación ya no necesita acceder a él es exactamente eso: una *reversión* del

proceso original. Las condiciones que activan la devolución del objeto al almacenamiento persistente varían de un OODBMS a otro, pero suelen contener un algoritmo tipo *el utilizado menos recientemente* (LRU, Least Recently Used). El algoritmo LRU es un proceso que se invoca cuando debe liberarse espacio para cargar más objetos en lugares de la memoria. El algoritmo encuentra los objetos a los que se tuvo acceso hace más tiempo (es decir, menos recientemente) y los retira de la memoria. Y, por supuesto, una solicitud para apagar la base de datos requiere que todos los objetos que están en la memoria vuelvan a ser persistentes antes de apagar dicha base. Ésta es la secuencia de eventos para trasladar un objeto de la memoria al almacenamiento persistente:

1. El objeto es retirado de un lugar en la memoria, y se elimina cualquier registro del objeto en el entorno de la aplicación.
2. Se elimina cualquier dirección de la memoria agregada al objeto cuando se revolvieron las referencias.
3. Si el objeto fue modificado mientras estaba en la memoria, se devuelve al OODBMS, que guarda la versión nueva.

Persistencia mediante una base de datos relacional

Cuando los datos de un objeto se guardan en una base de datos relacional, el resultado son algunas diferencias importantes. En primer lugar, todo en una base de datos relacional debe guardarse en una tabla. Por lo tanto, los objetos deben traducirse desde las tablas relacionales y a éstas. Cada clase se suele guardar en una tabla relacional diferente, y las filas de las tablas representan instancias de objetos de las clases correspondientes. En segundo lugar, las tablas relacionales no pueden guardar objetos en su formato nativo, porque los objetos están formados por métodos y una jerarquía de clases, junto con los datos mismos. Los métodos y la jerarquía de clases no suelen guardarse en la base de datos relacional; más bien, se conservan en un lugar del sistema de archivos (directorio) administrado por el entorno de la aplicación. La figura 13-3 ilustra este diseño.

Tome nota de las diferencias entre las figuras 13-2 y 13-3. El primer lugar, en esta última figura, los datos de un objeto se guardan en la base de datos en tablas. En segundo lugar, se requiere un paso adicional cuando se recuperan objetos para que queden disponibles en la memoria: los datos de la base de datos relacional deben convertirse a clases de objetos y variables. Esto se consigue de varias maneras. Un método común en las aplicaciones escritas en Java consiste en emitir el SQL relacional directamente desde un método de Java mediante un controlador para conectividad de base de datos de Java (JDBC, Java DataBase Connectivity), que se presentó en el capítulo 9, y dentro del mismo método, relacionar los resultados devueltos por el controlador JDBC para uno o más objetos. Éste es un método manual que representa mucho trabajo para los programadores en Java. Por suerte, existen soluciones más automatizadas, en donde un servidor de aplicaciones o un producto de middleware maneja todos los detalles de los objetos guardados de manera persistente en una base de datos relacional, entre

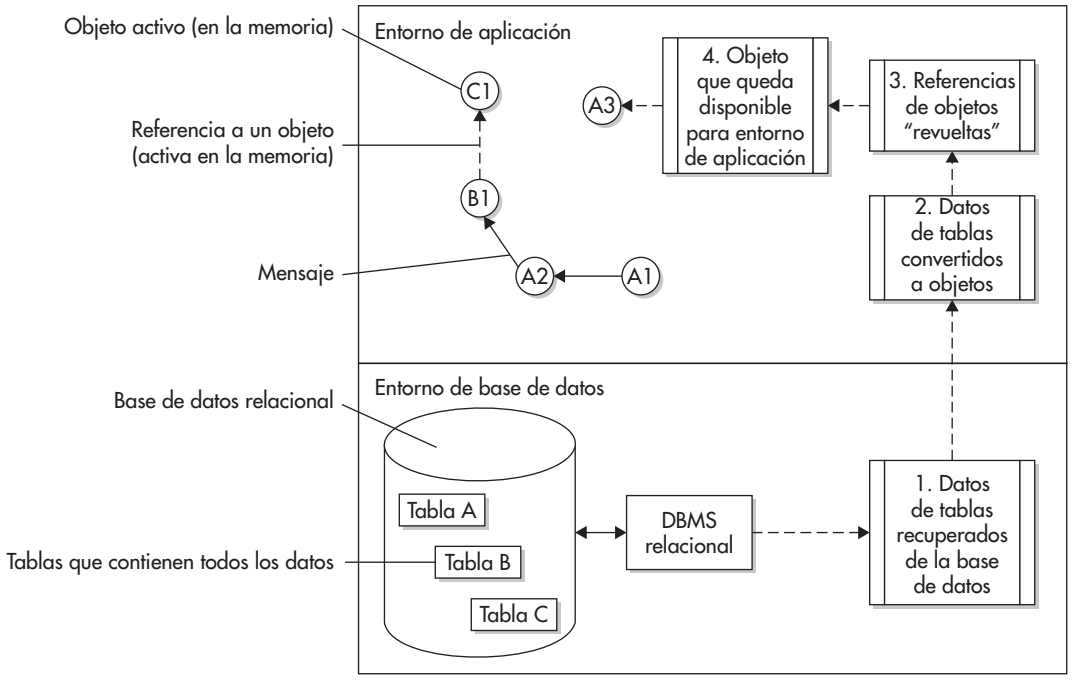


Figura 13-3 La persistencia mediante una base de datos relacional.

ellos la traducción entre tablas relacionales y objetos. La figura 13-3 ha sido simplificada para mostrar los pasos requeridos para ensamblar un objeto guardado en una base de datos relacional y que ha quedado disponible en el entorno de la aplicación, sin los detalles de cuáles componentes manejan los pasos.

Tal como se ilustró en la figura 13-3, ésta es la secuencia de eventos requerida para ensamblar un objeto a partir de los datos guardados en una base de datos relacional:

1. Se envía una consulta de SQL al RDBMS para recuperar la tabla (que suele ser una fila) de la base de datos. El RDBMS ejecuta la consulta y los datos resultantes se envían al entorno de la aplicación.
2. La tabla de datos se convierte en el objeto. Esto suele incluir la asignación de los datos de la tabla a una clase y las columnas individuales a variables de esa clase, junto con la recuperación de los métodos definidos para la clase del lugar donde se guardan en el sistema de archivos. Este paso de la conversión es el proverbial talón de Aquiles de esta arquitectura: es muy costoso, y requiere soluciones intermedias de diseño porque los datos de los objetos no siempre pueden representarse a la perfección en las tablas de una base de datos relacional.

3. Igual que con la figura 13-2, se revuelven las referencias a los objetos.
4. Igual que con la figura 13-2, el objeto se pone en un lugar de la memoria y se registra con el entorno de la aplicación, con lo cual queda disponible para la aplicación.

Cuando ya no se necesita un objeto en la memoria, debe volverse al almacenamiento persistente. Ésta es la secuencia de eventos:

1. El objeto se retira de la memoria, y se elimina cualquier registro con el entorno de la aplicación. Si el objeto no fue modificado mientras estaba en la memoria, no se requiere ninguna otra acción; de lo contrario, la secuencia continúa con el paso siguiente.
2. Se retiran las direcciones de la memoria agregadas para las referencias a un objeto.
3. Los datos del objeto se vuelven a convertir en las filas de la tabla relacional de la que provinieron. Se forman una o más instrucciones de SQL (INSERT, UPDATE o DELETE) para modificar los datos de la base de datos relacional con el fin de que coincidan con los datos de objetos. Por eficiencia, a menudo esto requiere la comparación con versiones anteriores y posteriores del objeto (si están disponibles) para que sólo sea necesario hacer referencia a las variables que se modificaron de algún modo en las instrucciones SQL generadas. No es necesario hacer nada con la estructura de clases o los métodos porque no se modifican cuando el objeto es utilizado en el entorno de la aplicación. Estos componentes sólo cambian cuando se instala una versión nueva de la aplicación.
4. Las instrucciones SQL son transferidas al DBMS relacional para su procesamiento. Si el objeto no se modificó mientras estaba en la memoria, no se requiere este paso.

Bases de datos de objetos-relacionales

En esta lección supone que ya leyó y comprendió la sección “El modelo de objetos-relacional” del capítulo 1. Antes de continuar, es probable que quiera repasarlo. El DBMS de objetos-relacional (ORDBMS) evolucionó como respuesta a las dificultades para convertir objetos a bases de datos relacionales y para presionar al mercado de los vendedores de OODBMS. Los vendedores de bases de datos relacionales, como Informix (adquirida después por IBM) y Oracle, agregaron extensiones de objetos con la esperanza de no reducir su participación en el mercado ante los vendedores de OODBMS. En gran medida, esta táctica parece haber funcionado, porque las bases de datos orientadas a objetos puras sólo ganaron terreno en mercados de nicho. Además, es evidente que la falta de capacidad para consultas *ad hoc* en las bases de datos orientadas a objetos puras no ayudó a este mercado. El ORDBMS ofrece una combinación de funciones convenientes del mundo de los objetos, como el almacenamiento de tipos de datos complejos, con la relativa sencillez y facilidad de uso del modelo relacional. Casi todos los expertos en la industria creen que la tecnología de objetos-relacional seguirá aumentando su participación en el mercado.

Éstas son las ventajas de una base de datos de objetos-relacional:

- Se permiten directamente los tipos de datos complejos (es decir, tipos de datos formados por la combinación de otros tipos de datos), al mismo tiempo que se conserva la capacidad para consultas *ad hoc*.
- El DBMS puede extenderse para realizar de manera central las funciones (métodos) comunes, lo que mejora la reutilización de la lógica del programa en comparación con un DBMS relacional puro.
- Como las funciones (métodos) se guardan en la base de datos, están disponibles para todas las aplicaciones, lo que permite compartir mejor los objetos en comparación con un DBMS relacional puro.
- La capacidad para consultas *ad hoc* se permite por completo, que es una función que no está permitida en las bases de datos orientadas a objetos puras.

Éstas son las desventajas del método de objetos-relacional:

- La combinación es más compleja que una base de datos relacional pura o una base de datos orientada a objetos pura, lo que genera un aumento de los costos de desarrollo.
- Los objetos están *centrados en las tablas*, lo que significa que todos los objetos persistentes deben guardarse dentro de una tabla.
- Los puristas relacionales argumentan que la sencillez esencial del modelo relacional es empañada por las extensiones de objetos.
- A los puristas de los objetos no les atrae la extensión de los objetos dentro de las bases de datos relacionales, y argumentan que el ORDBMS es poco más que una base de datos relacional que incluye tipos de datos definidos por el usuario.
- Los ORDBMS actuales carecen de la estructura de clases y la herencia que son la base de los OODBMS.
- Las aplicaciones de objetos no se centran tanto en los datos como las aplicaciones relacionales y, por lo tanto, las bases de datos orientadas a objetos puras atienden mejor las necesidades de las aplicaciones de objetos.

Para decidir cuál modelo de bases de datos se ajusta mejor para una aplicación específica, considere los elementos siguientes:

- Los datos sencillos que no requieren la capacidad de consultas *ad hoc*, como las páginas Web estáticas, se guardan adecuadamente en archivos comunes del sistema de archivos.
- Los datos sencillos que necesitan la capacidad de consultas *ad hoc*, como los datos de los clientes, se acoplan bien con una base de datos relacional.
- Los datos complejos que no requieren la capacidad de consultas *ad hoc*, como imágenes, mapas, dibujos, se adaptan bien a una base de datos orientada a objetos.

- Los datos complejos que requieren la capacidad de consulta *ad hoc*, como los pedidos de compras guardados como tipos de datos compuestos, se adaptan bien a una base de datos de objetos-relacional.

Autoexamen Capítulo 13

Elija las respuestas correctas para cada una de las preguntas de opción múltiple y de llenado de espacios en blanco. Tome en cuenta que puede haber más de una respuesta correcta para cada pregunta.

1. XML es _____.

2. ¿Cómo varían las bases de datos de SQL y los documentos de XML en cuanto a la estructura de los datos?

3. Si dos organizaciones emplean XML, ¿esto significa que tienen un modo estándar para intercambiar datos sin tener que crear un software de interfaz?

4. Los modificadores de tipo secundario válidos para el modificador de tipo SEQUENCE son _____.

5. El tipo de esquema XML _____ se convierte a partir del tipo de datos NUMERIC de SQL.

6. El tipo de esquema XML _____ se convierte a partir del tipo de datos DATE de SQL.

7. Las dos maneras en que los valores nulos de la base de datos se representan en SQL/XML son _____ y _____.

8. ¿Cuáles son usos comunes de XML?

A Desplegar los datos de una base de datos en una página Web.

B Crear páginas Web estáticas.

C Transmitir los datos de una base de datos a otra persona.

D Imponer las reglas de negocios sobre los documentos.

9. ¿Cuáles son modificadores de tipo válidos para el tipo de datos XML?

A DOCUMENT.

B SEQUENCE.

C SQLXML.

D CONTENT.

10. ¿Cuáles función de SQL/XML crea un elemento basado en una columna de una tabla?

- **A** XMLQUERY.
- **B** XMLELEMENT.
- **C** XMLFOREST.
- **D** XMLDOCUMENT.
- **E** XMLPARSE.

11. La programación orientada a objetos

- **A** Emplea mensajes como vehículo para la interacción con los objetos.
- **B** Permite a un objeto acceder directamente a las variables en un objeto relacionado.
- **C** Utiliza métodos para definir el comportamiento de un objeto.
- **D** Requiere que los objetos tengan una clave principal.
- **E** Permite el uso de objetos complejos.

12. Las aplicaciones orientadas a objetos

- **A** Requieren el uso de una base de datos orientada a objetos.
- **B** Están escritas en un lenguaje orientado a objetos.
- **C** Emplean entornos de desarrollo que suelen incluir clases predefinidas.
- **D** Emplean entornos de desarrollo que suelen incluir métodos predefinidos.
- **E** Pueden escribirse en el lenguaje de programación C.

