

PROCESAMIENTO DE BASES DE DATOS

OCTAVA EDICIÓN



**FUNDAMENTOS,
DISEÑO E
IMPLEMENTACIÓN**

DAVID M. KROENKE

PEARSON
Prentice
Hall

®

PROCESAMIENTO DE BASES DE DATOS



PROCESAMIENTO DE BASES DE DATOS

Fundamentos, diseño e implementación

Octava edición

David M. Kroenke

TRADUCCIÓN

Dra. Ana Elizabeth García Hernández

*Escuela Superior de Ingeniería Química e Industrias Extractivas,
Instituto Politécnico Nacional*

REVISIÓN TÉCNICA

M. en C. Juan Raúl Esparza Martínez

*Departamento de Computación Básica,
Instituto Tecnológico y de Estudios Superiores de Monterrey, Campus Monterrey*



MÉXICO • ARGENTINA • BRASIL • COLOMBIA • COSTA RICA • CHILE
ESPAÑA • GUATEMALA • PERÚ • PUERTO RICO • VENEZUELA
www.detodoprogramacion.com

KROENKE, DAVID M.

Procesamiento de Bases de Datos
Fundamentos, diseño e implementación

PEARSON EDUCACIÓN, México, 2003

ISBN: 970-26-0325-0

Área: Universitario

Formato: 21 × 27 cm

Páginas: 688

Authorized translation from the English Language edition, entitled Database Processing: Fundamentals, Design & Implementation. Eighth Edition by David M. Kroenke, published by Pearson Education, Inc., publishing as PRENTICE HALL, INC., Copyright ©2002. All rights reserved.
ISBN 0-13-064839-6

Traducción autorizada de la edición en idioma inglés, titulada *Database Processing: Fundamentals, Design & Implementation, Eighth Edition*, por David M. Kroenke, publicada por Pearson Education, Inc., publicada como PRENTICE-HALL INC., Copyright © 2002.
Todos los derechos reservados

Esta edición en español es la única autorizada.

Edición en español

Editor: Guillermo Trujano Mendoza
e-mail: guillermo.trujano@pearsoned.com
Supervisor de desarrollo: Lorena Pontones
Supervisor de producción: Enrique Trejo Hernández

Edición en inglés

Publisher: Natalie E. Anderson
Executive Editor, MIS: Bob Horan
Associate Editor: Kyle Hannon

OCTAVA EDICIÓN, 2003

D.R. © 2003 por Pearson Educación de México, S.A. de C.V.
Atacomulco Núm. 500, 5º Piso
Industrial Atoto, Naucalpan de Juárez
C.P. 53519, Edo. de México

Cámara Nacional de la Industria Editorial Mexicana.
Reg. Núm. 1031.

Prentice Hall es una marca registrada de Pearson Educación de México, S.A. de C.V.

Reservados todos los derechos. Ni la totalidad ni parte de esta publicación pueden reproducirse, registrarse o transmitirse, por un sistema de recuperación de información, en ninguna forma ni por ningún medio, sea electrónico, mecánico, fotoquímico, magnético o electroóptico, por fotocopia, grabación o cualquier otro, sin permiso previo por escrito del editor.

El préstamo, alquiler o cualquier otra forma de cesión de uso de este ejemplar requerirá también la autorización del editor o de sus representantes.



ISBN 970-26-0325-0

Impreso en México. *Printed in Mexico.*

1 2 3 4 5 6 7 8 9 0 - 05 04 03 02

RESUMEN DE CONTENIDO



Prefacio xiii

PARTE I INTRODUCCIÓN I

- 1 Introducción al procesamiento de bases de datos 3
- 2 Introducción al desarrollo de una base de datos 25

PARTE II MODELACIÓN DE DATOS 49

- 3 El modelo entidad-relación 51
- 4 El modelo de objeto semántico 79

PARTE III DISEÑO DE BASES DE DATOS 119

- 5 El modelo relacional y la normalización 121
- 6 Diseño de bases de datos utilizando modelos de entidad-relación 151
- 7 Diseño de bases de datos con modelos de objeto semántico 183

PARTE IV IMPLEMENTACIÓN DE BASES DE DATOS CON EL MODELO RELACIONAL 209

- 8 Fundamentos de la implementación relacional 211
- 9 Lenguaje de consulta estructurado 235
- 10 Diseño de aplicaciones de bases de datos 257

PARTE V PROCESAMIENTO DE BASES DE DATOS MULTIUSUARIO 293

- 11 Administración de bases de datos multiusuario 295
- 12 Administración de bases de datos con Oracle 329
- 13 Manejo de bases de datos con SQL Server 2000 365

PARTE VI PROCESAMIENTO DE BASES DE DATOS EMPRESARIAL 405

- 14 Redes, arquitecturas multicapa y XML 407
- 15 ODBC, OLE DB, ADO y ASP 441
- 16 JDBC, páginas del Servidor Java y MySQL 481
- 17 Distribución de datos de la empresa 515

PARTE VII PROCESAMIENTO DE BASES DE DATOS ORIENTADA A OBJETOS 553

- 18 Procesamiento de Bases de Datos orientadas a objetos 555

Apéndice A Estructuras de datos para el procesamiento de bases de datos 585

Apéndice B Modelos de objeto semántico con Tabledesigner 607

Glosario 633

Bibliografía 655

CONTENIDO



Prefacio *xiii*

PARTE I INTRODUCCIÓN I

- 1 INTRODUCCIÓN AL PROCESAMIENTO DE BASES DE DATOS 3
 - Cuatro ejemplos de bases de datos 3
 - La relación de los programas de aplicaciones y los DBMS 11
 - Sistemas de procesamiento de archivos 12
 - Sistemas de procesamiento de base de datos 14
 - Definición de una base de datos 15
 - Historia del procesamiento de la base de datos 17
 - Resumen 21
 - Preguntas del grupo I 22
 - Proyectos 23
 - Preguntas del proyecto FiredUp 23

- 2 INTRODUCCIÓN AL DESARROLLO DE UNA BASE DE DATOS 25
 - La base de datos 25
 - DBMS 29
 - Creación de la base de datos 30
 - Componentes de aplicaciones 34
 - Procesos para el desarrollo de una base de datos 41
 - Resumen 44
 - Preguntas del grupo I 45
 - Preguntas del grupo II 46
 - Preguntas del proyecto FiredUp 46

PARTE II MODELACIÓN DE DATOS 49

- 3 EL MODELO ENTIDAD-RELACIÓN 51
 - Elementos del modelo entidad-relación 51
 - Diagramas de entidad-relación de estilo UML 62
 - Ejemplos 67
 - Bases de datos como modelos de modelos 73
 - Resumen 74
 - Preguntas del grupo I 75
 - Preguntas del grupo II 76
 - Proyectos 76
 - Preguntas del proyecto FiredUp 77

- 4 EL MODELO DE OBJETO SEMÁNTICO 79
 - Objetos semánticos 80
 - Creación de modelos de datos con objetos semánticos 86
 - Tipos de objetos 94
 - Comparación del objeto semántico con el modelo E-R 111
 - Resumen 113
 - Preguntas del grupo I 114
 - Preguntas del grupo II 115
 - Proyectos 115
 - Preguntas del proyecto FiredUp 116

PARTE III DISEÑO DE BASES DE DATOS 119

- 5 EL MODELO RELACIONAL Y LA NORMALIZACIÓN 121
 - El modelo relacional 122
 - Normalización 126
 - De la primera a la quinta formas normales 128
 - Forma normal dominio-llave 133
 - Síntesis de relaciones 139
 - Dependencias multivaluadas, iteración 2 142
 - Optimización 143
 - Resumen 145
 - Preguntas del grupo I 146
 - Preguntas del grupo II 147
 - Preguntas del proyecto FiredUp 150
- 6 DISEÑO DE BASES DE DATOS UTILIZANDO MODELOS DE ENTIDAD-RELACIÓN 151
 - Transformación de los modelos entidad-relación en diseños de bases de datos relacionales 151
 - Ejemplo de diseño 167
 - Árboles, redes y listas de materiales 169
 - Resumen 177
 - Preguntas del grupo I 178
 - Preguntas del grupo II 180
 - Proyectos 180
 - Preguntas del proyecto FiredUp 180
- 7 DISEÑO DE BASES DE DATOS CON MODELOS DE OBJETO SEMÁNTICO 183
 - Transformación de objetos semánticos en diseños de bases de datos relacionales 183
 - Objetos de muestra 198
 - Resumen 203
 - Preguntas del grupo I 204
 - Preguntas del grupo II 205
 - Proyectos 206
 - Preguntas del proyecto FiredUp 207

PARTE IV IMPLEMENTACIÓN DE BASES DE DATOS CON EL MODELO RELACIONAL 209

- 8 FUNDAMENTOS DE LA IMPLEMENTACIÓN RELACIONAL 211
 - Definición de datos relacionales 211
 - Manejo de datos relacionales 216
 - Álgebra relacional 221
 - Resumen 231
 - Preguntas del grupo I 231
- 9 LENGUAJE DE CONSULTA ESTRUCTURADO 235
 - Consulta de una sola tabla 236
 - Consultas de tablas múltiples 244
 - Exists y not exists 248
 - Cambio de datos 250
 - Resumen 252
 - Preguntas del grupo I 252
 - Preguntas del grupo II 254
 - Preguntas del proyecto FiredUp 254
- 10 DISEÑO DE APLICACIONES DE BASES DE DATOS 257
 - Funciones de una aplicación de base de datos 257
 - Caso de aplicación: Galería View Ridge 259
 - Crear, leer, actualizar y borrar instancias de vistas 262
 - Diseño de formas 269
 - Diseño de reportes 274
 - Cumplimiento de restricciones 277
 - Seguridad y control 286
 - Lógica de la aplicación 288
 - Resumen 288
 - Preguntas del grupo I 289
 - Preguntas del grupo II 291
 - Proyectos 291
 - Preguntas del proyecto FiredUp 292

PARTE V PROCESAMIENTO DE BASES DE DATOS MULTIUSUARIO 293

- 11 ADMINISTRACIÓN DE BASES DE DATOS MULTIUSUARIO 295
 - Administración de la base de datos 296
 - Control de concurrencia 298
 - Seguridad de la base de datos 311
 - Recuperación de la base de datos 318
 - Resumen 324
 - Preguntas del grupo I 325
 - Preguntas del grupo II 327

14 REDES, ARQUITECTURAS MULTICAPA Y XML 407
Entornos de Red 407
Arquitectura multicapa 410
Lenguajes de marcaje HTML y DHTML 415
XML: Lenguajes de marcaje extensible 417
Resumen 436
Preguntas del grupo I 437
Preguntas del grupo II 438
Preguntas del proyecto FiredUp 439

15 ODBC, OLE DB, ADO Y ASP 441
El ambiente de datos en un servidor de la red 441
Estándar de la conectividad abierta de una base de datos (ODBC) 444
OLE DB 449
ADO (Objetos de Datos Activos) 453
Ejemplos de ADO 458
Resumen 477
Preguntas del grupo I 478
Preguntas del grupo II 479
Preguntas del proyecto FireUp 480

16	JDBC, PÁGINAS DEL SERVIDOR JAVA Y MySQL	481
	JDBC	481
	Páginas del Servidor Java	491
	MySQL	503
	Resumen	509
	Preguntas del grupo I	510
	Preguntas del grupo II	512
	Proyectos	512
	Preguntas del proyecto FiredUp	513
17	DISTRIBUCIÓN DE DATOS DE LA EMPRESA	515
	Arquitecturas del procesamiento de la base de datos empresarial	515
	Descarga de datos	522
	Procesamiento analítico en línea (OLAP)	527
	Data warehouses	535
	Administración de datos	542
	Resumen	547
	Preguntas del grupo I	549
	Preguntas del grupo II	551

PARTE VII PROCESAMIENTO DE BASES DE DATOS ORIENTADA A OBJETOS 553

18	PROCESAMIENTO DE BASES DE DATOS ORIENTADAS A OBJETOS	555
	Esbozo de la programación orientada a objetos	556
	Ejemplo de OOP	557
	Persistencia de objetos	560
	Persistencia de objetos mediante Oracle	564
	Estándares ODBMS	572
	Resumen	581
	Preguntas del grupo I	582
	Preguntas del grupo II	583
APÉNDICE A	ESTRUCTURAS DE DATOS PARA EL PROCESAMIENTO DE BASES DE DATOS	585
	Archivos planos	585
	Representación de las relaciones binarias	593
	Representaciones de las llaves secundarias	601
	Resumen	604
	Preguntas del grupo I	605
	Preguntas del grupo II	606
APÉNDICE B	MODELOS DE OBJETO SEMÁNTICO CON TABLEDESIGNER	607
	Creación de un modelo de objeto semántico	609
	Un modelo SOM de ingeniería de reversa	617
	Edición de las vistas de bases de datos en la red	624
	Pasos siguientes	630
	Ejercicios	630

GLOSARIO	633
-----------------	------------

BIBLIOGRAFÍA	655
---------------------	------------

ÍNDICE	659
---------------	------------

PREFACIO



De acuerdo con Alan Greenspan, Presidente de la Reserva Federal de los Estados Unidos de América, la información tecnológica ha permitido aumentos sin precedente en la productividad de los negocios. En tanto que Internet tiene la mayoría de los créditos, detrás del escenario de la base de datos la tecnología desempeña un papel clave. Después de todo, Internet es sólo un sistema de comunicación; su valor primordial radica en los datos y en la información que transmite hacia y desde las bases de datos.

Las novedades del punto com pueden ocasionar que los estudiantes se pregunten si el valor de estas tecnologías declinará con el tiempo. Nada puede estar más lejos de la verdad. Lou Gestner, presidente de IBM, estableció hace varios años los verdaderos beneficios de Internet y describió las tecnologías que existirían, sólo después de que éstas hubieran sido aceptadas por la corriente principal: la corporativa de Estados Unidos —por las así llamadas compañías de la “vieja economía”—. En la aplicación actual de esta tecnología de bases de datos subyacen importantes oportunidades para cada tipo de negocios y actividades empresariales (así como para los futuros profesionales de bases de datos).

Lo anterior significa que nunca ha existido un momento mejor para estudiar el procesamiento de bases de datos. De las bases de datos de los escritorios personales, a las grandes bases de datos interorganizacionales distribuidas en las computadoras de la red mundial, las bases de datos están aumentando en forma importante los activos de las empresas. La tecnología de las bases de datos se utiliza en mercadotecnia, ventas, producción, operaciones, finanzas, cuentas bancarias, administración y, de hecho, en todas las disciplinas empresariales, con el fin de aumentar la productividad en sus respectivas actividades.

Además, después del frenesí de años recientes con respecto a las nuevas tecnologías y sus productos, los elementos principales de la administración moderna de las bases de datos ahora se han vuelto claras. Continúa siendo esencial el conocimiento conceptual de la modelación y diseño de bases de datos; así mismo, el modelo relacional y el SQL son tan importantes como en el pasado. La administración de bases de datos, especialmente la tecnología que apoya a los administradores de bases de datos multiusuarios, ha adquirido mayor importancia debido a que todas las que usan la nueva tecnología son multiusuario.

Además, la tecnología para las bases de datos publicadas en la Web, especialmente las arquitecturas tritier y multitier, XML, páginas de servidor activo (ASP) y páginas del servidor Java (JSP), han surgido como vencedoras entre muchos contendientes para la publicación de la base de datos. De acuerdo con estas tecnologías, tanto ODBC, con OLE DB, como JDBC continúan siendo importantes.

En resumen, la tecnología de bases de datos es más importante que nunca y es necesario enseñarla de una manera mucho más clara que durante los cinco años anteriores.

CARACTERÍSTICAS DE LA PRESENTE EDICIÓN

De acuerdo con lo que ya comentamos, la segunda parte de este texto ha sido reescrita por completo. Casi todo el contenido de los capítulos 11 al 16 es nuevo. Las principales tareas de la administración de bases de datos se estudian en el capítulo 11, en el 12 se ilustran con Oracle, y en el capítulo 13 de nuevo con el servidor SQL. En el capítulo 14 se estudian las tecnologías básicas para la publicación de bases de datos en la Web, y dichas tecnologías se ilustran para ODBC, OLE DB, IIS, y ASP en el capítulo 15, y de

nueva cuenta con JDBC, JSP, y MySQL en el capítulo 16. El capítulo 17 incluye información sobre OLAP, mientras que el 18 introduce construcciones con el nuevo objeto relacional de Oracle.