13. Smalltalk

- **A** Fue desarrollado por Linus Torvalds.
- **B** Fue desarrollado en 1972.
- **C** Fue desarrollado en la instalación PARC de Xerox.
- **D** Está basado en el lenguaje de programación C.
- **E** Fue el primer lenguaje de programación orientada a objetos que incluye un sistema de ventanas y el uso de un ratón.

14. C++

- **A** Fue desarrollado por Alan Kay.
- **B** Fue desarrollado en 1976.
- **C** Fue desarrollado en AT&T Bell Laboratories.

D Está basado en el lenguaje de programación Java.

E Permite a los programadores ignorar el paradigma de objetos, si lo prefieren.

15. Java

A Fue desarrollado por Sun Microsystems.

B Sólo puede ejecutarse en sistemas grandes con mucha memoria.

C Fue desarrollado alrededor de 1995.

D Es un lenguaje intérprete.

E Es un lenguaje orientado a objetos de propósito general.

16. La persistencia de objetos

A Conserva el estado de un objeto entre ejecuciones de una aplicación.

B Conserva el estado de un objeto tras el apagado y encendido de una computadora.

C Carga los objetos en la memoria para preservarlos de manera permanente.

D Ocurre cuando la aplicación solicita que se guarde un objeto.

E Sólo se consigue con una base de datos orientada a objetos.

17. Algunos eventos necesarios para recuperar un objeto de una base de datos orientada a objetos son

A Se envía un mensaje al objeto, de modo que éste debe cargarse en la memoria.

B Se envía a la base de datos orientada a objetos una solicitud para recuperar el objeto.

C Las referencias a objetos se revuelven en direcciones de la memoria.

D Se asignan datos relacionales a una clase de objetos.

E El objeto queda disponible para el entorno de la aplicación.

18. Algunas ventajas de las bases de datos de objetos-relacionales son

A Los objetos se guardan dentro de las tablas.

B Se permiten tipos de datos complejos.

C Permiten por completo la capacidad de consultas *ad hoc*.

D Permiten por completo las estructuras de clases y la herencia.

E Las funciones (métodos) guardadas de manera central mejoran la reutilización.

19. Entre las desventajas de las bases de datos de objetos-relacionales están

- **A** La combinación es más compleja que en las bases de datos orientadas a objetos puras o en las relacionales puras.
- **B** Es limitada la capacidad para consultas *ad hoc.*
- **C** Los objetos están centrados en las tablas.
- **D** Ni los puristas relacionales ni los de objetos están enamorados de la combinación.
- **E** Las aplicaciones de objetos no están tan centradas en los datos como las relacionales.

20. Al considerar la elección de un modelo de base de datos, ¿cuáles de los hechos siguientes deben tomarse en cuenta?

- **A** Los archivos de un sistema de archivos comunes pueden manejar los datos sencillos, siempre y cuando no existan requisitos para consultas *ad hoc.*
- **B** Las bases de datos relacionales pueden manejar los datos sencillos que tienen requisitos para consultas *ad hoc.*
- **C** Las bases de datos orientadas a objetos son mejores para manejar datos complejos.
- **D** Las bases de datos de objetos-relacionales pueden manejar datos complejos que tienen requisitos de consultas *ad hoc*.
- **E** Las bases de datos orientadas a objetos pueden manejar datos complejos, siempre y cuando no haya requisitos de consultas *ad hoc*.

Parte IV

Apéndices

Apéndice A

Soluciones a las autoevaluaciones

Capítulo 1: Fundamentos de bases de datos

1. ¿Cuál de los siguientes conceptos es aportado por la capa lógica del modelo ANSI/SPARC?

A Independencia física de los datos.

B Relaciones elementos principales-secundarios.

C Independencia lógica de los datos.

D Encapsulado.

A es la respuesta correcta.

2. ¿Cuál de los siguientes conceptos es proporcionado por la capa externa del modelo ANSI/SPARC?

A Independencia física de los datos.

B Relaciones elementos principales-secundarios.

C Independencia lógica de los datos.

D Encapsulado.

C es la respuesta correcta.

3. En relación con las vistas de usuario, ¿cuál de las siguientes afirmaciones *no* es verdadera?

A Los programas de aplicación hacen referencia a ellas.

B Las personas que consultan la base de datos hacen referencia a ellas.

C Se ajustan a las necesidades del usuario de la base de datos.

D Las actualizaciones de los datos se muestran en forma desfasada.

D es la respuesta correcta.

4. El esquema de la base de datos está contenido en la capa _____ del modelo ANSI/SPARC.

lógica

5. Las vistas de usuario están en la capa _____ del modelo ANSI/SPARC.

externa

6. Cuando los programas de aplicación emplean los sistemas de archivo simple, ¿dónde residen las definiciones del archivo?

En los programas de aplicación.

7. En relación con el modelo jerárquico de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

A Fue desarrollado primero por Peter Chen.

B Los datos y los métodos se guardan juntos en la base de datos.

C Cada nodo puede tener muchos padres.

D Los registros se conectan mediante apuntadores a una dirección física.

D es la respuesta correcta.

8. En relación con el modelo de red de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

A Fue desarrollado primero por E. F. Codd.

B Los datos y los métodos se guardan juntos en la base de datos.

C Cada nodo puede tener muchos elementos principales.

D Los registros se conectan mediante apuntadores a una dirección física común.

C y **D** son las respuestas correctas.

9. En relación con el modelo relacional de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

A Fue desarrollado por Charles Bachman.

B Los datos y los métodos se guardan juntos en la base de datos.

C Los registros se conectan mediante apuntadores a una dirección física.

D Los registros se conectan mediante elementos de datos comunes en cada registro.

D es la respuesta correcta.

10. En relación con el modelo orientado a objetos de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

A Fue desarrollado por Charles Bachman.

B Los datos y los métodos se guardan juntos en la base de datos.

C Los datos se presentan en tablas bidimensionales.

D Los registros se conectan mediante elementos de datos comunes en cada registro.

B es la respuesta correcta.

11. En relación con el modelo de objetos-relacional de bases de datos, ¿cuál de las siguientes afirmaciones es verdadera?

A Sólo atiende a un mercado reducido y casi todos los expertos creen que así se mantendrá.

B Los registros se conectan mediante apuntadores a una dirección física.

C Fue desarrollado al incorporar propiedades similares a las de objetos al modelo relacional.

D Fue creado al agregar propiedades similares a las relacionales al modelo orientado a objetos.

C es la respuesta correcta.

12. Según los defensores del modelo relacional, ¿cuál de las siguientes afirmaciones describe los problemas con el modelo de CODASYL?

A Es demasiado matemático.

B Es muy complicado.

C Las consultas orientadas a conjuntos son muy difíciles.

D No tiene sustento formal en la teoría matemática.

B, **C** y **D** son las respuestas correctas.

13. De acuerdo con los defensores del modelo de CODASYL, ¿cuál de las siguientes afirmaciones describe los problemas del modelo relacional?

A Es demasiado matemático.

B Las consultas orientadas a conjuntos son muy difíciles.

C Los sistemas de aplicación necesitan procesar de registro en registro.

D Es menos eficiente que las bases de datos del modelo CODASYL.

A, **C** y **D** son las respuestas correctas.

14. La capacidad para incorporar un objeto nuevo a una base de datos sin alterar los procesos existentes es un ejemplo de _____.

independencia lógica de los datos.

15. La propiedad que más distingue a una tabla de base de datos relacional de una hoja de cálculo es la capacidad para presentar a varios usuarios _____ propias.

vistas de los datos.

Capítulo 2: Exploración de los componentes de una base de datos relacional

1. Ejemplos de una entidad son

A Un cliente.

B Un pedido de clientes.

C El cheque de pago de un empleado.

D El nombre de un cliente.

A, **B** y **C** son las respuestas correctas.

2. Ejemplos de un atributo son

A Un empleado.

B El nombre de un empleado.

C El cheque de pago de un empleado.

D La lista alfabética de empleados.

B es la respuesta correcta.

3. ¿Cuál de los siguientes elementos denota la cardinalidad de "cero, uno o más" en una línea de relación?

A Una línea perpendicular cerca del extremo de la línea y una pata de gallo en el extremo de la línea.

B Un círculo cerca del extremo de la línea y una pata de gallo en el extremo de la línea.

C Dos líneas perpendiculares cerca del extremo de la línea.

D Un círculo y una línea perpendicular cerca del extremo de la línea.

B es la respuesta correcta.

4. Los tipos válidos de relaciones en una base de datos relacional son

A Uno a varios.

B Ninguno a varios.

C Varios a varios.

D Uno a uno.

A, **C** y **D** son las respuestas correctas.

5. Si un producto se puede fabricar en varias plantas, y una planta puede fabricar varios productos, ¿éste es un ejemplo de qué tipo de relación?

A Uno a uno.

B Uno a varios.

C Varios a varios.

D Recursiva.

C es la respuesta correcta.

6. Entre los siguientes, ¿cuáles son ejemplos de relaciones recursivas?

A Una unidad organizativa formada por departamentos.

B Un empleado que dirige a otros empleados.

C Un empleado que dirige un departamento.

D Un empleado que tiene muchos dependientes.

B es la respuesta correcta.

7. Ejemplos de una regla de negocios son

A Una restricción referencial debe hacer referencia a la clave principal de la tabla principal.

B Un empleado debe tener cuando menos 18 años de edad.

C Una consulta de base de datos elimina las columnas que un empleado no debe ver.

D No se permite modificar pedidos a los empleados con una calificación de pago menor de 6.

B y **D** son las respuestas correctas.

8. Una tabla relacional

A Está formada por filas y columnas.

B Debe tener asignado un tipo de datos.

C Debe tener asignado un nombre único.

D Es la unidad de almacenamiento principal del modelo relacional.

A, **C** y **D** son las respuestas correctas.

9. Una columna de una tabla relacional

A Debe tener asignado un tipo de datos.

B Debe tener asignado un nombre único dentro de la tabla.

C Se deriva de una entidad en el diseño conceptual.

D Es la unidad de almacenamiento con nombre más pequeña en una base de datos relacional.

A, **B** y **D** son las respuestas correctas.

10. Un tipo de datos

A Ayuda al DBMS a guardar los datos de manera eficiente.

B Ofrece un conjunto de comportamientos para una columna que ayuda al usuario de la base de datos.

C Puede ser seleccionada con base en las reglas de negocios de un atributo.

D Limita los caracteres permitidos en la columna de una base de datos.

A, **B**, **C** y **D** son las respuestas correctas.

11. Una restricción de clave principal

A Debe hacer referencia a una o más columnas de una sola tabla.

B Debe definirse para cada tabla de una base de datos.

C Suele implementarse mediante un índice.

D Garantiza que dos filas de una tabla no tengan valores de clave principal duplicados.

A, **C** y **D** son las respuestas correctas.

12. Una restricción referencial

A Debe tener columnas de clave principal y clave externa con nombres idénticos.

B Asegura que una clave principal no tenga valores duplicados en una tabla.

C Define una relación varios a varios entre dos tablas.

D Asegura que un valor de clave externa siempre haga referencia a un valor existente de clave principal en la tabla principal.

D es la respuesta correcta.

13. Una restricción referencial se define

A Mediante el panel Relaciones en Microsoft Access.

B Mediante SQL en casi todas las bases de datos relacionales.

C Mediante el tipo de datos referencial para las columnas de clave externa.

D Mediante un desencadenador de base de datos.

A y **B** son las respuestas correctas.

14. Los tipos principales de restricciones de integridad son

A Las restricciones de comprobación (CHECK).

B Las relaciones uno a uno.

C Las restricciones NOT NULL.

D las restricciones impuestas con desencadenadores.

A, **C** y **D** son las respuestas correctas.

15. Las tablas _____ se emplean para resolver las relaciones varios a varios.
de intersección

16. Una entidad en el diseño conceptual se vuelve _____ en el diseño lógico.
una tabla

17. Un atributo en el diseño conceptual se vuelve _____ en el diseño lógico.

una columna

18. Los elementos en el nivel externo del modelo ANSI/SPARC se vuelven _____ en el modelo lógico.

vistas

19. Una relación en el diseño conceptual se vuelve _____ en el modelo lógico.

una restricción referencial

20. Una restricción de clave principal se implementa mediante _____ en el diseño lógico.

un índice

Capítulo 3: Consultas a bases de datos con formularios

1. Un lenguaje de consultas mediante formularios

A Fue desarrollado por primera vez por IBM en la década de 1980.

B Describe cómo debe procesarse una consulta y no cuáles deben ser los resultados.

C Se parece a SQL.

D Usa una GUI (interfaz gráfica de usuario).

E Se demostró que es claramente superior en estudios controlados.

D es la respuesta correcta.

2. Los tipos de objetos en Microsoft Access que se relacionan estrictamente con la administración de la base de datos (en contraste con el desarrollo de una aplicación) son

A Las tablas.

B Las consultas.

C Los formularios.

D Las macros.

E Los módulos.

A y **B** son las respuestas correctas.

3. Cuando se elimina una tabla del panel Relaciones de Microsoft Access, ¿qué ocurre después?

A Es eliminada inmediatamente de la base de datos.

B Permanece intacta en la base de datos y sólo es retirada del panel Relaciones.

C Se mantiene en la base de datos, pero se eliminan todas las filas de datos.

D También se eliminan las relaciones que pertenecen a la tabla.

B es la respuesta correcta.

4. Las relaciones en el panel Relaciones de Microsoft Access representan _____ en la base de datos.

restricciones referenciales

5. Una columna en los resultados de una consulta de Microsoft Access se forma a partir de

A Una columna de una tabla.

B Una columna de una consulta.

C Una constante.

D Un cálculo.

E Todos los anteriores.

E es la respuesta correcta.

6. Cuando se ejecuta una consulta sin criterios incluidos, el resultado es

A Un mensaje de error.

B No se exhiben filas.

C Se exhiben todas las filas de la tabla.

D Ninguno de los anteriores.

C es la respuesta correcta.

7. Cuando no se incluye el ordenamiento de las filas en una consulta a la base de datos, las filas devueltas por la consulta están en orden _____.

no específico

8. En una consulta, el criterio de búsqueda REGIÓN NO = "CA" O REGIÓN NO = "NV" mostrará

A Un mensaje de error.