Dirigir todos estos temas a una sola meta es un reto y creo que debemos considerar muy seriamente que es necesario dedicarse todo un año al estudio de las bases de datos. Mientras tanto, si quiere estudiar sólo un tema y no tiene tiempo, esta edición ha sido escrita para que usted elija entre tres series de tecnologías alternativas.

Específicamente, considerando el modelado de datos, el texto se refiere al modelo entidad-relación y al modelo de objeto semántico. Si el tiempo es breve, quizás usted quiera abarcar sólo el modelo E-R, porque es el más popular. De manera similar, considerando las bases de datos de usuarios múltiples, elija Oracle en el capítulo 12 o el Servidor SQL en el capítulo 13, dependiendo de las necesidades de los profesionales en su comunidad. Finalmente, considere la publicidad en la Web y si el tiempo es breve, elija IIS, ASP y ODBC en el capítulo 15; o Java, JDBC y JSP en el 16. No se perderá la continuidad si selecciona sólo uno de estos. Por supuesto, si no tiene restricciones de tiempo, todos los temas son importantes.

Concepto	Alternativa 1	Alternativa 2
Modelación de datos	Modelo entidad-relación capítulos 3 y 6	Modelo de objeto semántico capítulos 4 y 7
DBMS ¹ multiusuario	Oracle capítulo 13	SQL Server capítulo 14
Publicidad en la Web	IIS, ASP, ODBC capítulo 15	Java, JDBC, JSP capítulo 16

Esta edición incluye también nuevas series de ejercicios al final del capítulo, los cuales se aplican a una compañía pequeña que produce, comercializa y promociona una línea de estufas de campamento. El objetivo de estos ejercicios es capacitar a los estudiantes para que apliquen los conocimientos que obtengan de cada capítulo en una pequeña, real, aunque limitada aplicación.

PANORAMA CAPÍTULO POR CAPÍTULO

Este texto consta de siete partes. La primera introduce al procesamiento de bases de datos. El capítulo 1 ilustra aplicaciones simples, define términos básicos y bosqueja la historia del procesamiento de bases de datos. El capítulo 2 ilustra el desarrollo de una base de datos simple y una aplicación usando Microsoft Access XP.

La segunda parte versa sobre el modelado de datos. En el capítulo 3 se analiza el modelo entidad-relación y muestra cómo se ha integrado a UML, o al lenguaje de modelación unificado. El capítulo 4 presenta el modelo de objeto semántico, una alternativa de modelación de datos para el modelo E-R. El tema de la tercera parte es el diseño de la base de datos. En el capítulo 5 se analizan el modelo relacional y la normalización. El capítulo 6 aplica las ideas de los capítulos 3 y 5 para transformar los modelos entidad-relación en diseños de bases de datos relacionales. En el capítulo 7 se trabajan las ideas de los capítulos 4 y 5 para transformar los modelos de los objetos semánticos en diseños de bases de datos relacionales.

La cuarta parte aborda los fundamentos de la implementación de la base de datos relacional. El capítulo 8 presenta un panorama, el 9 estudia un proceso SQL y el 10 describe el diseño de las aplicaciones de la base de datos relacional. En la quinta parte se

¹ Database management system, DBMS por sus siglas en inglés.

aborda la administración de una base de datos multiusuarios. En el capítulo 11 se describe la administración de la base de datos y se analizan importantes aspectos de un proceso de base de datos multiusuarios, incluyendo control de concurrencia, seguridad, respaldo y recuperación. Las ideas que se presentan en el capítulo 11 se ilustran con Oracle en el capítulo 12. En el capítulo 12 también se ilustra SQL para la definición de datos. De igual manera, el 13 refleja el análisis del capítulo 11 para ilustrar la administración de bases de datos multiusuarios usando SQL Server.

La publicación de bases de datos en la Web se analiza en la sexta parte. En el capítulo 14 se establecen los fundamentos del procesamiento de la red, las arquitecturas multicapa y XML. En el capítulo 15 se aplican estos conceptos utilizando la tecnología Microsoft, incluyendo ODBC, OLE DB, IIS y ASP. En el capítulo 16 se aplican los conceptos del capítulo 14 utilizando Java; esto incluye JDBC, JSP y MySQL. Los conceptos se ilustran con ejemplos utilizando Linux y Apache Tomcat. Después, en el capítulo 17 se abordan los usos de la administración de datos y se analiza OLAP.

La séptima parte sólo contiene un capítulo en el cual se analiza el procesamiento de bases de datos orientadas a objetos. Lo nuevo en este capítulo es un análisis del objeto relacional de Oracle, sus características y funciones. El apéndice A contiene un breve comentario con respecto a las estructuras de datos, y el apéndice B ilustra el uso del diseñador de tablas, un producto que se puede utilizar para desarrollar los modelos de objeto semántico y convertirlos en diseños de bases de datos y páginas ASP.

RECONOCIMIENTOS

El cambio drástico en la segunda mitad de esta edición se debe a conversaciones muy constructivas y útiles con mi editor, Bob Horan, así como a las revisiones excepcionalmente provechosas e ilustrativas de las siguientes personas:

Jack Becker, Universidad del Norte de Texas
 William D. Burg, Universidad de Alabama en Birmingham
 Bhushan Kapoor, Universidad Estatal de California-Fullerton
 Donald. R. Moscato, Colegio Iona
 Nancy L. Russo, Universidad del Norte de Illinois
 Behrooz Saghaei, Universidad Estatal de Chicago
 Joseph L. Sessum, Universidad Estatal Kennesaw
 Ashraf Shirani, Universidad estatal de San José
 Ludwig Slusky, Universidad estatal de California-Los Ángeles

Además, a Marty Murray del Colegio de la Comunidad de Portland quien me alentó y ayudó a desarrollar el material de Oracle en el capítulo 12. Estoy en deuda con Warner Schyerer de la Corporación SoundDev, en Seattle, por la idea de incluir JSP y MySQL en el capítulo 16. También gracias a Jude Stoller, de SoundDev, quien revisó el material. Agradezco especialmente a Chris Wilkens de Safaridog.com, quien me condujo a través del proceso de instalación de Linux, Apache y Tomcat.

Gracias a Microsoft por comercializar la versión para estudiantes de su producto. También gracias a Thorsten Ganz, presidente de Coolstrategy.com, quien hizo disponible el diseñador de tablas para estudiantes utilizando este texto y a Kenji Kawai, el cual continúa desempeñando un papel fundamental en el desarrollo de los productos basados en el modelo de objeto semántico. Gracias especiales a Kyle Hannon de Prentice-Hall quien dirigió el texto y los componentes de apoyo y se aseguró que funcionarían bien; también agradezco a Vanessa Nuttry, de Prentice-Hall, la cual hábilmente dirigió el proceso de producción. Por último, expreso mi agradecimiento a Lynda por su amor, apoyo y aliento.

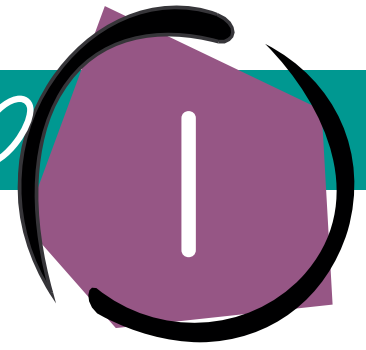
El procesamiento de bases de datos es un tema fascinante que se ha desarrollado y ha evolucionado durante más de treinta años, y creo que continuará cuando menos en los siguientes años. ¡A usted lector, le deseo un gran éxito en su exploración de muchas oportunidades futuras!

David Kroenke
 Seattle, Washington

INTRODUCCIÓN

Los dos capítulos que conforman la parte I introducen el tema del procesamiento de la base de datos. El capítulo 1 describe cuatro aplicaciones fundamentales de ésta y analiza sus ventajas con respecto al antiguo procesamiento de sistemas de archivos. También se define el término *base de datos* y se estudia la historia de su procesamiento. En el capítulo 2 se describen los elementos de una base de datos y se analizan las funciones de un sistema de administración de base de datos (DBMS, por sus siglas en inglés). Esta introducción concluye en el capítulo 2 con el resumen de las tareas que son necesarias para desarrollar una base de datos y sus aplicaciones.

La parte I proporciona un panorama con respecto a las necesidades de las bases de datos, la naturaleza de sus componentes y sus aplicaciones. El propósito es establecer el fundamento para estudiar los detalles relativos a los conceptos y la tecnología de la base de datos en los capítulos siguientes.



Introducción al procesamiento de bases de datos

El procesamiento de bases de datos ha sido siempre un tema importante en el estudio de los sistemas de información. Sin embargo, en los últimos años la expansión de Internet y el drástico desarrollo de la nueva tecnología para Internet ha hecho del conocimiento de la tecnología de bases de datos una de las carreras más apasionantes. La tecnología de bases de datos permite que las aplicaciones de Internet den un paso más allá del simple folleto publicitario que caracterizó a las primeras aplicaciones. Al mismo tiempo, proporciona un medio estandarizado y de rápido acceso para dar a conocer a los usuarios el contenido de bases de datos. Ninguno de estos nuevos desarrollos ignora la necesidad de bases de datos de aplicaciones básicas que fueron vitales para los intereses de los negocios antes de que surgiera Internet. Simplemente aumentan la importancia que tiene el conocimiento de bases de datos.

Para muchos estudiantes este tema es agradable e interesante, aunque lo considere un gran reto. El diseño y desarrollo de bases de datos implica arte e ingeniería. La comprensión de los requerimientos del usuario y su traducción en diseños eficaces de bases de datos es un proceso artístico. Transformar esos diseños en bases de datos físicas con plena funcionalidad, y hacer aplicaciones de alto rendimiento es un proceso de ingeniería. Ambos aspectos están llenos de retos y son acertijos intelectuales divertidos.

Debido a la inmensa necesidad tecnológica de bases de datos, a las habilidades que usted desarrollará y al conocimiento que obtendrá en este curso, le auguramos que tendrá muchas oportunidades de trabajo. El objetivo de este texto es proporcionarle fundamentos sólidos sobre la tecnología fundamental de bases de datos para que pueda iniciar una afortunada carrera en este campo, si así lo decide.

► CUATRO EJEMPLOS DE BASES DE DATOS

El propósito de una base de datos es ayudar a las personas a dar seguimiento a las cosas. Las aplicaciones clásicas de bases de datos se refieren al seguimiento de artículos tales como órdenes, clientes, empleos, empleados, llamadas telefónicas, u otros aspectos de interés para una persona de negocios. Recientemente se ha aplicado la tecnología de

bases de datos en Internet, no sólo en el caso de aplicaciones clásicas, sino también en el de nuevas aplicaciones como anuncios dirigidos a las características del cliente y al seguimiento de las observaciones de éste y de sus hábitos de compra en las páginas Web. Estas bases de datos incluyen datos fotográficos, de audio y de vídeo, así como información tradicional como nombres, antecedentes y números telefónicos. Los cuatro ejemplos que veremos a continuación ilustran el uso de la tecnología de las bases de datos a través de este amplio rango de aplicaciones.

PINTORA DE CASAS MARY RICHARDS

Mary Richards es una pintora profesional de casas que es la propietaria y operadora de una pequeña compañía, integrada por ella, otro pintor profesional y, cuando es necesario, pintores de medio tiempo. Mary ha estado en el negocio durante diez años y se ha ganado la reputación de ser profesional y ofrecer precios razonables. Ella consigue la mayoría de los trabajos con clientes que la contratan en varias ocasiones y también a través de las referencias y recomendaciones que dan sus clientes a otras personas; además consigue algo de trabajo con los constructores y diseñadores profesionales de interiores.

Los clientes recuerdan mejor a Mary de lo que ella los recuerda a ellos. De hecho, algunas veces se avergüenza cuando un cliente le llama y le dice algo como: “Hola Mary, habla John Maples. Pintaste mi casa hace tres años.” Ella sabe que debería recordar a todo aquel que le llama por el trabajo que realizó, pero pinta más de 50 casas al año y es difícil que recuerde algunos clientes en particular. Esta situación empeora cuando ellos le dicen: “A mi vecino le gustó el trabajo que realizaste en nuestra casa y le gustaría algo similar para la suya.”

Con el fin de ayudar a su memoria y dar un mejor seguimiento a sus registros, Mary tenía un especialista que desarrolló una base de datos y la aplicación que utiliza en su computadora personal. La base de datos almacena registros de los clientes, trabajos y fuentes de referencias en forma de tablas, como se muestra en el ejemplo de la figura 1-1.

Éste es el trabajo de un programa llamado sistema administrador de base de datos (DBMS) que almacena y recupera datos de estas tablas. Por desgracia, cuando estos da-

► FIGURA 1-1

Tablas de datos para la pintora de casas Mary Richards

The screenshot shows the Microsoft Access interface with three tables displayed in a stacked view:

SOURCE : Table

SOURCE_ID	Name	PhoneNumber
1	Valley Designs	(303) 549-8879
2	Aspen Construction	(303) 776-8899
3	Mary Engers Design	(303) 767-7703

CUSTOMER : Table

CUSTOMER_ID	CustomerName	Street	City	State	Zip	PhoneNumber	SOURCE_ID
2	Wu, Jason	123 E Elm	Denver	CO	80210-7786	(303) 555-0089	2
3	Maples, Marilyn	2518 S. Link Lane	Denver	CO	80243-	(303) 777-8898	3
4	Jackson, Chris	4700 Lafayette	Boulder	CO	81237-3484	(549) 388-1243	2

JOB : Table

JOB_ID	JobDate	Description	AmountBilled	AmountPaid	CUSTOMER_ID
1	3/3/2000	Paint exterior in 794 White	\$2,750.00	\$2,750.00	2
2	7/7/2000	Paint dining room and kitchen	\$1,778.00	\$1,778.00	2
3	3/15/2001	Prep and paint upstairs bath	\$550.00	\$550.00	2
4	4/3/2001	Paint exterior doors in 633 Red	\$885.00	\$885.00	4
5	7/14/2001	Prep and paint interior wood trim	\$1,299.00	\$1,299.00	3
			\$0.00	\$0.00	0

► FIGURA 1-2

Muestra de la forma de entrada de datos de la pintora de casas Mary Richards

JobDate	Description	AmountBilled	AmountPaid
3/3/2000	Paint exterior in 794 White	\$2,750	\$2,750
7/7/2000	Paint dining room and kitchen	\$1,778	\$1,778
3/15/2001	Prep and paint upstairs bath	\$550	\$550
*		\$0	\$0

tos están en forma de tablas no son muy útiles para Mary, porque lo que ella necesita es saber cómo se relacionan entre sí los clientes, trabajos y referencias; por ejemplo, qué trabajos se han hecho para determinado cliente, o quién ha llegado gracias a una recomendación de alguien en particular.

Para lograr esto, el asesor de Mary creó una aplicación de base de datos que procesa formas de entrada de datos y produce reportes. Considere como ejemplo la forma de la figura 1-2. Aquí, Mary teclea los datos de sus clientes, tales como nombre, teléfono y dirección. También los relaciona con una fuente de referencia en particular y teclea los datos de los trabajos que realizó para ellos. Dichos datos se pueden mostrar entonces en forma de reportes, como el que se muestra en la figura 1-3. Otros usos de la base de datos incluyen el registro de las cotizaciones, el seguimiento de las fuentes de referencia y la producción de etiquetas para enviar por correo la publicidad de ventas directas.