B Todas las filas de la tabla.

C Sólo las filas en que REGIÓN es igual a "CA" o "NV".

D Todas las filas de la tabla, excepto las que tienen NULL en REGIÓN.

E Todas las filas de la tabla, excepto las que tienen un valor "CA" o "NV" en REGIÓN.

D es la respuesta correcta.

9. Los criterios en líneas diferentes de una consulta de Microsoft Access se conectan con el operador lógico _____.

O

10. El conector de combinación entre las tablas en una consulta de Microsoft Access puede

A Crearse de manera manual al arrastrar una columna de una tabla o vista a una columna de otra tabla o vista.

B Heredarse de los metadatos definidos en el panel Relaciones.

C Ser alterado para definir combinaciones externas izquierda, derecha y completa.

D Generar un producto cartesiano, si no se define entre dos tablas o vistas de la consulta.

E Todos los anteriores.

E es la respuesta correcta.

11. Cuando se emplea una combinación externa, los datos de columna de las tablas (o vistas) en que no se encontraron filas que coincidieran contendrán _____.

valores nulos

12. Una función de obtención de totales

A Combina los datos de varias columnas.

B Combina los datos de varias filas.

C Puede aplicarse a columnas de tablas, pero no a columnas calculadas.

D Requiere que cada columna de una consulta sea una función de totalización o aparezca por nombre en la lista AGRUPAR POR de la consulta.

E Todos los anteriores.

B y **D** son las respuestas correctas.

13. Las autocombinaciones de una consulta son un método para resolver una _____.

relación recursiva

14. El nombre de una columna calculada en los resultados de la consulta es _____ cuando no es proporcionado en la definición de la consulta.

asignado automáticamente por Microsoft Access

15. Las tablas se pueden combinar

A Usando sólo una clave principal en una tabla y una clave externa en otra.

B Mediante cualquier columna en cualquier tabla (en teoría).

C Sólo consigo mismas.

D Sólo con otras tablas.

E Sólo mediante una fórmula de un producto cartesiano.

B es la respuesta correcta.

Capítulo 4: Introducción a SQL

1. SQL se divide en los subconjuntos siguientes:

A Lenguaje de selección de datos (DSL)

B Lenguaje de control de datos (DCL)

C Lenguaje de consulta de datos (DQL)

D Lenguaje de definición de datos (DDL)

B y **D** son las respuestas correctas.

2. SQL fue desarrollado por _____.

IBM

3. A un programa utilizado para conectarse a la base de datos e interactuar con ella se le llama _____.

cliente de SQL

4. Una instrucción SELECT sin una cláusula WHERE

A Selecciona todas las filas de la tabla o vista de origen.

B No devuelve filas en el grupo de resultados.

C Produce un mensaje de error.

D Sólo presenta la definición de la tabla o vista.

A es la respuesta correcta.

5. En SQL, el orden de las filas en los resultados de la consulta

A Es especificado por la cláusula SORTED BY.

B Es impredecible, a menos que se especifique en la consulta.

C Cuando no se especifica una secuencia, la opción predeterminada es descendente.

D Sólo se puede especificar para las columnas en los resultados de la consulta.

B es la respuesta correcta.

6. El operador BETWEEN

A Incluye los valores del punto final.

B Selecciona las filas que se agregan a una tabla durante un intervalo.

C También se puede escribir mediante los operadores <= y NOT =.

D También se puede escribir mediante los operadores <= y >=.

A y **D** son las respuestas correctas.

7. El operador LIKE emplea _____ como comodines de posición y _____ como comodines que no están relacionados con la posición.

guiones bajos (_), signos de porcentaje (%)

8. Una selección secundaria

A Puede ser corrugada o no corrugada.

B Permite una selección flexible de las filas.

C No debe estar entre paréntesis.

D Puede usarse para seleccionar valores para aplicarlos a las condiciones de la cláusula WHERE.

B y **D** son las respuestas correctas.

9. Una combinación sin una cláusula WHERE o JOIN

A Produce un mensaje de error.

B Genera una combinación externa.

C Obtiene un producto cartesiano.

D No devuelve filas en el grupo de resultados.

C es la respuesta correcta.

10. A una combinación que devuelve todas las filas de ambas tablas, se encuentren coincidencias o no, se le conoce como _____.

combinación externa completa

11. Una autocombinación

A Incluye dos tablas diferentes.

B Puede ser una combinación interna o externa.

C Resuelve las relaciones recursivas.

D Puede emplear una selección secundaria para limitar más las filas devueltas.

B, **C** y **D** son las respuestas correctas.

12. Una instrucción SQL que contiene una función de obtención de totales

A Debe contener una cláusula GROUP BY.

B También puede incluir columnas comunes.

C No puede incluir cláusulas GROUP BY y ORDER BY.

D También puede incluir columnas calculadas.

B y **D** son las respuestas correctas.

13. _____ hace que los cambios realizados por una transacción adquieran permanencia.

Una instrucción COMMIT

14. Una instrucción INSERT

A Debe contener una lista de columnas.

B Debe contener una lista de valores.

C Puede crear varias filas de una tabla.

D Puede contener una consulta secundaria.

C y **D** son las respuestas correctas.

15. Una instrucción UPDATE sin una cláusula WHERE

A Produce un mensaje de error.

B No actualiza las filas de una tabla.

C Actualiza cada fila de una tabla.

D Genera un producto cartesiano.

C es la respuesta correcta.

16. Una instrucción DELETE sin una lista de columnas

A Produce un mensaje de error.

B Sólo elimina datos de las columnas de la lista.

C Elimina cada columna de la tabla.

D Se utiliza para eliminar de una vista.

A es la respuesta correcta.

17. Una instrucción CREATE

A Es una forma de DML.

B Crea nuevos privilegios del usuario.

C Genera un objeto de la base de datos.

D Se puede revertir después mediante una instrucción DROP.

C y **D** son las respuestas correctas.

18. Una instrucción ALTER

A Se utiliza para agregar una restricción.

B Sirve para descartar una restricción.

C Es útil para agregar una vista.

D Se emplea para descartar una columna de una tabla.

A, **B** y **D** son las respuestas correctas.

19. El modo _____ hace que cada instrucción SQL se confirme en cuanto concluye.

de confirmación automática

20. Los privilegios de una base de datos

A Se modifican con una instrucción **ALTER PRIVILEGE**.

B Pueden ser del sistema o de objetos.

C Se administran mejor cuando se integran en grupos mediante **GROUP BY**.

D Son administrados mediante **GRANT** y **REVOKE**.

B y **D** son las respuestas correctas.

Capítulo 5: El ciclo de vida de una base de datos

1. ¿Cuáles de los siguientes elementos son parte de las fases de una metodología de ciclo de vida de desarrollo de sistemas (CVDS)?

A Diseño físico.

B Diseño lógico.

C Creación de prototipos.

D Recopilación de requisitos.

E Soporte continuo.

A, **B**, **D** y **E** son las respuestas correctas.

2. Durante la fase de requisitos de un proyecto de CVDS,

A Se descubren las vistas de usuarios.

B Se emplea un entorno de aseguramiento de la calidad.

C Es posible efectuar encuestas.

D Se suele hacer entrevistas.

E Es posible emplear observaciones.

A, **C**, **D** y **E** son las respuestas correctas.

3. Las ventajas de realizar entrevistas son

A Las entrevistas requieren menos tiempo que otros métodos.

B Se obtienen respuestas a preguntas no formuladas.

C Es posible aprender mucho de las respuestas no verbales.

D Las preguntas se presentan de manera más objetiva, en comparación con las técnicas de la encuesta.

E Las entidades se descubren con mayor facilidad.

B y **C** son las respuestas correctas.

4. Algunas ventajas de efectuar encuestas son

A Se cubre mucho terreno con rapidez.

B No se incluyen respuestas no verbales.

C Las responden casi todos los participantes.

D Es sencillo redactarlas.

E No es necesario crear prototipos de los requisitos.

A es la respuesta correcta.

5. Las ventajas de la observación son

A Siempre se observa a las personas actuando normalmente.

B Es probable que se vean muchas situaciones en que se manejan excepciones.

C Realmente se ve cómo son las cosas, y no cómo las presenta la administración o la documentación.

D El efecto Hawthorne mejora sus resultados.

E Pueden observarse eventos que nadie más describiría.

C y **E** son las respuestas correctas.

6. Las ventajas de la revisión de documentos son

A Las imágenes y los diagramas son recursos valiosos para comprender los sistemas.

B Las revisiones de documentos se hacen con relativa rapidez.

C Los documentos siempre se mantendrán actualizados.

D Los documentos siempre reflejarán las prácticas actuales.

E Los documentos suelen presentar compendios que son mejores que otras técnicas.

A, **B** y **E** son las respuestas correctas.

7. Los módulos de un programa de aplicaciones se especifican durante la fase _____ del CVDS.

de diseño conceptual

8. Se suele efectuar un estudio de factibilidad durante la fase _____ de un proyecto de CVDS.

de planeación

9. Ocurre una normalización durante la fase _____ de un proyecto de CVDS.

de diseño lógico

10. Se escribe DDL para definir los objetos de una base de datos durante la fase _____ de un proyecto de CVDS.

de diseño físico

11. Las especificaciones de un programa se escriben durante la fase _____ de un proyecto de CVDS.

de diseño lógico

12. Durante la implementación y el lanzamiento,

A Se ubican los usuarios en el sistema real.

B Se diseñan mejoras.

C Las aplicaciones antigua y nueva pueden ejecutarse en paralelo.

D Se realiza una prueba de aseguramiento de la calidad.

E Sucede la capacitación de usuarios.

A, **C** y **E** son las respuestas correctas.

13. Durante el soporte continuo,

A Se implementan de inmediato las mejoras.

B El almacenamiento para la base de datos puede requerir una expansión.

C Ya no se requiere el entorno de presentación.

D Ocurre la reparación de defectos.

E Se aplican parches, si es necesario.

B, **D** y **E** son las respuestas correctas.

14. Cuando se crea un borrador de los requisitos, _____ puede funcionar bien.

un prototipo

15. El desarrollo rápido de aplicaciones crea sistemas con rapidez al omitir _____.

20% de los requisitos

16. Los tres objetivos representados en el triángulo de una aplicación son _____, _____ y _____.

calidad, costo, tiempo de entrega (o bueno, rápido y barato)

17. Al principio la base de datos se construye en el entorno _____.

de desarrollo

18. La conversión de una base de datos se prueba durante la fase _____ de un proyecto de CVDS.

de implementación y asignación

19. Las vistas de usuarios se analizan durante la fase _____ de un proyecto de CVDS.

de recopilación de requisitos

20. La base de datos relacional fue inventada por _____.

E. F. (Ted) Codd

Capítulo 6: Diseño de una base de datos mediante normalización

1. La normalización

A Fue desarrollada por E. F. Codd.

B Fue introducida primero con cinco formas normales.

C Apareció primero en 1972.

D Proporciona un grupo de reglas para cada forma normal.

E Ofrece un procedimiento para convertir las relaciones en cada forma normal.

A, **C**, **D** y **E** son las respuestas correctas.

2. El propósito de una normalización es

A Eliminar los datos redundantes.

B Eliminar ciertas anomalías de las relaciones.

C Aportar una razón para desnormalizar la base de datos.

D Optimizar el desempeño en la recuperación de datos.

E Optimizar los datos para inserciones, actualizaciones y eliminaciones.

B y **E** son las respuestas correctas.

3. Cuando se implementa, una relación en la tercera forma normal se convierte en _____.

una tabla

4. Anomalía de inserción alude a una situación en que

A Los datos deben insertarse antes de que puedan eliminarse.

B Demasiadas inserciones hacen que la tabla se llene.

C Los datos deben eliminarse antes de que puedan insertarse.

D Una inserción requerida no puede hacerse debido a una dependencia artificial.

E Una inserción requerida no puede hacerse a causa de datos duplicados.

D es la respuesta correcta.

5. Anomalía de eliminación alude a una situación en que

A Los datos deben eliminarse antes de que puedan insertarse.

B Los datos deben insertarse antes de que puedan eliminarse.

C Una eliminación de datos provoca una pérdida no intencional de datos de otra entidad.

D Una eliminación requerida no puede hacerse debido a restricciones referenciales.

E Una eliminación requerida no puede hacerse por falta de privilegios.

C es la respuesta correcta.

6. Anomalía de actualización alude a una situación en que

A Una actualización sencilla requiere actualizaciones a varias filas de datos.

B Los datos no pueden actualizarse porque no existen en la base de datos.

C Los datos no pueden actualizarse a causa de una falta de privilegios.

D Los datos no pueden actualizarse debido a una restricción de unicidad.

E Los datos no pueden actualizarse a causa de una restricción referencial existente.

A es la respuesta correcta.

7. Las reglas de los identificadores únicos en la normalización

A Son innecesarias.

B Son necesarias una vez que alcanza la tercera forma normal.

C Todas las formas normalizadas requieren la designación de una clave principal.

D No es posible normalizar relaciones sin primero elegir una clave principal.

E No es posible elegir una clave principal hasta que se normalizan las relaciones.

C y **D** son las respuestas correctas.

8. Escribir vistas de usuarios de ejemplo con datos representativos en ellas es

A El único modo de normalizar las vistas de usuarios con éxito.

B Un proceso tedioso y que consume tiempo.

C Un modo eficaz para comprender los datos que se normalizan.

D Tan bueno como los ejemplos presentados en los datos de ejemplo.

E Una técnica de normalización ampliamente utilizada.

B, **C** y **D** son las respuestas correctas.

9. Algunos criterios útiles para elegir una clave principal entre varias claves candidatas son

A Seleccionar la candidata más sencilla.

B Escoger la candidata más breve.

C Elegir la candidata con más probabilidad de que cambie su valor.

D Preferir las claves unidas a las claves con un solo atributo.

E Inventar una clave sustituta si es la mejor clave posible.

A, **B** y **E** son las respuestas correctas.