► FIGURA 1-3

Muestra de un reporte para la pintora de casas Mary Richards

CustomerName	PhoneNumber	JobDate	Description	AmountBilled	AmountPaid
Wu, Jason	(303) 555-0089	3/3/2000	Paint exterior in 794 White	\$2,750	\$2,750
		7/7/2000	Paint dining room and kitchen	\$1,778	\$1,778
		3/15/2001	Prep and paint upstairs bath	\$550	\$550
		Total		\$5,078	\$5,078
Maples, Marilyn	(303) 777-689	7/14/2001	Prep and paint interior wood trim	\$1,299	\$1,299
		Total		\$1,299	\$1,299
Jackson, Chris	(549) 388-124	4/3/2001	Paint exterior doors in 633 Red	\$285	\$285
		Total		\$285	\$285
Grand Total				7262	7262

La aplicación de la base de datos y el DBMS procesan la forma y almacenan los datos que se introducen en tablas como las de la figura 1-1. De igual manera, la aplicación y el DBMS extraen los datos de dichas tablas para crear un reporte similar al de la figura 1-3.

Considere nuevamente los datos de la figura 1-1 y observe que los renglones en las tablas son referencias cruzadas y están vinculados entre sí. Cada JOB (TRABAJO) contiene CUSTOMER_ID (identificación del cliente) (CLIENTE_ID), el CLIENTE que pagó por ese TRABAJO y cada CLIENTE lleva SOURCE_ID (la identificación de la fuente) (FUENTE_ID), o la persona que lo recomendó. Estas referencias se usan para combinar datos y producir formas y reportes como los que se muestran en las figuras 1-2 y 1-3.

Como se podrá imaginar, es poco probable que Mary sepa cómo diseñar las tablas de la figura 1-1, cómo utilizar un DBMS para crear esas tablas, y cómo desarrollar la aplicación para elaborar formas y reportes. Cuando termine este curso, usted sabrá cómo usar la tecnología de bases de datos para crear una, así como sus aplicaciones. También será capaz de diseñar y manipular tablas para crear formatos y reportes de mayor complejidad.

CASA DE MÚSICA TREBLE CLEF

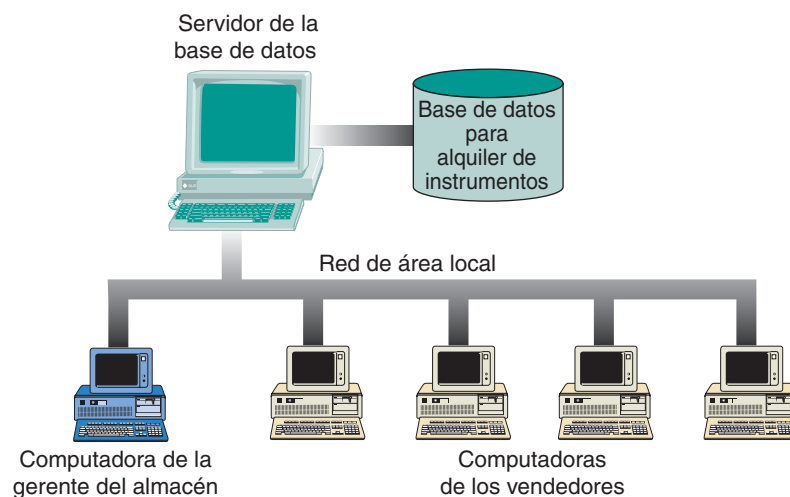
La base de datos de Mary Richards se llama base de datos de un *solo usuario*, porque sólo uno tiene acceso a ésta en un momento determinado. En algunos casos, esta limitación es demasiado restrictiva; mucha gente necesita tener acceso a la base de datos simultáneamente desde múltiples computadoras. Estas bases de datos *multiusuarios* son más complicadas porque el DBMS y la aplicación deben sostener el trabajo del usuario sin la interferencia de los otros usuarios.

La casa de música Treble Clef usa una aplicación de base de datos para mantener el seguimiento de los instrumentos musicales que renta. Necesita una aplicación multiusuarios porque, durante los periodos de trabajo, varios vendedores pueden rentar instrumentos musicales al mismo tiempo. También, la gerente del almacén necesita tener acceso a la base de datos de renta para determinar cuándo ordenar más instrumentos de determinado tipo, sin que se interrumpa el proceso de alquiler al hacer esto.

El almacén de la casa Treble Clef tiene una red local que conecta a varias computadoras personales con un servidor, el cual mantiene la base de datos de alquiler, como se muestra en la figura 1-4. Cada empleado tiene acceso a la aplicación de la base de datos mediante las tres formas ilustradas en la figura 1-5. La Customer form (forma que el cliente llena) se usa para mantener los datos de éste, la Rental Agreement form (forma

► FIGURA 1-4

Red de área local con el servidor de la base de datos que usa la casa de música Treble Clef



► FIGURA 1-5

Tres formas que usa la casa de música Treble Clef:
 (a) Forma para el cliente,
 (b) Forma del convenio de renta y
 (c) Forma del instrumento

CUSTOMER

Treble Clef Music – Customer Form

CustomerName: Mary & Fred Jackson
 HomePhone: (703) 443-7788
 WorkPhone: (703) 443-4482
 Street: 1200 Seventeenth Ave
 City: Alexandria
 State: VA Zip: 02234-5567

Children: Katherine, Jaymalina, *

Record: 1 of 1

INVOICES

InvoiceNumber	InvoiceDate	Total
100087	10/16/2001	\$45
98884	10/16/2000	\$37
*	0	\$0

Record: 1 of 2

(a)

Rental Agreement

Treble Clef Music – Rental Agreement Form

InvoiceNumber: 100087
 InvoiceDate: 10/16/2001

Customer: Mary & Fred Jackson
 WorkPhone: (703) 443-4482
 HomePhone: (703) 443-7788

Rental Items

SerialNumber	Category	DateOut	DateReturned	MonthlyFee
478990	B flat clarinet	10/16/2001		\$17.50
556788	Standard violin	10/16/2001		\$27.25
478990	B flat clarinet			
556788	Standard violin			
556790	Premium violin			

Total: \$44.75

Record: 1 of 2

(b)

INSTRUMENT

Treble Clef Music – Instrument Data Form

SerialNumber: 478990
 Category: B flat clarinet
 MonthlyFee: \$18
 Rented?: No

INVOICES

InvoiceNumber	InvoiceDate	Total
100087	10/16/2001	\$44.75
*		

Record: 1 of 1

Record: 1 of 3

(c)

del convenio de renta) se utiliza para darle seguimiento a los instrumentos que han sido rentados, si ya fueron regresados o no, y la Instrument Data form (forma de datos del instrumento) que se utiliza para mostrar sus características y los antecedentes del alquiler.

Para comprender los problemas que se deben enfrentar en una base de datos multiusuarios, considere lo que sucedería cuando dos clientes intentaran rentar el clarinete plano-B al mismo tiempo. El DBMS y los programas de aplicación deben detectar que esta situación está ocurriendo e informar a los empleados que tienen que escoger otro instrumento.

OFICINA ESTATAL DE LICENCIAS Y REGISTRO DE VEHÍCULOS

Ahora consideremos una aplicación aún mayor de la tecnología de base de datos: una oficina estatal de licencias y de registro de vehículos. Tiene 52 centros de pruebas de manejo, expedición de licencias para conductores, renovación de licencias de manejo, y también 37 oficinas que expiden registros de vehículos.

El personal tiene acceso a una base de datos para realizar su trabajo. Antes de que a las personas se les otorgue o renueve su licencia de conducir, hay que verificar sus registros en la base de datos para buscar posibles infracciones de tránsito, accidentes o arrestos. Estos datos se utilizan para determinar si la licencia debe o no ser renovada, o si se debe otorgar con ciertas limitaciones. De igual manera, el personal del departamento de registro de automóviles tiene acceso a la base de datos para determinar si un auto ha sido registrado antes y, si es así, quién lo registró, o si existe algún asunto importante que impida expedir el registro.

Esta base de datos tiene cientos de usuarios, incluyendo no sólo al personal de las licencias y registros, sino al del departamento estatal de contribuciones y del departamento jurídico. No es de extrañar que la base de datos sea grande y compleja, con más de 40 diferentes tablas de datos, muchas de las cuales contienen cientos de miles de renglones.