10. La primera forma normal resuelve las anomalías provocadas por _____.

atributos con valores múltiples

11. La segunda forma normal soluciona las anomalías inducidas por _____.

dependencias parciales

12. La tercera forma normal elimina las anomalías generadas por _____.

dependencias transitorias

13. En general, las violaciones a una regla de normalización son resueltas por medio de

A Combinación de relaciones.

B Desplazamiento de atributos o grupos de atributos a una relación nueva.

C Combinación de atributos.

D Creación de tablas de resumen.

E Una desnormalización.

B es la respuesta correcta.

14. Una clave externa en una relación normalizada puede ser

A La clave principal completa de la relación.

B Una parte de la clave principal completa de la relación.

C Un grupo repetido.

D Un atributo que no es una clave en la relación.

E Un atributo con varios valores.

A, **B** y **D** son las respuestas correctas.

15. La forma normal de Boyce-Codd atiende las anomalías causadas por _____.

determinantes que no son claves primarias o candidatas

16. La cuarta forma normal resuelve las anomalías provocadas por _____.

atributos con valores múltiples

17. La quinta forma normal enfrenta las anomalías generadas por _____.

dependencias de combinación

18. La forma normal dominio-clave soluciona las anomalías derivadas de _____.

restricciones que no son resultado de las definiciones de dominios y claves

19. Casi todos los sistemas de negocios requieren ser normalizados sólo hasta _____.

la tercera forma normal

20. El manejo adecuado de los atributos con varios valores, al convertir relaciones a la primera forma normal, suele evitar problemas subsecuentes con_____.

la cuarta forma normal

Capítulo 7: Modelado de datos y de procesos

1. ¿Por qué es importante que un diseñador de base de datos comprenda el modelo de procesos?

A El diseño de procesos es principalmente una responsabilidad del DBA.

B El modelo de procesos debe ser concluido antes del modelo de datos.

C El modelo de datos debe ser concluido antes del modelo de procesos.

D El diseñador de una base de datos debe colaborar estrechamente con el diseñador de procesos.

E El diseño de una base de datos debe permitir el modelo de procesos buscado.

D y **E** son las respuestas correctas.

2. El formato de ERD de Peter Chen representa "varios" con _____.

el símbolo *V* colocado cerca del final la línea de una relación

3. El rombo en el formato de Peter Chen representa _____.

una relación

4. El formato de ERD relacional representa "varios" con _____.

una pata de gallo

5. El formato ERD de IDEF1X

A Fue publicado en 1983.

B Sigue una norma desarrollada por el National Institute of Standards and Technology.

C Tiene muchas variantes.

D Ha sido adoptado como una norma por muchas dependencias gubernamentales de Estados Unidos.

E Cubre los modelos de datos y de procesos.

B y **D** son las respuestas correctas.

6. El formato de ERD de IDEF1X muestra

A Relaciones de identificación con una línea continua.

B La cardinalidad mínima con una combinación de círculos pequeños y líneas verticales sobre la línea de una relación.

C La cardinalidad máxima con una combinación de pequeñas líneas verticales y una pata de gallo dibujadas sobre la línea de una relación.

D Las entidades dependientes con un rectángulo de esquinas cuadradas.

E Las entidades independientes con un rectángulo de esquinas redondeadas.

A es la respuesta correcta.

7. Un tipo secundario

A Es un subconjunto de un supertipo.

B Tiene una relación uno a varios con el supertipo.

C Posee una relación uno a uno condicional con el supertipo.

D Muestra diversos estados del supertipo.

E Es un superconjunto de un supertipo.

A y **C** son las respuestas correctas.

8. Algunos ejemplos de tipos secundarios posibles de un supertipo de la entidad Pedido son

A Artículos de línea de pedido.

B Pedido enviado, pedido no enviado, pedido facturado.

C Pedido de artículos de oficina, pedido de servicios profesionales.

D Pedido aprobado, pedido pendiente, pedido cancelado.

E Pedido de pieza de automóvil, pedido de piezas de aeronave, pedido de piezas de camión.

C y **E** son las respuestas correctas.

9. En la notación II, los tipos secundarios

A Se muestran con un nombre de atributo discriminador de un tipo.

B Se conectan al supertipo mediante un símbolo formado por un círculo con una línea debajo.

C Tienen la clave principal del tipo secundario mostrado como clave externa en el supertipo.

D Suelen tener la misma clave principal que el supertipo.

E Se señalan mediante una pata de gallo.

A, **B** y **D** son las respuestas correctas.

10. Cuando se consideran los tipos secundarios del diseño de una base de datos, existe una solución intermedia entre _____ y _____.

la generalización, la especialización

11. En un diagrama de flujo, los pasos de un proceso se muestran como _____ y los puntos de decisión se presentan como _____.

rectángulos, rombos

12. Éstas son algunas ventajas de los diagramas de flujo

A Son naturales y fáciles de usar para los programadores de lenguajes de procedimientos.

B Son útiles para detectar componentes reutilizables.

C Sólo son específicos para la programación de aplicaciones.

D También son útiles para los lenguajes que no son de procedimientos y orientados a efectos.

E Se pueden modificar fácilmente conforme cambian los requisitos.

A, **B** y **E** son las respuestas correctas.

13. Los componentes básicos de un diagrama de jerarquía de funciones son

A Elipses para mostrar atributos.

B Rectángulos para mostrar funciones de procesos.

C Líneas que conectan los procesos en su orden de ejecución.

D Una jerarquía para presentar cuáles funciones están subordinadas a otras.

E Rombos para indicar puntos de decisión.

B y **D** son las respuestas correctas.

14. Las ventajas de un diagrama de jerarquía de funciones son

A La comprobación de la calidad es fácil y directa.

B Se modelan fácilmente las interacciones complejas entre las funciones.

C El aprendizaje de su uso es rápido y fácil.

D Muestra con claridad la secuencia de pasos de un proceso.

E Proporciona un buen panorama de los niveles de detalle alto e intermedio.

C y **E** son las respuestas correctas.

15. Los componentes básicos de un diagrama de carril de alberca son

A Líneas con flechas para mostrar la secuencia de pasos de un proceso.

B Rombos para señalar puntos de decisión.

C Carriles verticales para exponer las unidades organizacionales que efectúan los pasos de un proceso.

D Elipses para indicar los pasos de un proceso.

E Rectángulos con un extremo abierto para exhibir los almacenes de datos.

A, **C** y **D** son las respuestas correctas.

16. El diagrama de flujo de datos (DFD)

A Es el más centrado en los datos de todos los modelos de procesos.

B Fue desarrollado en la década de 1980.

C Combina las páginas de un diagrama de manera jerárquica.

D Fue desarrollado por E. F. Codd.

E Combina lo mejor de un diagrama de flujo y un diagrama de funciones.

A, **C** y **E** son las respuestas correctas.

17. En un DFD, los almacenes de datos se representan como _____, y los procesos se indican mediante _____.

rectángulos con un extremo abierto, rectángulos con esquinas redondeadas

18. Entre las ventajas de un DFD están

A Es bueno para trabajar un diseño de lo general a lo particular.

B Su desarrollo es rápido y fácil, incluso para sistemas complejos.

C Expone la estructura general sin sacrificar detalles.

D Presenta fácilmente la lógica compleja.

E Es estupendo para presentaciones ante la administración.

A, **C** y **E** son las respuestas correctas.

19. Los componentes de una matriz CRUD son

A Elipses para mostrar atributos.

B Los procesos principales se muestran sobre un eje.

C Las entidades principales se presentan en el otro eje.

D Números de referencia para señalar la jerarquía de procesos.

E Letras para indicar las operaciones que efectúan los procesos sobre las entidades.

B, **C** y **E** son las respuestas correctas.

20. La matriz CRUD ayuda a detectar los problemas siguientes:

A Entidades que nunca son leídas.

B Procesos que nunca son eliminados.

C Procesos que sólo leen.

D Entidades que nunca son actualizadas.

E Procesos que no tienen una entidad de creación.

A, **C** y **D** son las respuestas correctas.

Capítulo 8: Diseño de la base de datos física

1. Las reglas de negocios se implementan en la base de datos mediante _____.

restricciones

2. Dos diferencias importantes entre las restricciones de unicidad y de clave principal son _____ ____ y _____.

que una tabla sólo puede tener una restricción de clave principal; que las columnas a las que se hace referencia en las restricciones de clave principal deben definirse como NOT NULL.

3. Las relaciones en el modelo lógico se vuelven _____ en el modelo físico.

restricciones referenciales

4. Los nombres de restricciones son importantes porque _____.

aparecen en los mensajes de error

5. Al momento de diseñar tablas,

A Cada correlación normalizada se vuelve una tabla.

B Cada atributo en la correlación se vuelve una columna de tabla.

C Las relaciones se vuelven restricciones de comprobación.

D Los identificadores únicos se vuelven desencadenadores.

E Las columnas de clave principal deben definirse como NOT NULL.

A, **B** y **E** son las respuestas correctas.

6. Los supertipos y los tipos secundarios

A Deben implementarse exactamente como se especifica en el diseño lógico.

B Se pueden contraer en el diseño físico de una base de datos.

C Pueden tener las columnas del supertipo dobladas hacia cada tipo secundario en el diseño físico.

D Suelen tener la misma clave principal en las tablas físicas.

E Se aplican sólo al diseño lógico.

B, **C** y **D** son las respuestas correctas.

7. Los nombres de tablas

A Deben basarse en los nombres de atributos del diseño lógico.

B Siempre deben incluir la palabra "tabla".

C Sólo deben utilizar letras mayúsculas.

D Deben incluir los nombres de la organización o la ubicación.

E Pueden contener abreviaturas, cuando sea necesario.

E Pueden contener abreviaturas, cuando sea necesario.

C y **E** son las respuestas correctas.

8. Los nombres de columnas

A Deben ser únicos dentro de la base de datos.

B Deben basarse en los nombres de atributos correspondientes del diseño lógico.

C Deben emplear como sufijo el nombre de la tabla.

D Deben ser únicos dentro de la tabla.

E Deben emplear abreviaturas, cuando sea posible.

B y **D** son las respuestas correctas.

9. Las restricciones referenciales

A Definen las relaciones identificadas en el modelo lógico.

B Siempre se definen en la tabla primaria.

C Requieren que las claves externas se definan como NOT NULL.

D Deben tener nombres descriptivos.

E Asignan nombres a las tablas primaria y secundaria y a la columna de clave externa.

A y **D** son las respuestas correctas.

10. Las restricciones de comprobación

A Pueden utilizarse para obligar a una columna a que coincida con una lista de valores.

B Pueden aplicarse para que una columna coincida con un rango de valores.

C Pueden emplearse para forzar a una columna a coincidir con otra en la misma fila.

D Pueden usarse para obligar a una columna a coincidir con una que existe en otra tabla.

E Pueden servir para imponer una restricción de clave externa.

A, **B** y **C** son las respuestas correctas.

11. Los tipos de datos

A Evitan que se inserten datos incorrectos en una tabla.

B Sirven para evitar que se guarden caracteres alfabéticos en columnas numéricas.

C Se utilizan para evitar que se guarden caracteres numéricos en columnas con un formato de caracteres.

D Requieren que también se especifiquen la precisión y la escala.

E Se emplean para evitar que se guarden fechas no válidas en columnas de fechas.

B y **E** son las respuestas correctas.

12. ¿Cuáles son restricciones de vistas?

A Las vistas que contienen combinaciones nunca se actualizan.

B Se prohíben las actualizaciones a columnas calculadas en las vistas.

C Se requieren privilegios para actualizar datos mediante las vistas.

D Si una vista omite una columna obligatoria, no se permiten inserciones en la vista.

E Cualquier actualización relacionada con una vista puede hacer referencia sólo a las columnas de una tabla.

B, **C**, **D** y **E** son las respuestas correctas.

13. Algunas ventajas de las vistas son

A Siempre proporcionan ventajas en el desempeño.

B Pueden evitar que los usuarios de una base de datos cambien los nombres de tablas y columnas.

C Sirven para ocultar las combinaciones y los cálculos complejos.

D Pueden filtrar columnas o filas que los usuarios no deben ver.

E Se ajustan a las necesidades de los departamentos individuales.

A, **B**, **C**, **D** y **E** son las respuestas correctas.

14. Los índices

A Se utilizan como ayuda en las restricciones de clave principal.

B Sirven para mejorar el desempeño de una consulta.

C Se pueden emplear para mejorar el desempeño de inserciones, actualizaciones y eliminaciones.

D Suelen ser más pequeños que las tablas a las que hacen referencia.

E Son más lentos para examinar en forma secuencial que las tablas correspondientes.

A, **B** y **D** son las respuestas correctas.

15. ¿Cuáles son reglas generales aplicadas a los índices?

A Cuanto más grande sea una tabla, más importantes serán los índices.

B La inclusión de índices en columnas de clave externa suele ayudar en el desempeño de las combinaciones.

C Las columnas que se actualizan con frecuencia siempre deben contener índices.

D Cuantas más veces se actualice una tabla, más ayudarán los índices al rendimiento.

E En las tablas muy pequeñas, los índices tienden a no ser muy útiles.

A, **B** y **E** son las respuestas correctas.

Capítulo 9: Conexión de bases de datos al mundo exterior

1. En el modelo de implementación centralizada,

A Un servidor Web contiene todas las páginas Web.

B Se utiliza una terminal "tonta" como estación de trabajo cliente.

C La administración es muy fácil porque todo está centralizado.

D No existen puntos únicos para fallas.

E Los costos de desarrollo suelen ser muy altos.

B y **C** son las respuestas correctas.

2. En el modelo de implementación distribuida,

A La base de datos, la aplicación, o ambas está particionada y se implementa en varios sistemas computacionales.

B Las implementaciones iniciales tuvieron mucho éxito.

C La distribución puede ser transparente para el usuario.

D Los costos y la complejidad son reducidos en comparación con el modelo centralizado.

E La tolerancia a fallas es mejor en comparación con el modelo centralizado.

A, **C** y **E** son las respuestas correctas.

3. En el modelo cliente/servidor de dos niveles,

A Toda la lógica de las aplicaciones se ejecuta en un servidor de aplicaciones.

B Un servidor Web contiene las páginas Web.

C La estación de trabajo cliente maneja toda la lógica de presentación.

D La base de datos se aloja en un servidor centralizado.

E Las estaciones de trabajo cliente deben ser sistemas con gran potencia.

C, **D** y **E** son las respuestas correctas.

4. En el modelo cliente/servidor de tres niveles,

A Toda la lógica de las aplicaciones se ejecuta en un servidor de aplicaciones.

B Un servidor Web contiene las páginas Web.

C La estación de trabajo cliente maneja toda la lógica de presentación.

D La base de datos se aloja en un servidor centralizado.

E Las estaciones de trabajo cliente deben ser sistemas con gran potencia.

A, **C** y **D** son las respuestas correctas.

5. En el modelo cliente/servidor de N niveles,

A Toda la lógica de las aplicaciones se ejecuta en un servidor de aplicaciones.

B Un servidor Web contiene las páginas Web.

C La estación de trabajo cliente maneja toda la lógica de presentación.

D La base de datos se aloja en un servidor centralizado.

E Las estaciones de trabajo cliente deben ser sistemas con gran potencia.

A, **B**, **C** y **D** son las respuestas correctas.

6. Internet

A Comenzó como ARPANET del Departamento de Educación de Estados Unidos.

B Data de fines de la década de 1960 y principios de la de 1970.

C Siempre utilizó TCP/IP como estándar.

D Es un conjunto mundial de redes de computadoras interconectadas.

E Permite varios protocolos, entre ellos HTTP, FTP y Telnet.

B, **D** y **E** son las respuestas correctas.

7. Un URL puede contener

A Un protocolo.

B Un nombre de host o una dirección IP.

C Un puerto.

D La ruta absoluta hacia un recurso en un servidor Web.

E Argumentos.

A, **B**, **C**, **D** y **E** son las respuestas correctas.

8. Una intranet está disponible para _____.

miembros internos autorizados de una organización

9. Una extranet está disponible para _____.

personas externas autorizadas

10. World Wide Web emplea _____ para desplazarse por las páginas.

hipervínculos

11. HTTP no permite directamente el concepto de sesión porque _____.

no tiene estado

12. XML es extensible porque es posible definir _____.

etiquetas personalizadas

13. Las soluciones de middleware para conexiones de Java hacen que el RDBMS se vea como _____.

una base de datos orientada a objetos

14. La "pila de tecnología" de Web incluye

A Una estación de trabajo cliente que ejecuta un navegador Web.

B Un servidor Web.

C Un servidor de aplicaciones.

D Un servidor de la base de datos.

E Hardware de red (firewall, enrutadores y demás).

A, **B**, **C**, **D** y **E** son las respuestas correctas.

15. Las ventajas de CGI son

A No tiene estado.

B Su uso es sencillo.

C Resulta inherentemente segura.

D Es ampliamente aceptada.

E Es independiente del lenguaje y el servidor.

B, **D** y **E** son las respuestas correctas.

16. Las inclusiones del lado del servidor (SSI)

A Son comandos incrustados en un documento Web.

B Son puertas de enlace que no son CGI.

C Son macros de HTML.

D Resuelven algunos de los problemas de desempeño de CGI.

E Son inherentemente seguras.

A, **C** y **D** son las respuestas correctas.

17. Algunas ventajas de una puerta de enlace que no es CGI son

A Es reconocida por su estabilidad.

B Es una solución patentada.

C Ofrece mejor seguridad que las soluciones de CGI.

D Es más sencilla que CGI.

E Funciona en el espacio de direcciones de un servidor.

C y **E** son las respuestas correctas.

18. ODBC es

A Una API estándar para conectarse a un DBMS.

B Independiente de cualquier lenguaje, sistema operativo o DBMS específico.

C Un estándar de Microsoft.

D Utilizada por programas de Java.

E Flexible para manejar el SQL patentado.

A, **B** y **E** son las respuestas correctas.

19. JDBC es

A Una API estándar para conectarse a un DBMS.

B Independiente de cualquier lenguaje, sistema operativo o DBMS específico.

C Un estándar de Microsoft.

D Utilizada por programas de Java.

E Flexible para manejar el SQL patentado.

A, **D** y **E** son las respuestas correctas.

20. JSQL es

A Una norma de Sun Microsystems.

B Un método para incrustar instrucciones SQL en Java.

C Una extensión de una norma ISO/ANSI.

D Una solución de middleware.

E Independiente de cualquier lenguaje, sistema operativo o DBMS específico.

B y **C** son las respuestas correctas.

Capítulo 10: Seguridad de una base de datos

1. **A un conjunto de privilegios que se conceden a varios usuarios se le denomina _____.**

   función

2. **Los privilegios se rescinden mediante el comando _____ de SQL.**

   REVOKE

3. **Para los servidores de una base de datos conectados a una red, la aplicación exclusiva de seguridad física es _____.**

   inadecuada

4. **Los empleados que se conectan a una red empresarial desde su hogar u otro lugar remoto deben tener _____ entre la computadora y su módem de cable o DSL.**

   una firewall.

5. **Cuando las credenciales de inicio de sesión se guardan en la computadora, siempre deben _____.**

   estar cifradas

6. **La seguridad de red**

   A Sólo puede manejarse mediante enrutadores.

   B Sólo puede manejarse mediante una firewall.

   C Debe incluir consideraciones para los empleados que trabajan en otro lugar.

   D Debe ser obligatoria para todos los sistemas computacionales conectados a cualquier red.

   C y **D** son las respuestas correctas.

7. **La protección con firewall puede incluir**

   A Filtrado de paquetes.

   B Selección de paquetes mediante una tabla de enrutamiento.

   C Traducción de dirección de red.

   D Delimitación de los puertos que pueden emplearse para acceso.

   E Falsificación de IP.

   A, **C** y **D** son las respuestas correctas.

8. **Las redes inalámbricas deben asegurarse porque**

   A Existen en el mercado puntos de acceso inalámbricos económicos.

   B Cualquier persona con un adaptador de red inalámbrica se puede conectar a una red no protegida.

   C Los empleados pueden utilizar la red inalámbrica para comunicarse en secreto con hackers.

   D Las ondas de radio penetran las paredes de las oficinas vecinas.

E Las ondas de radio pueden llegar a la vía pública fuera de un edificio.

A, **B**, **D** y **E** son las respuestas correctas.

9. Algunos componentes de la seguridad de un punto de acceso inalámbrico son

A Traducción de dirección de red.

B La política de seguridad de una organización.

C Cifrado.

D Redes privadas virtuales.

E Listas de direcciones MAC.

B, **C** y **E** son las respuestas correctas.

10. Entre las precauciones para la seguridad en el nivel del sistema están

A La instalación de un mínimo de componentes de software necesarios.

B Conceder sólo los privilegios sobre tablas necesarias para los usuarios.

C Aplicar los parches de seguridad de manera oportuna.

D Modificar todas las contraseñas predeterminadas.

E Emplear contraseñas sencillas y fáciles de recordar.

A, **C** y **D** son las respuestas correctas.

11. El cifrado

A Debe utilizarse para todos los datos confidenciales.

B Siempre deben usar claves de cuando menos 28 bits de longitud.

C Debe aplicarse a los datos confidenciales enviados en una red.

D Puede emplear claves simétricas o asimétricas.

E Nunca debe usarse para credenciales de inicio de sesión.

A, **C** y **D** son las respuestas correctas.

12. ¿Cuáles son consideraciones de seguridad del cliente?

A Listas de direcciones MAC.

B Nivel de seguridad del navegador Web.

C Conceder sólo los privilegios sobre las tablas de una base de datos que sean absolutamente necesarios.

D Utilizar un rastreador de virus.

E Probar los riesgos de una aplicación.

B, **D** y **E** son las respuestas correctas.

13. En Microsoft SQL Server, un inicio de sesión de usuario

A Se puede conectar a cualquier cantidad de bases de datos.

B Automáticamente tiene privilegios de acceso a una base de datos.

C Puede utilizar autentificación de Windows.

D Se puede autentificar mediante Microsoft SQL Server.

E Posee un esquema de base de datos.

A, **C** y **D** son las respuestas correctas.

14. En Microsoft SQL Server, una base de datos

A Se vuelve una propiedad mediante un inicio de sesión.

B Puede tener uno o más usuarios asignados a ella.

C Puede contener datos del sistema (por ejemplo, datos maestros) o datos de usuarios (aplicaciones).

D Puede tener privilegios concedidos.

E Es un conjunto lógico de objetos de base de datos.

B, **C** y **E** son las respuestas correctas.

15. En Oracle, una cuenta de usuario

A Se puede conectar a cualquier cantidad de bases de datos.

B Automáticamente tiene privilegios de base de datos.

C Puede emplear autentificación por parte del sistema operativo.

D Se puede autentificar mediante el DBMS de Oracle.

E Es propietaria de un esquema de base de datos.

B, **C**, **D** y **E** son las respuestas correctas.

16. En Oracle, una base de datos

A Es propiedad de un usuario.

B Puede tener una o más cuentas de usuarios definidas en ella.

C Puede contener datos del sistema (por ejemplo, un esquema del sistema) y datos de usuarios (aplicaciones).

D Es igual que un esquema.

E Es administrada por una instancia de Oracle.

B, **C** y **E** son las respuestas correctas.

17. Los privilegios de sistema

A Son concedidos de manera similar en Oracle, Sybase ASE y Microsoft SQL Server.

B Son específicos para un objeto de una base de datos.

C Permiten al concesionario efectuar ciertas funciones administrativas en el servidor, como apagarlo.

D Son rescindidos mediante la instrucción REMOVE de SQL.

E Varían entre las bases de datos de diferentes vendedores.

A, **C** y **E** son las respuestas correctas.

18. Los privilegios de objetos

A Son concedidos de manera similar en Oracle, Sybase ASE y Microsoft SQL Server.

B Son específicos para un objeto de una base de datos.

C Permiten al concesionario efectuar ciertas funciones administrativas en el servidor, como apagarlo.

D Son rescindidos mediante la instrucción REMOVE de SQL.

E Son concedidos mediante la instrucción GRANT de SQL.

A, **B** y **E** son las respuestas correctas.

19. El uso de WITH GRANT OPTION cuando se conceden privilegios de objetos

A Permite al concesionario conceder privilegios a otras personas.

B Otorga al concesionario privilegios de DBA sobre toda la base de datos.

C Puede provocar problemas de seguridad.

D Generará una eliminación en cascada, si después se revocan los privilegios.

E Es una práctica muy recomendada porque es muy conveniente utilizarla.

A, **C** y **D** son las respuestas correctas.

20. Las vistas ayudan a la implementación de una política de seguridad al

A Limitar las columnas de una tabla a las que tiene acceso un usuario.

B Limitar las bases de datos a las que tiene acceso un usuario.

C Limitar las filas de una tabla a las que tiene acceso un usuario.

D Almacenar los resultados de una auditoría de la base de datos.

E Vigilar las intrusiones en una base de datos.

A y **C** son las respuestas correctas.

Capítulo 11: Implementación de las bases de datos

1. Un cursor es _____.

un apuntador a un grupo de resultados

2. Un grupo de resultados es _____.

el conjunto de filas devueltas por una consulta a una base de datos

3. La *I* en las siglas ACID significa _____.

independencia

4. Antes que sea posible buscar filas con un cursor, el cursor debe

A Declararse.

B Consignarse.

C Abrirse.

D Cerrarse.

E Purgarse.

A y **C** son las respuestas correctas.

5. Una transacción

A Puede procesarse y consignarse de manera parcial.

B No puede procesarse y consignarse de modo parcial.

C Cambia la base de datos de un estado consistente a otro.

D A veces se le denomina unidad de trabajo.

E Posee las propiedades descritas mediante las siglas ACID.

B, **C**, **D** y **E** son las respuestas correctas.

6. Microsoft SQL Server permite los modos de transacciones siguientes:

A Confirmación automática.

B Automático.

C Durable.

D Explícito.

E Implícito.

A, **D** y **E** son las respuestas correctas.

7. Oracle permite los modos de transacciones siguientes:

A Confirmación automática.

B Automático.

C Durable.

D Explícito.

E Implícito.

A y **E** son las respuestas correctas.

8. Las instrucciones SQL (comandos) que concluyen una transacción son

A SET AUTOCOMMIT.

B BEGIN TRANSACTION (en SQL Server).

C COMMIT.

D ROLLBACK.

E SAVEPOINT.

C y **D** son las respuestas correctas.

9. El problema de actualización concurrente

A Es una consecuencia de compartir datos de manera simultánea.

B No puede ocurrir cuando AUTOCOMMIT se establece en ON.

C Es la razón de que deba permitirse el bloqueo de transacciones.

D Ocurre cuando dos usuarios de una base de datos emiten instrucciones SELECT en conflicto.

E Ocurre cuando dos usuarios de una base de datos hacen actualizaciones en conflicto sobre los mismos datos.

A, **C** y **E** son las respuestas correctas.

10. Un bloqueo

A Es un control aplicado a los datos para reservarlos, de modo que el usuario pueda actualizarlos.

B Se suele liberar cuando ocurre una instrucción COMMIT o ROLLBACK.

C En DB2 y otros productos RDBMS, tiene establecido un tiempo de expiración.

D Puede provocar contiendas cuando otros usuarios intentan actualizar los datos bloqueados.

E Puede tener varios niveles y un protocolo de escala en algunos productos RDBMS.

A, **B**, **C**, **D** y **E** son las respuestas correctas.

11. Un bloqueo mutuo

A Es un bloqueo cuyo tiempo ha expirado y, por lo tanto, ya no es necesario.

B Ocurre cuando dos usuarios de una base de datos solicitan un bloqueo sobre datos que están bloqueados por el otro.

C En teoría puede poner a dos o más usuarios en un interminable estado de espera debido a bloqueo.

D En algunos RDBMS, se resuelve mediante detección de bloqueos mutuos.

E En algunos RDBMS, se resuelve mediante un tiempo de expiración de bloqueo.

B, **C**, **D** y **E** son las respuestas correctas.

12. La afinación del rendimiento

A Es un proceso que nunca concluye.