Las bases de datos de las grandes organizaciones, como la oficina de licencias y registros, fueron las primeras aplicaciones de este tipo de tecnología. Estos sistemas han existido durante 20 o 30 años y se han modificado para satisfacer los cambios que ocurrieron durante ese periodo. Otros ejemplos de bases de datos organizacionales se relacionan con el procesamiento de cuentas en bancos e instituciones financieras, sistemas de producción y de suministro de material en fábricas grandes, procesamiento de registros médicos en hospitales, y en compañías de seguros y agencias gubernamentales.

Actualmente muchas organizaciones están adaptando sus aplicaciones de bases de datos para permitir a los clientes tener acceso, e incluso cambiar sus datos, por medio de Internet. Si usted llegara a trabajar en una gran organización importante, probablemente le podrían asignar ese proyecto.

CENTRO DE RESERVACIONES DE LA ISLA CALVERT

La isla Calvert es muy poco conocida; es una isla hermosa en la costa oeste de Canadá. Para promover el turismo en un mercado internacional, la Cámara de Comercio de la isla Calvert ha desarrollado un sitio Web que tiene tres finalidades:

- Promover la belleza y las oportunidades recreativas que brinda la isla
- Obtener y almacenar datos de nombres y direcciones de los visitantes del sitio Web para darles seguimiento con el envío de promociones por correo
- Obtener y almacenar solicitudes de reservación para hoteles, cabañas y servicios turísticos y comunicárselo a los vendedores

Se utilizan dos bases de datos para apoyar a este sitio Web. La primera es una base de datos promocional que almacena datos, fotos, vídeos y algunos pequeños sonidos.

dos del lugar, actividades e instalaciones en la isla Calvert. Esta base de datos tiene dos tipos de usuarios: el normal, que ingresa sólo con fines de lectura y utiliza exploradores estándar. Estos usuarios pueden posicionarse y hacer “clic” alrededor del sitio Web para ver las actividades e instalaciones que les pueden interesar. Detrás de escena, una aplicación de base de datos está extrayendo datos y elementos multimedia de la base de datos promocional (véase la figura 1-6).

El segundo tipo de usuario de la base de datos promocional es un empleado de la Cámara de Comercio que se encarga del sitio. El empleado agrega, cambia, borra datos y archivos multimedia en la base de datos, como por ejemplo cambios de promociones. Los vendedores entran y salen del programa y dan respuesta a las preguntas de los usuarios.

Además de la base de datos promocional, las aplicaciones en el sitio Web procesan una base de clientes y reservaciones, la cual almacena los datos que ingresan cuando los visitantes del sitio Web llenan un formato de encuesta y solicitan una reservación. Los datos acerca del nombre del cliente, la dirección postal y de correo electrónico, intereses, preferencias y las solicitudes de reservación se conservan en la base de datos.

► FIGURA 1-6

*Página Web
del Centro de
reservaciones
de la isla Calvert*

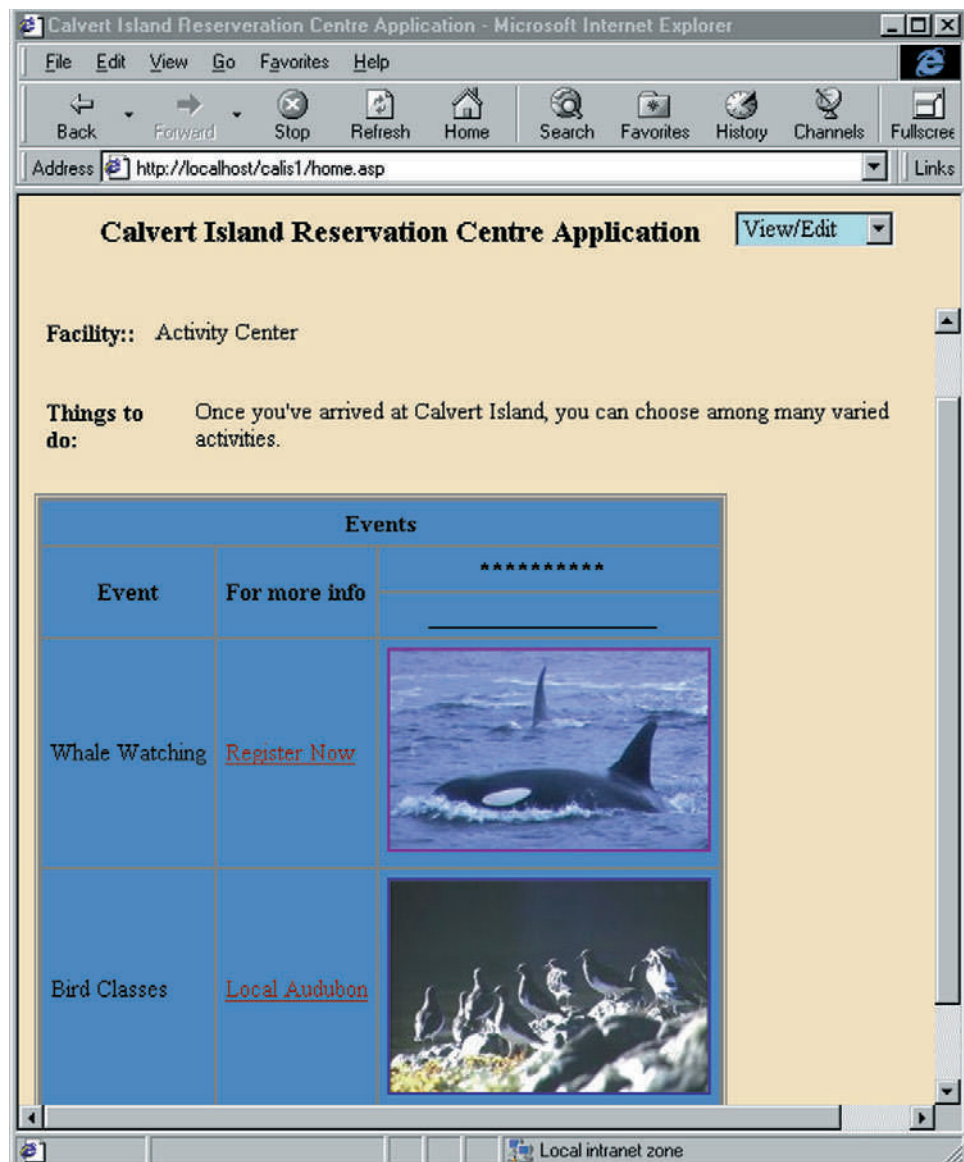


 FIGURA 1-7

Características de las aplicaciones de tecnología de Internet

- Incluye datos estructurados y datos multimedia
- Las formas y reportes se despliegan a través de un explorador estándar
- Los datos se transfieren a través de estándares de Internet, tales como HTTP, DHTML y XML

Cuando ingresa una solicitud, la aplicación la envía a un vendedor a través del correo electrónico. Periódicamente se prepara un resumen de los reportes de reservación y se le envía por correo electrónico a los vendedores para que den seguimiento a los posibles clientes, pero también sirve para otros propósitos de administración.

Tres características principales de la base de datos de la isla Calvert la distinguen de las aplicaciones anteriores. En primer lugar, una gran parte de la primera base de datos contiene no sólo datos estructurados, tales como los nombres de los vendedores, sino también corrientes de bits no estructuradas de archivos multimedia. Segundo, el contenido de la aplicación se reparte al usuario a través de un explorador estándar. Las formas que se usan para Treble Clef y la oficina de licencias tienen un formato específico que creó el diseñador y sólo cambia cuando se modifica la aplicación. En contraste, los usuarios de la isla Calvert ven la forma en un formato que se determina no sólo mediante las aplicaciones, sino también por la etiqueta, la versión y las opciones locales que usan sus exploradores.

La tercera característica diferente de la aplicación de la isla Calvert es que se usó la tecnología estandarizada orientada a la Web para transferir datos entre el explorador, la aplicación y la base de datos. Se utilizan **protocolo de transferencia de hipertexto (HTTP**, por sus siglas en inglés), **lenguaje dinámico para marcar hipertexto (DHTML**, por sus siglas en inglés), y **el lenguaje de marcado extensible (XML**, por sus siglas en inglés). Usando estos medios estándar cualquier usuario que tenga un explorador puede acceder a esta aplicación. El software no debe estar instalado previamente en sus computadoras. Consecuentemente, el uso de esta aplicación es virtualmente ilimitado. Analizaremos el papel de HTTP, DHTML y XML para las aplicaciones de bases de datos en los capítulos 14 al 16. (Véase la figura 1-7.)