B Debe utilizarse en cada consulta hasta que no pueda realizarse ninguna mejora.

C Debe aplicarse sólo a consultas que no se apegan a los requisitos de rendimiento.

D Implica no sólo afinar el SQL, sino también la CPU, las entradas/salida del sistema de archivos, y el uso de la memoria.

E Debe basarse en los requisitos.

A, **C**, **D** y **E** son las respuestas correctas.

13. La afinación de una consulta SQL

A Se efectúa del mismo modo en todos los sistemas de base de datos relacional.

B Suele implicar el uso de una opción de plan de explicación.

C Siempre conlleva la colocación de instrucciones SQL en un procedimiento almacenado.

D Sólo se aplica a las instrucciones SELECT de SQL.

E Requiere un conocimiento detallado del RDBMS en que se va a ejecutar la consulta.

B y **E** son las respuestas correctas.

14. ¿Cuáles son sugerencias generales para afinar el SQL?

A Evitar exploraciones de tablas grandes.

B Emplear un índice cada vez que sea posible.

C Incluir una cláusula ORDER BY cada vez que sea posible.

D Utilizar una cláusula WHERE para filtrar las filas cada vez que sea posible.

E Usar vistas cada vez que sea posible.

A, **B** y **D** son las respuestas correctas.

15. Las prácticas de SQL que vuelven obvio el uso de un índice son

A El uso de la cláusula WHERE.

B La inclusión de un operador NOT.

C La aplicación de combinaciones de tablas.

D La utilización del operador NOT EQUAL.

E El uso de comodines en la primera columna de las cadenas de comparación mediante LIKE.

B, **D** y **E** son las respuestas correctas.

16. Los índices funcionan bien para filtrar filas cuando

A Son muy selectivos.

B El cociente de selectividad es muy alto.

C El cociente de selectividad es muy bajo.

D Son únicos.

E No son únicos.

A, **B** y **D** son las respuestas correctas.

17. Las principales consideraciones relacionadas con rendimiento de las instrucciones INSERT son

A La expansión de filas.

B El mantenimiento de índices.

C El uso del espacio libre.

D La afinación de las consultas secundarias.

E Las tablas muy grandes relacionadas.

B y **C** son las respuestas correctas.

18. Las principales consideraciones relacionadas con el rendimiento de las instrucciones UPDATE son

A La expansión de filas.

B El mantenimiento de índices.

C El uso del espacio libre.

D La afinación de consultas secundarias.

E Las tablas muy grandes relacionadas.

A y **B** son las respuestas correctas.

19. Un proceso de control de cambios

A Evita que los errores de programación sean aplicados en el entorno de producción.

B También se denomina administración de cambios.

C Ayuda a comprender cuándo pueden instalarse cambios.

D Proporciona un registro de todos los cambios realizados.

E Permite que se descarten versiones de software defectuosas.

B, **C**, **D** y **E** son las respuestas correctas.

20. ¿Cuáles son características comunes de los procesos de control de cambios?

A Soporte a transacciones.

B Numeración de versión.

C Prevención de bloqueo mutuo.

D Numeración de lanzamiento.

E Priorización.

B, **D** y **E** son las respuestas correctas.

Capítulo 12: Bases de datos para procesamiento analítico en línea

1. **Las bases de datos OLTP están diseñadas para manejar volúmenes _____ de transacciones.**

   altos

2. **Las consultas OLAP suelen acceder a cantidades _____ de datos.**

   grandes

3. **Comparados con los sistemas OLTP, los sistemas de almacén de datos tienden a tener consultas _____.**

   más prolongadas

4. **El pionero en los almacenes de datos fue _____.**

   Bill Inmon

5. **Al proceso de pasar de datos más resumidos a datos más detallados se le conoce como _____.**

   profundización

6. **El esquema de copo de nieve permite que las dimensiones tengan _____.**

   dimensiones propias

7. **El esquema de copo de estrella es un híbrido que contiene dimensiones _____ y _____.**

   normalizadas, no normalizadas

8. **Un almacén de datos**

   A Está orientado a temas.

   B Se integra a partir de varios orígenes de datos.

   C Varía con el tiempo.

   D Se actualiza en tiempo real.

   E Se organiza alrededor de un departamento o una función empresarial.

   A, **B** y **C** son las respuestas correctas.

9. **Algunos desafíos del método de almacén de datos son**

   A La actualización de los datos operativos del almacén de datos.

   B La subestimación de los recursos requeridos.

   C La disminución de las demandas de los usuarios.

   D Son proyectos grandes y complejos.

   E Las demandas de recursos son altas.

   B, **D** y **E** son las respuestas correctas.

10. La arquitectura de tabla de resumen

A Fue desarrollada originalmente por Bill Inmon.

B Incluye una tabla de hechos.

C Incluye tablas de dimensiones.

D Incluye tablas poco y muy resumidas.

E Debe incluir metadatos.

A, **D** y **E** son las respuestas correctas.

11. El esquema de estrella

A Fue desarrollado por Ralph Kimball.

B Incluye una tabla de dimensiones y una o más tablas de hechos.

C Siempre tiene tablas de dimensiones completamente normalizadas.

D Era una función clave del DBMS Red Brick.

E Incorpora varios niveles de tablas de dimensiones.

A y **D** son las respuestas correctas.

12. Algunos factores que deben considerarse cuando se diseña una tabla de hechos son

A Agregar columnas a la tabla de hechos.

B Reducir el tamaño de las columnas entre las tablas de origen y de hechos.

C Particionar la tabla de hechos.

D Con cuánta frecuencia debe actualizarse.

E Cuánto historial debe contener.

B, **C**, **D** y **E** son las respuestas correctas.

13. Las bases de datos multidimensionales

A Emplean una tabla de hechos completamente normalizada.

B Se visualizan mejor como cubos.

C Tienen tablas de dimensiones completamente normalizadas.

D En ocasiones se les denomina bases de datos MOLAP.

E Alojan más de tres dimensiones al repetir cubos para cada dimensión adicional.

B, **D** y **E** son las respuestas correctas.

14. Un mercado de datos

A Es un subconjunto de un almacén de datos.

B Es una tienda que vende datos a personas y empresas.

C Da soporte a los requisitos de un departamento o una función empresarial específicos.

D Puede ser un buen punto inicial para organizaciones sin experiencia en almacenes de datos.

E Puede ser un buen punto inicial cuando los requisitos son imprecisos.

A, **C**, **D** y **E** son las respuestas correctas.

15. Algunas razones para crear un mercado de datos son

A Es más pormenorizado que un almacén de datos.

B Es un proyecto con menores posibilidades de riesgo.

C Los datos se ajustan a un departamento o una función empresarial específicos.

D Contiene más datos que un almacén de datos.

E El proyecto tiene un costo general más bajo que un proyecto de almacén de datos.

B, **C** y **E** son las respuestas correctas.

16. Crear primero un almacén de datos, seguido por mercados de datos

A Retrasará el despliegue de un mercado de datos si se vuelve lento el proyecto de almacén de datos.

B Tiene un riesgo menor que tratar de crearlos juntos.

C Tiene el menor riesgo de las tres estrategias posibles.

D Tiene el mayor riesgo de las tres estrategias posibles.

E Tal vez necesite mucha repetición de trabajo.

A y **B** son las respuestas correctas.

17. Crear primero uno o más mercados de datos, seguidos por el almacén de datos

A Retrasará la entrega del almacén de datos si se vuelven lentos los proyectos de mercado de datos.

B Tiene el potencial para entregar más rápido algunas funciones OLAP.

C Tiene el menor riesgo de las tres estrategias posibles.

D Tiene el mayor riesgo de las tres estrategias posibles.

E Tal vez necesite mucha repetición de trabajo.

B, **C** y **E** son las respuestas correctas.

18. Crear el almacén de datos y los mercados de datos al mismo tiempo

A Genera el proyecto único más grande de todas las estrategias posibles.

B Tiene el potencial para tardar el máximo de tiempo para entregar cualquiera de las funciones OLAP.

C Tiene el menor riesgo de las tres estrategias posibles.

D Tiene el mayor riesgo de las tres estrategias posibles.

E Tal vez necesite mucha repetición de trabajo.

A, **B** y **D** son las respuestas correctas.

19. La minería de datos

A Crea un almacén de datos a escala menor.

B Extrae relaciones de datos previamente desconocidas del almacén de datos.

C Puede tener éxito con cantidades de datos pequeñas.

D Es más útil cuando la organización es lo bastante ágil para aplicar una acción con base en la información.

E Suele requerir grandes volúmenes de datos para producir resultados precisos.

B, **D** y **E** son las respuestas correctas.

20. Algunas propiedades de los sistemas de almacén de datos son

A Contienen información histórica en lugar de actual.

B Incluyen consultas de ejecución prolongada que procesan muchas filas de datos.

C Permiten las operaciones cotidianas.

D Están orientados a los procesos.

E Su volumen de transacciones va de mediano a bajo.

A, **B** y **E** son las respuestas correctas.

Capítulo 13: Integración de documentos y objetos XML en bases de datos

1. XML es _____.

un lenguaje de marcado de propósito general utilizado para describir datos.

2. ¿Cómo varían las bases de datos de SQL y los documentos de XML en cuanto a la estructura de los datos?

El XML define una secuencia y una estructura de árbol jerárquica, en tanto que el SQL no lo hace.

3. Si dos organizaciones emplean XML, ¿esto significa que tienen un modo estándar para intercambiar datos sin tener que crear un software de interfaz?

No necesariamente. Las dos organizaciones deben emplear una definición común para los elementos dentro de los documentos de XML que se van a intercambiar, antes de que puedan procesarse sin la necesidad de efectuar ninguna interpretación o conversión.

4. Los modificadores de tipo secundario válidos para el modificador de tipo SEQUENCE son _____.

que no puede tener un modificador de tipo secundario.

5. El tipo de esquema XML _____ se convierte a partir del tipo de datos NUMERIC de SQL.

xs:decimal

6. El tipo de esquema XML _____ se convierte a partir del tipo de datos DATE de SQL.

xs:date

7. Las dos maneras en que los valores nulos de la base de datos se representan en SQL/XML son _____ y _____.

un elemento ausente, **xsi:nil="true"**

8. ¿Cuáles son usos comunes de XML?

A Desplegar los datos de una base de datos en una página Web.

B Crear páginas Web estáticas.

C Transmitir los datos de una base de datos a otra persona.

D Imponer las reglas de negocios sobre los documentos.

A y **C** son las respuestas correctas.

9. ¿Cuáles son modificadores de tipo válidos para el tipo de datos XML?

A DOCUMENT.

B SEQUENCE.

C SQLXML.

D CONTENT.

A, **B** y **D** son las respuestas correctas.

10. ¿Cuáles función de SQL/XML crea un elemento basado en una columna de una tabla?

A XMLQUERY.

B XMLELEMENT.

C XMLFOREST.

D XMLDOCUMENT.

E XMLPARSE.

C es la respuesta correcta.

11. La programación orientada a objetos

A Emplea mensajes como vehículo para la interacción con los objetos.

B Permite a un objeto acceder directamente a las variables en un objeto relacionado.

C Utiliza métodos para definir el comportamiento de un objeto.

D Requiere que los objetos tengan una clave principal.

E Permite el uso de objetos complejos.

A, **C** y **E** son las respuestas correctas.

12. Las aplicaciones orientadas a objetos

A Requieren el uso de una base de datos orientada a objetos.

B Están escritas en un lenguaje orientado a objetos.

C Emplean entornos de desarrollo que suelen incluir clases predefinidas.

D Emplean entornos de desarrollo que suelen incluir métodos predefinidos.

E Pueden escribirse en el lenguaje de programación C.

B, **C** y **D** son las respuestas correctas.

13. Smalltalk

A Fue desarrollado por Linus Torvalds.

B Fue desarrollado en 1972.

C Fue desarrollado en la instalación PARC de Xerox.

D Está basado en el lenguaje de programación C.

E Fue el primer lenguaje de programación orientada a objetos que incluye un sistema de ventanas y el uso de un ratón.

B, **C** y **E** son las respuestas correctas.

14. C++

A Fue desarrollado por Alan Kay.

B Fue desarrollado en 1976.

C Fue desarrollado en AT&T Bell Laboratories.

D Está basado en el lenguaje de programación Java.

E Permite a los programadores ignorar el paradigma de objetos, si lo prefieren.

C y **E** son las respuestas correctas.

15. Java

A Fue desarrollado por Sun Microsystems.

B Sólo puede ejecutarse en sistemas grandes con mucha memoria.

C Fue desarrollado alrededor de 1995.

D Es un lenguaje intérprete.

E Es un lenguaje orientado a objetos de propósito general.

A, **C**, **D** y **E** son las respuestas correctas.

16. La persistencia de objetos

A Conserva el estado de un objeto entre ejecuciones de una aplicación.

B Conserva el estado de un objeto tras el apagado y encendido de una computadora.

C Carga los objetos en la memoria para preservarlos de manera permanente.

D Ocurre cuando la aplicación solicita que se guarde un objeto.

E Sólo se consigue con una base de datos orientada a objetos.

A, **B** y **D** son las respuestas correctas.

17. Algunos eventos necesarios para recuperar un objeto de una base de datos orientada a objetos son

A Se envía un mensaje al objeto, de modo que éste debe cargarse en la memoria.

B Se envía a la base de datos orientada a objetos una solicitud para recuperar el objeto.

C Las referencias a objetos se revuelven en direcciones de la memoria.

D Se asignan datos relacionales a una clase de objetos.

E El objeto queda disponible para el entorno de la aplicación.

A, **B**, **C** y **E** son las respuestas correctas.

18. Algunas ventajas de las bases de datos de objetos-relacionales son

A Los objetos se guardan dentro de las tablas.

B Se permiten tipos de datos complejos.

C Permiten por completo la capacidad de consultas ad hoc.

D Permiten por completo las estructuras de clases y la herencia.

E Las funciones (métodos) guardadas de manera central mejoran la reutilización.

B, **C** y **E** son las respuestas correctas.

19. Entre las desventajas de las bases de datos de objetos-relacionales están

A La combinación es más compleja que en las bases de datos orientadas a objetos puras o en las relacionales puras.