COMPARACIÓN DE APLICACIONES DE BASES DE DATOS

Estos ejemplos representan una muestra con respecto a los usos de la tecnología de bases de datos. Cientos de miles de bases de datos son como la que usa la pintora de casas Mary Richards: de un solo usuario con una cantidad relativamente pequeña de datos, digamos de tan sólo 10 megabytes. Las formas y reportes por lo general son simples y sencillas.

Hay otras bases, como la que usa la casa de música Treble Clef, que tienen más de un usuario, pero casi siempre un poco menos de 20 o 30 usuarios juntos. Contienen una cantidad moderada de datos, digamos 50 o 100 MB. Las formas y reportes son necesariamente bastante más complicados para apoyar diferentes funciones de negocios.

Las bases de datos más grandes, como las del registro de autos, tienen miles de usuarios y trillones de bytes de datos. Están en uso muchas aplicaciones diferentes, y cada una tiene sus propias formas y reportes. Por último, algunas bases de datos implican el uso de la tecnología de Internet y de procesos, de caracteres y datos multimedia tales como fotografías, sonidos, animaciones, películas y similares. En la figura 1-8 se resumen las características de estos tipos de bases de datos.

Cuando usted concluya este libro deberá ser capaz de diseñar y crear bases de datos y aplicaciones como las que se usaron con Mary Richards y la casa Treble Clef. Probablemente no podrá crear una tan grande y compleja como la de la oficina de registro de vehículos, pero podrá desempeñarse como un miembro eficaz de un equipo que diseñe y cree una. También deberá ser capaz de crear una base de datos pequeña o mediana utilizando la tecnología de Internet.

► FIGURA 1-8

Características de los diferentes tipos de bases de datos

Tipo	Ejemplo	Número típico de usuarios concurrentes	Tamaño común de la base de datos
Personal	Pintora de casas Mary Richards	1	<10 Megabytes
Grupo de trabajo	Casa de música Treble Clef	<25	<100 Megabytes
Organizacional	Licencias y registros	Cientos a miles	>1 trillón de bytes
Internet	Reservaciones para la isla Calvert	Cientos a miles	<Cualquiera

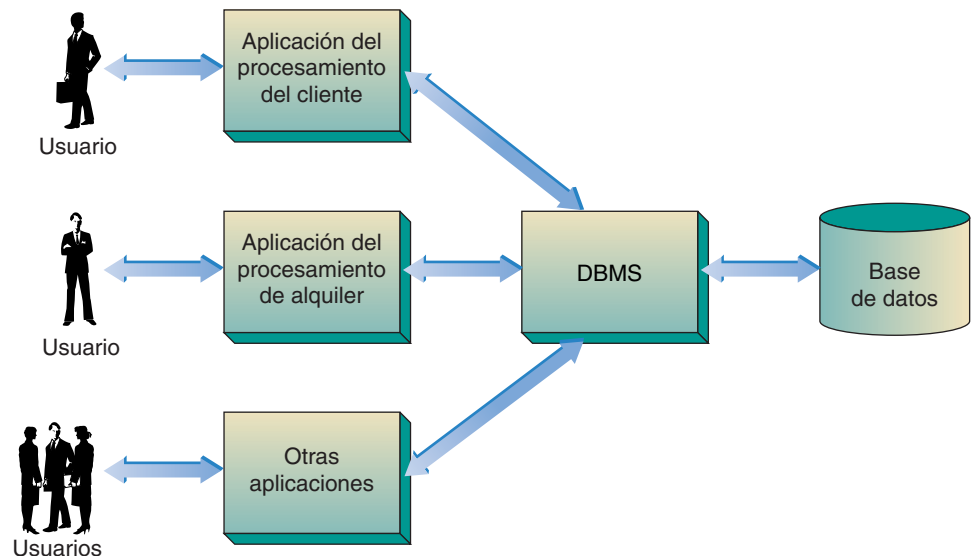
► LA RELACIÓN DE LOS PROGRAMAS DE APLICACIONES Y LOS DBMS

Todos los ejemplos anteriores y, en realidad, todas las aplicaciones de bases de datos tienen la estructura general que se muestra en la figura 1-9: el usuario interactúa con una aplicación que hace interfaz alternadamente con el DBMS, el cual tiene acceso a los datos de la base de datos.

En una época la frontera entre el programa de aplicación y el DBMS estaba claramente definida. Las aplicaciones estaban escritas en lenguajes de tercera generación tales como COBOL, y esas aplicaciones invocaban a los DBMS para la organización de los datos. De hecho, esto aún se hace con mayor frecuencia en las bases de datos de macrocomputadoras.

Sin embargo, actualmente las características y funciones de muchos productos DBMS se han desarrollado tanto que ahora el propio DBMS puede procesar grandes partes de la aplicación. Por ejemplo, la mayoría de los productos DBMS contiene escritores de reportes y generadores de formas que se pueden integrar en una aplicación. Este hecho es importante para nosotros por dos razones. Primera, aunque la mayor parte de es-

► FIGURA 1-9

Relaciones de usuarios, aplicaciones de bases de datos, DBMS, y base de datos

te texto considera el diseño y la formación de bases de datos, con frecuencia nos referiremos al diseño y desarrollo de la aplicación de la base de datos. Después de todo, ningún usuario quiere sólo una base de datos. En realidad, los usuarios quieren formas, reportes y consultas que estén basadas en sus datos.

Segunda, de vez en cuando observará una sobreposición entre el material analizado en esta clase y el que estudió en su clase de desarrollo de sistemas, debido a que desarrollar aplicaciones de bases de datos eficaces requiere muchas de las habilidades que ha aprendido o aprenderá en su clase de desarrollo de sistemas. Asimismo, actualmente la mayoría de las clases de desarrollo de sistemas comprenden el diseño de bases de datos. La diferencia entre los dos cursos es el énfasis. Aquí, nuestro enfoque se centra en el diseño, la construcción y el procesamiento de bases de datos. En una clase de sistemas, el interés principal radica en el desarrollo de los sistemas de información, muchos de los cuales usan tecnología de bases de datos.

► SISTEMAS DE PROCESAMIENTO DE ARCHIVOS

La mejor forma para entender la naturaleza general y las características de las bases de datos actuales es ver las características de los sistemas que precedieron al uso de la tecnología de bases de datos. Estos sistemas revelan los problemas que ha resuelto dicha tecnología.

Los primeros sistemas de información de negocios almacenaban grupos de registros en archivos por separado y eran llamados sistemas de procesamiento de archivos. Por ejemplo, la figura 1-10 representa dos sistemas de procesamiento de archivos que podría usar Treble Clef. Un sistema procesa los datos de CUSTOMER y otro los datos del RENTAL (ALQUILER).

Aunque los sistemas de procesamiento de archivos han mejorado mucho con respecto a los sistemas manuales de registro, tienen importantes limitaciones:

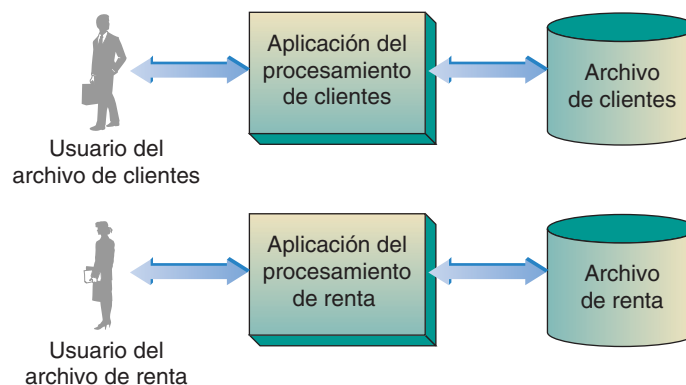
- Los datos están separados y aislados
- La mayoría de los datos están duplicados
- Los programas de aplicación dependen de los formatos de los archivos
- Con frecuencia los archivos son incompatibles entre sí
- Es difícil representar los datos de acuerdo con las perspectivas de los usuarios

DATOS SEPARADOS Y AISLADOS

Los vendedores de Treble Clef necesitan relacionar a sus clientes con los instrumentos que alquilan. Para el sistema de la figura 1-10, los datos necesitan extraerse de algún modo de los archivos CUSTOMER y RENTAL y combinarse en un tercer archivo. Con el procesamiento de archivos esto es difícil de realizar. Primero, los analistas de sistemas y los programadores deben determinar qué partes de cada archivo son necesarias; entonces, tienen que decidir cómo se relacionan entre sí los archivos, y por último, deben coordinar el pro-

► FIGURA 1-10

Sistemas de procesamiento de archivos



cesamiento de éstos de tal modo que se extraigan los datos correctos. La coordinación de dos archivos es bastante difícil, pero ¡imagínese la tarea de coordinar 10 o más de ellos!