B Es limitada la capacidad para consultas ad hoc.

C Los objetos están centrados en las tablas.

D Ni los puristas relacionales ni los de objetos están enamorados de la combinación.

E Las aplicaciones de objetos no están tan centradas en los datos como las relacionales.

A, **C**, **D** y **E** son las respuestas correctas.

20. Al considerar la elección de un modelo de base de datos, ¿cuáles de los hechos siguientes deben tomarse en cuenta?

A Los archivos de un sistema de archivos comunes pueden manejar los datos sencillos, siempre y cuando no existan requisitos para consultas ad hoc.

B Las bases de datos relacionales pueden manejar los datos sencillos que tienen requisitos para consultas ad hoc.

C Las bases de datos orientadas a objetos son mejores para manejar datos complejos.

D Las bases de datos de objetos-relacionales pueden manejar datos complejos que tienen requisitos de consultas ad hoc.

E Las bases de datos orientadas a objetos pueden manejar datos complejos, siempre y cuando no haya requisitos de consultas ad hoc.

A, **B**, **C**, **D** y **E** son las respuestas correctas.

Apéndice B

Soluciones a los ejercicios Pruebe esto

Este apéndice contiene las soluciones a los ejercicios Pruebe esto incluidos en diferentes capítulos. En cualquier actividad de diseño, existen muchas soluciones posibles. En este aspecto, ocurre lo mismo con el diseño de una base de datos: no es una ciencia exacta y, por lo tanto, hay cierta libertad en las soluciones alternas. Si su solución es diferente a ésta, no necesariamente significa que ella sea incorrecta, siempre y cuando cumpla con los requisitos establecidos en el ejercicio.

Los archivos de origen (en inglés) de estas soluciones también pueden descargarse del sitio Web de McGraw-Hill Educación. Siga estos pasos para descargar los archivos:

1. Abra su navegador Web y escriba www.mcgraw-hill-educacion.com. Seleccione su país.
2. En el cuadro Buscar, en la parte superior de la página, escriba Oppel Fundamentos de bases de datos y haga clic en Buscar.
3. Haga clic en la imagen mostrada del libro para mostrar su página de información.
4. Los vínculos para las descargas están bajo la imagen del libro al margen izquierdo. Seleccione los archivos que necesite y haga clic para abrirlos o haga clic con el botón secundario del ratón para guardarlos en su computadora.

Pruebe esto 5-1 Solución: Proyecto de tareas de administración de una base de datos

Fase del proyecto	Tarea
Planeación	w. Evaluación de las opciones disponibles de DBMS.
Recopilación de requisitos	f. Determinación de las vistas requeridas por los usuarios de negocios.
	r. Identificación de los atributos requeridos por los usuarios de negocios.
	t. Identificación y documentación de los requisitos de datos empresariales.
Diseño conceptual	l. Especificación de un nombre lógico para cada entidad y atributo.
	q. Documentación de las reglas de negocios que no se pueden representar en el modelo de datos.
	s. Identificación de las relaciones entre las entidades.
Diseño lógico	a. Normalización.
	b. Adición de claves externas a la base de datos.
	d. Especificación del identificador único de cada relación.
	g. Eliminación de los datos que se derivan con facilidad.
	h. Resolución de las relaciones varios a varios.
	l. Especificación de un nombre lógico para cada entidad y atributo.
	p. Traducción del modelo conceptual de datos a un modelo lógico.
	q. Documentación de las reglas de negocios que no se pueden representar en el modelo de datos.
	s. Identificar las relaciones entre las entidades.

Fase del proyecto	Tarea
Diseño físico	c. Especificación de la ubicación física de los objetos de la base de datos en los medios de almacenamiento.
	e. Establecimiento de la clave principal a cada tabla.
	j. Modificación de la base de datos para cumplir los requisitos de negocios.
	m. Asignación de un nombre físico a cada tabla y columna.
	o. Especificación de índices de la base de datos.
Construcción	i. Definición de las vistas en la base de datos.
	u. Aseguramiento de que se cumplan los requisitos de datos de usuarios.
Implementación y soporte	k. Desnormalización la base de datos para mejorar el rendimiento.
	n. Adición de datos que se puedan derivar para mejorar el rendimiento.
	v. Afinación de la base de datos para mejorar el desempeño.
Soporte continuo	j. Modificación de la base de datos para cumplir los requisitos de negocios.
	k. Desnormalización la base de datos para mejorar el rendimiento.
	n. Adición de datos que se puedan derivar para mejorar el rendimiento.
	u. Aseguramiento de que se cumplan los requisitos de datos de usuarios.
	v. Afinación de la base de datos para mejorar el desempeño.

Pruebe esto 6-1 Solución: Registros académicos en UATL

Éstas son las relaciones normalizadas para el ejercicio Pruebe esto 6-1, en donde los atributos de clave principal se indican mediante (**CP**):

```
CURSO: ID DE CURSO (CP), TÍTULO, DESCRIPCIÓN, NÚMERO DE CRÉDITOS

INSTRUCTOR: ID DE INSTRUCTOR (CP), NOMBRE DE INSTRUCTOR, CALLE DE
            DIRECCIÓN DE CASA, CIUDAD DE DIRECCIÓN DE CASA, ESTADO DE
            DIRECCIÓN DE CASA,CÓDIGO DE DIRECCIÓN DE CASA, TELÉFONO DE
            CASA, TELÉFONO DE OFICINA

SECCIÓN DE CURSO: ID DE SECCIÓN (CP), AÑO CALENDARIO, SEMESTRE, ID DE
                  CURSO,
                  EDIFICIO, AULA, DÍA DE CLASE, HORA DE CLASE,
                  ID DE INSTRUCTOR

ESTUDIANTE: ID DE ESTUDIANTE (CP), NOMBRE DE ESTUDIANTE, DIRECCIÓN DE
            CASA,CIUDAD DE DIRECCIÓN DE CASA, ESTADO DE DIRECCIÓN DE
            CASA,CÓDIGO DE DIRECCIÓN DE CASA, TELÉFONO DE CASA

SECCIÓN DE ESTUDIANTE: ID DE ESTUDIANTE (CP), ID DE SECCIÓN (CP),
                       CALIFICACIÓN

PRERREQUISITO DE CURSOS: ID DE CURSO (CP), ID DE CURSO DE PRERREQUISITO (CP)

INSTRUCTOR CALIFICADO DE CURSO: ID DE INSTRUCTOR (CP), ID DE CURSO (CP)
```

Es conveniente incluir algunas precisiones sobre esta solución específica:

- No existe una clave natural sencilla para la relación Sección de cursos, por lo que se agregó una clave sustituta.
- La relación Curso de prerrequisito puede ser confusa. Ésta es la relación de intersección para una relación recursiva varios a varios. Un curso puede tener varios prerrequisitos, que se encuentran al combinar ID DE CURSO de la relación CURSO con ID DE CURSO de la relación PRERREQUISITO DE CURSOS. Al mismo tiempo, cualquier curso puede ser un prerrequisito para muchos otros. Éstos se encuentran al combinar ID DE CURSO de la relación CURSO con ID DE CURSO DE PRERREQUISITO de la relación PRERREQUISITO DE CURSOS. Esto significa que existen *dos* relaciones entre CURSO y PRERREQUISITO DE CURSOS: una en que ID DE CURSO es la clave externa y otra en que ID DE CURSO DE PRERREQUISITO es la clave externa. La comparación de las ilustraciones de las tablas CURSO y PRERREQUISITO CURSO sirve para aclarar este punto.

Como ayuda para que visualice cómo funciona todo, las ilustraciones siguientes presentan cada una de las tablas implementadas en una base de datos de Microsoft Access, cada una cargada con los datos de los ejemplos de la vista (informe) del usuario original. La última ilustración muestra el ERD de la solución, que utiliza el panel Relaciones de Microsoft Access como medio de presentación.

Tabla CURSO:

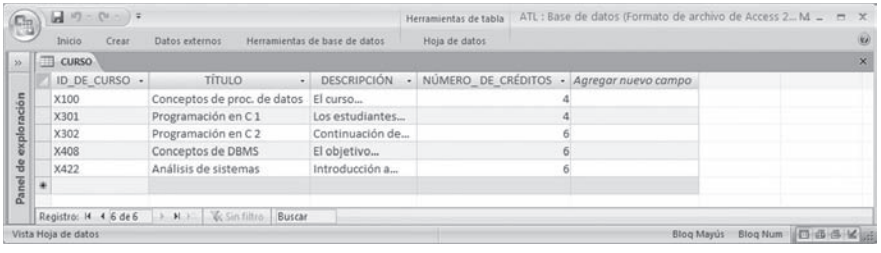


Tabla INSTRUCTOR:

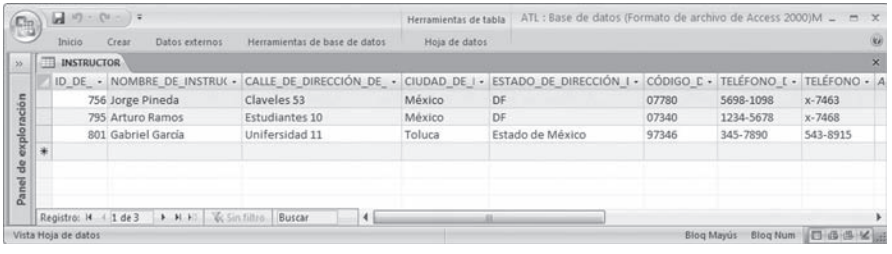


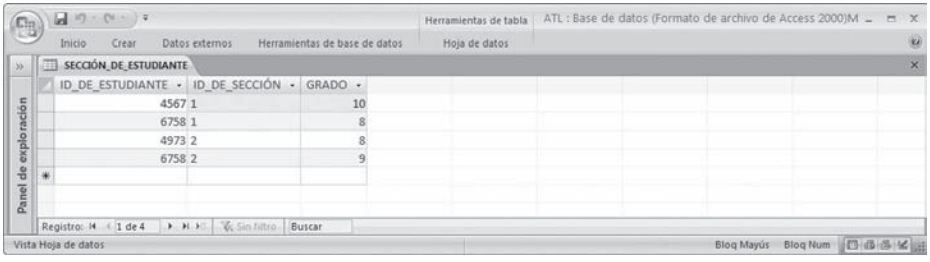
Tabla SECCIÓN CURSO:

ID_DE_SECCIÓN	AÑO_CALENDARIO	SEMESTRE	ID_DE_CURSO	EDIFICIO	AULA	DÍA_DE_CLASE	HORA_DE_C	ID_DE_INSTRUC
1	2010	Spr	X408	Facultad	70	Mar	7-10	756
2	2010	Spr	X408	Anexo	7	Mie	7-10	756
3	2010	Spr	X100	Facultad	70	Mar, Vie	7-9	801
								0

Tabla ESTUDIANTE:

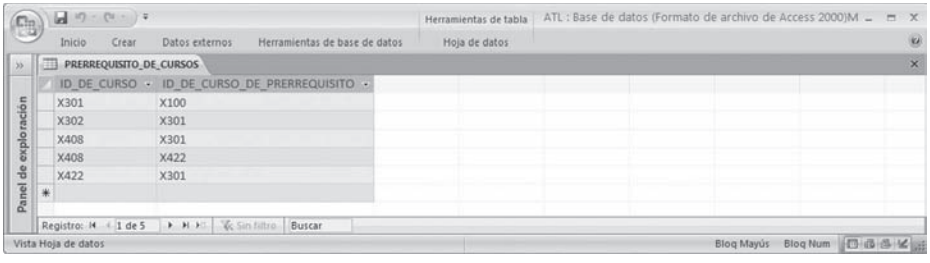
ID_DE_ESTUD	NOMBRE_DE_ESTUDIANTE	DIRECCIÓN_DE_CASA	CIUDAD_DE_DIRECCIÓN	ESTADO_DE_DIRECC	CÓDIGO_DE	TELÉFONO_DE_C
4567	Juan Pérez	Av. Laurel 38	México	DF	07780	5348-1581
4973	Arturo López	Estelar 43	México	DF	07340	5643-1790
6758	Ricardo Juárez	Calle 3 51	México	DF	07893	6789-0312

Tabla SECCIÓN ESTUDIANTE:



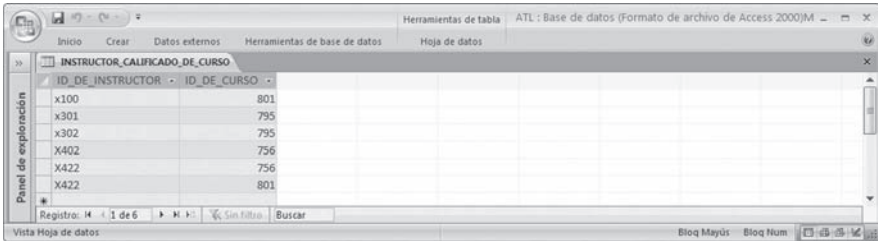
ID_DE_ESTUDIANTE	ID_DE_SECCIÓN	GRADO
4567	1	10
6758	1	8
4973	2	8
6758	2	9

Tabla PRERREQUISITO CURSO:

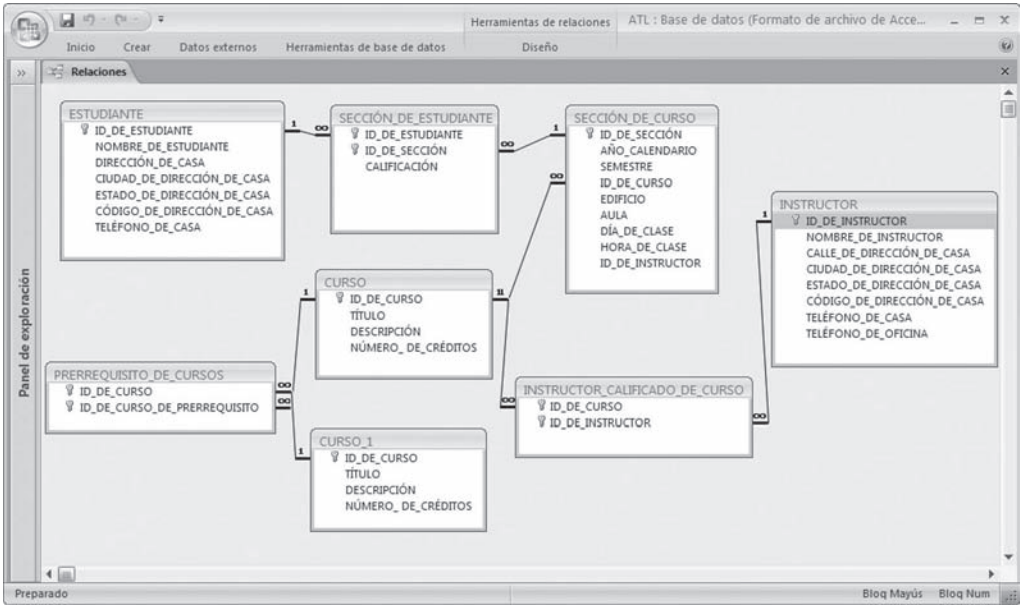


ID_DE_CURSO	ID_DE_CURSO_DE_PRERREQUISITO
X301	X100
X302	X301
X408	X301
X408	X422
X422	X301

Tabla INSTRUCTOR CALIFICADO CURSO:



ERD para la Universidad ATL:



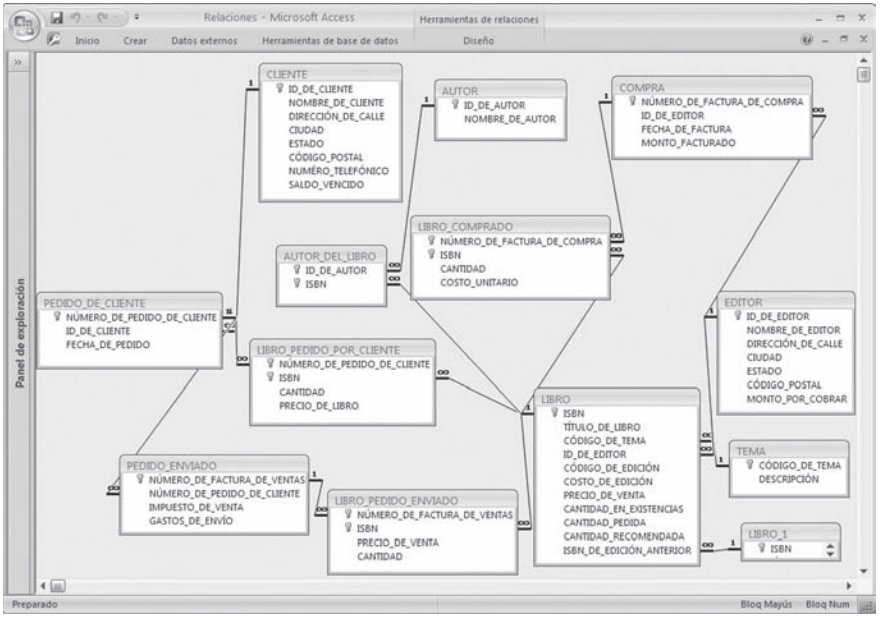
Pruebe esto 6-2 Solución: Compañía de Libros de Computación

Éstas son las relaciones normalizadas para el ejercicio Pruebe esto 6-2, en donde los atributos de clave principal se indican mediante (**CP**):

```
LIBRO: ISBN (CP), TÍTULO DE LIBRO, CÓDIGO DE TEMA, ID DE EDITOR,
       CÓDIGO DE EDICIÓN, COSTO DE EDICIÓN, PRECIO DE VENTA,
       CANTIDAD EN EXISTENCIAS, CANTIDAD PEDIDA,
       CANTIDAD RECOMENDADA, ISBN DE EDICIÓN ANTERIOR
```

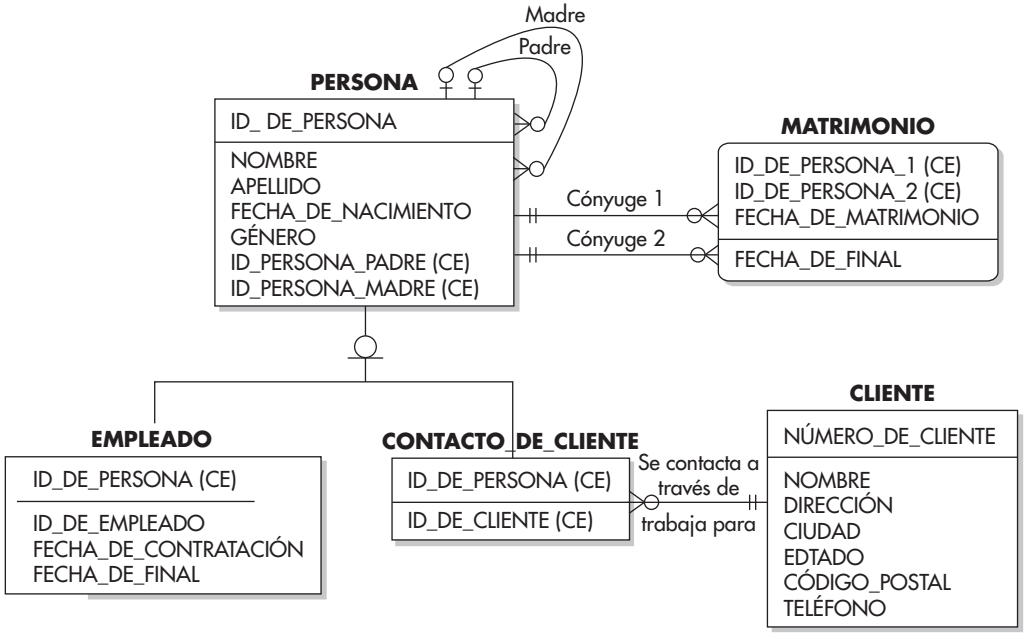
```
PEDIDO DE CLIENTE: NÚMERO DE PEDIDO DE CLIENTE (CP), ID DE CLIENTE,
       FECHA DE PEDIDO

LIBRO PEDIDO POR CLIENTE: NÚMERO DE PEDIDO DE CLIENTE (CP), ISBN (CP),
       CANTIDAD, PRECIO DE LIBRO

TEMA: CÓDIGO DE TEMA (CP), DESCRIPCIÓN

AUTOR: ID DE AUTOR (CP), NOMBRE DE AUTOR
AUTOR DEL LIBRO: ID DE AUTOR (CP), ISBN (CP)

CLIENTE: ID DE CLIENTE (CP), NOMBRE DE CLIENTE, DIRECCIÓN DE CALLE,
       CIUDAD,

 ESTADO, CÓDIGO POSTAL, NUMÉRO TELEFÓNICO, SALDO VENCIDO

EDITOR: ID DE EDITOR (CP), NOMBRE DE EDITOR, DIRECCIÓN DE CALLE,
       CIUDAD, ESTADO, CÓDIGO POSTAL, MONTO POR COBRAR

PEDIDO ENVIADO (POR RECIBIR): NÚMERO DE FACTURA DE VENTAS (CP),
       NÚMERO DE PEDIDO DE CLIENTE, IMPUESTO DE VENTA, GASTOS DE ENVÍO

LIBRO PEDIDO ENVIADO: NÚMERO DE FACTURA DE VENTAS (CP), ISBN (CP),
       PRECIO DE VENTA, CANTIDAD

COMPRA (POR PAGAR): NÚMERO DE FACTURA DE COMPRA (CP), ID DE EDITOR,
       FECHA DE FACTURA, MONTO FACTURADO

LIBRO COMPRADO: NÚMERO DE FACTURA DE COMPRA (CP), ISBN (CP), CANTIDAD,
       COSTO UNITARIO
```

Éste es un ERD que muestra el diseño completo, implementado en Microsoft Access:



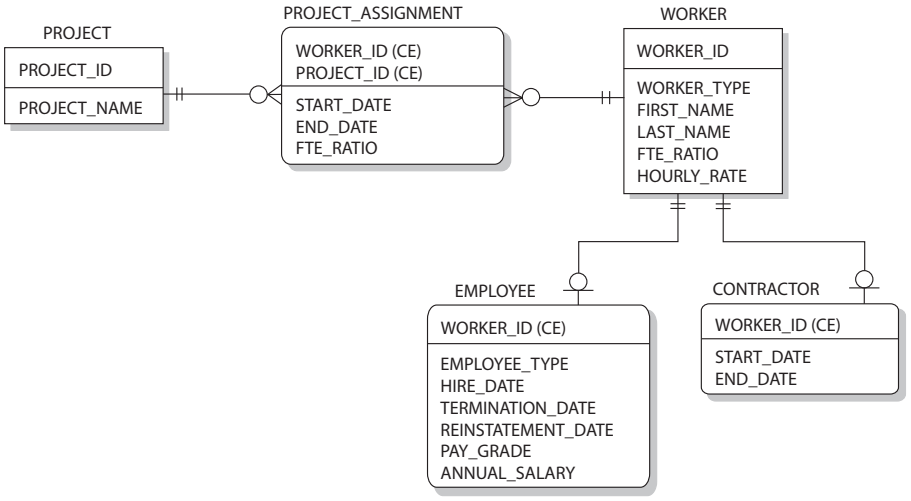
Pruebe esto 7-1 Solución: Dibujo de un ERD en un formato de ingeniería de la información (II)

A continuación se presenta una solución para el ejercicio Pruebe esto 7-1:



Pruebe esto 8-1 Solución: Ubicación de un modelo lógico en el diseño físico de una base de datos

Ésta es una solución posible para el ejercicio Pruebe esto 8-1:



Pruebe esto 10-1 Solución: Privilegios de los objetos de una base de datos

En seguida se presentan las instrucciones SQL para el ejercicio Pruebe esto 10-1:

```
CREATE TABLE DEPARTAMENTO
  (CÓDIGO_DEPARTAMENTO CHAR(3),
   NOMBRE_DEPARTAMENTO VARCHAR(50));

GRANT SELECT, INSERT ON DEPARTAMENTO TO USUARIO1;

INSERT INTO DATOS1.DEPARTAMENTO
VALUES ('001','Ejecutivo');

SELECT * FROM DATOS1.DEPARTAMENTO
 WHERE CÓDIGO_DEPARTAMENTO = '001';

DELETE FROM DATOS1.DEPARTAMENTO
 WHERE CÓDIGO_DEPARTAMENTO = '001';
```

```
DROP TABLE DATOS1.DEPARTAMENTO;

DROP TABLE DEPARTAMENTO;
```

Pruebe esto 11-1 Solución: Soporte a transacciones en SQL

Éstas son las instrucciones del SQL utilizadas en el ejercicio Pruebe esto 11-1:

```
DROP TABLE DEPARTAMENTO;

CREATE TABLE DEPARTAMENTO
  (CÓDIGO_DEPARTAMENTO CHAR(3),
   NOMBRE_DEPARTAMENTO VARCHAR(50));

SET IMPLICIT_TRANSACTIONS ON

INSERT INTO DEPARTAMENTO
VALUES ('001','Ejecutivo');

SELECT * FROM DEPARTAMENTO;

ROLLBACK;

INSERT INTO DEPARTAMENTO
VALUES ('001','Ejecutivo');

COMMIT;

ROLLBACK;

SELECT * FROM DEPARTAMENTO;

DROP TABLE DEPARTAMENTO;
COMMIT;
```

Pruebe esto 12-1 Solución: Diseño de tablas de hechos y de dimensiones para esquema de estrella

A continuación se exhiben las tablas de hechos y de dimensión diseñadas para el ejercicio Pruebe esto 12-1:

```
LIBRO (HECHO): ISBN (CP), CÓDIGO DE TEMA (CE), ID DE EDITOR (CE),
       COSTO DE EDICIÓN, PRECIO DE VENTA, CANTIDAD EN EXISTENCIA,
       CANDIDAD PEDIDA,CANTIDAD RECOMENDADA

TÍTULO DE LIBRO (DIMENSIÓN): TÍTULO DE LIBRO, CÓDIGO DE EDICIÓN,
       ISBN DE EDICIÓN ANTERIOR
```

```
TEMA (DIMENSIÓN): CÓDIGO DE TEMA (CP), DESCRIPCIÓN

AUTOR (DIMENSIÓN): ISBN (CP), ID DE AUTOR (PK), NOMBRE DE AUTOR

EDITOR (DIMENSIÓN): ID DE EDITOR (CP), NOMBRE DE EDITOR, DIRECCIÓN DE
       CALLE, CIUDAD, ESTADO, CÓDIGO POSTAL
```

Pruebe esto 13-1 Solución: Uso de las funciones de SQL/XML

Si emplea Oracle, a continuación se presenta el código SQL para crear una tabla Empleados con las columnas necesarias y para llenarlas con las tres filas requeridas para este ejercicio:

```
CREATE TABLE EMPLEADOS
  (ID_DE_EMPLEADO DECIMAL(6) NOT NULL,
   NOMBRE   VARCHAR(20),
   APELLIDO VARCHAR(20),
   NUMERO_TELEFONICO VARCHAR(20),
   ID_DE_DEPARTAMENTO DECIMAL(4));

INSERT INTO EMPLEADOS
  VALUES(100,'Juan','Campos','515.123.4567',90);
INSERT INTO EMPLEADOS
  VALUES(101,'Lilia','Corona','515.123.4568',90);
INSERT INTO EMPLEADOS
  VALUES(102,'Luis','Arana','515.123.4569',90);
COMMIT;
```

Éstas son las instrucciones SQL que se aplican al ejercicio Pruebe esto 13-1:

```
SELECT XMLELEMENT("Empleado",
       XMLATTRIBUTES(ID_DE_EMPLEADO AS ID),
       XMLFOREST(NOMBRE AS "Nombre",
                 APELLIDO AS "Apellido",
                 NUMERO_TELEFONICO AS "Tel."))
  FROM EMPLEADOS
 WHERE ID_DE_DEPARTAMENTO = 90
 ORDER BY ID_DE_EMPLEADO;
```

Índice

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z