DUPLICACIÓN DE DATOS

En el ejemplo de Treble Clef se puede almacenar muchas veces el nombre de un cliente, su dirección y otros datos. Esto es, los datos de CUSTOMER son almacenados una vez y otra más por cada contrato de RENTAL que tenga el cliente. Si bien con esta duplicación de datos se desperdicia espacio de archivo, este no es el problema más serio, sino que esa duplicación afecta la **integridad de datos**.

Un conjunto de datos tiene integridad si éstos son lógicamente consistentes. Con frecuencia se tienen datos de integridad pobre cuando están duplicados. Por ejemplo, si un cliente cambia su nombre o dirección, entonces deben actualizarse todos los archivos que contienen los datos; pero lo peligroso es que todos los archivos *no* puedan actualizarse y causen discrepancias entre sí. En el ejemplo de Treble Clef el cliente no puede tener una dirección para un registro de RENTAL y una dirección diferente para un segundo registro de RENTAL.

Los problemas de integridad de datos son serios. Si los elementos de datos difieren, se producirán resultados inconsistentes e incertidumbre. Si un reporte de una aplicación es diferente al de otra, ¿quién decidirá cuál es el correcto? Cuando los resultados son inconsistentes se duda de la credibilidad de los datos almacenados, e incluso de la propia función MIS.

DEPENDENCIA DEL PROGRAMA DE APLICACIÓN

Con el procesamiento de archivos los programas de aplicación dependen de los formatos de los archivos. Usualmente, en sistemas de procesamiento de archivos las formas físicas de archivos y registros son parte del código de aplicación. En COBOL, por ejemplo, los formatos de los archivos están escritos en DATA DIVISION. El problema es que cuando se hacen cambios en los formatos de archivos, también se deben modificar los programas de aplicaciones.

Por ejemplo, si se modifica el registro del cliente para extender el campo de zip code (código postal) de cinco a nueve dígitos, se deben modificar todos los programas que emplean el registro del cliente, aun cuando no se utilice el campo de código postal. Lo anterior se debe a que ahí puede haber 20 programas que procesen el archivo CUSTOMER; esto significa que cuando hay cambio el programador tiene que identificar todos los programas afectados, modificarlos y después probarlos nuevamente; estas tareas consumen tiempo y tienden a producir errores. También, los programadores requieren modificar los programas que no usen el campo cuyo formato ha cambiado, lo cual es una pérdida de dinero.

ARCHIVOS INCOMPATIBLES

Una de las consecuencias que tiene la dependencia de datos del programa es que los formatos de los archivos dependen del lenguaje o del producto que se usó para generarlo. De esta forma, el formato de un archivo procesado con un programa de COBOL es diferente al de uno procesado con un programa C++.

Como resultado, los archivos no se pueden combinar o comparar rápidamente. Se supone, por ejemplo, que el archivo FILE-A contiene los datos de CUSTOMER, que incluyen el CustomerNumber (Número de Cliente), y el archivo FILE-B contiene los datos de RENTAL, que también incluyen Número de Cliente. Suponga que una aplicación requiere que combinemos registros donde coinciden los números de clientes. Si el FILE-A fuera procesado mediante un programa de Visual Basic, y el FILE-B con un programa C++, necesitaríamos convertir ambos archivos a una estructura común antes de poder combinar los registros. Esto tomaría tiempo y a veces sería difícil. Los problemas empeoran conforme aumenta el número de archivos que hay que combinar.

LA DIFICULTAD DE REPRESENTAR LOS DATOS DESDE LA PERSPECTIVA DE LOS USUARIOS

Es difícil representar los datos del procesamiento de archivos de una manera que parezca natural a los usuarios. Ellos quieren ver los datos de RENTAL en un formato como el de la figura 1-5(b). Pero para mostrarlos de esta forma se necesitan extraer, combinar y presentar juntos varios archivos diferentes. Esta dificultad surge porque con el procesamiento de archivos las relaciones entre los registros no se representan o procesan rápidamente. Debido a que un sistema de procesamiento de archivos no puede determinar de inmediato cuál CUSTOMER ha alquilado qué instrumento, es difícil producir una forma que muestre las preferencias del cliente.

► SISTEMAS DE PROCESAMIENTO DE BASE DE DATOS

La tecnología de bases de datos se desarrolló para superar las limitaciones de los sistemas de procesamiento de archivos. Para entender cómo, compare el sistema de procesamiento de archivos de la figura 1-10 con el sistema de bases de datos de la figura 1-9. Los programas de procesamiento de archivos acceden directamente a los archivos de datos almacenados. En contraste, los programas de procesamiento de la base de datos invocan al DBMS para tener acceso a los datos almacenados. Esta diferencia es significativa porque facilita la programación de la aplicación; es decir, los programadores de aplicaciones no tienen que preocuparse por cómo se almacenan físicamente los datos, y quedan en libertad de concentrarse en asuntos que sean importantes para el usuario, en lugar de distraerse con aquellos que competen al sistema de computación.

DATOS INTEGRADOS

En un sistema de base de datos todos los datos de aplicación están almacenados en un medio simple llamado **base de datos**. Un programa de aplicación puede ordenarle al DBMS que acceda a los datos del cliente, a los de las ventas, o a ambos. Si los dos son necesarios, el programador de la aplicación sólo especifica cómo se combinan y el DBMS realiza las operaciones necesarias. Así, el programador no es responsable de escribir los programas para consolidar los archivos, como se debe hacer en el sistema de la figura 1-10.

REDUCCIÓN DE DATOS DUPLICADOS

Con el procesamiento de bases de datos la duplicación es mínima. Por ejemplo, en la base de datos de Treble Clef el número de cliente, nombre y dirección se almacenan una sola vez. Siempre que el DBMS necesita estos datos puede recuperarlos y después modificarlos, sólo requiere autorización. Debido a que los datos se almacenan en un solo lugar, los problemas de integridad son menos comunes y hay menos oportunidad de que existan discrepancias entre las múltiples copias de los mismos elementos de los datos.

INDEPENDENCIA DE DATOS/PROGRAMAS

El procesamiento de bases de datos reduce la dependencia de los formatos del archivo. Todos los formatos de registro se almacenan en la misma base (junto con los datos) y el DBMS tiene acceso a ellos, no mediante los de aplicación. A diferencia de los programas de procesamiento de archivos, los de aplicación de base de datos no necesitan incluir el formato de todos los registros y los archivos que procesan. En cambio, sólo los programas de aplicación deben contener una definición (la longitud y el tipo de datos) de cada uno de los datos que se necesitan de la base. El DBMS localiza los datos en los registros y maneja otras transformaciones similares.

La independencia de datos/programas minimiza el impacto de los cambios en el formato de los datos en los programas de aplicación. Los cambios en los formatos se introdu-

cen al DBMS, el cual actualiza los datos y mantiene la relación con la estructura de la base de datos. La mayoría de las veces los programas de aplicación no se enteran de que el formato ha cambiado. Esto también significa que cuando se agregan, cambian o borran datos de la base sólo deben modificarse los programas que usan esos datos en particular. En aplicaciones que constan de docenas de programas esto representa un considerable ahorro de tiempo.

REPRESENTACIÓN FÁCIL DE LAS PERSPECTIVAS DEL USUARIO

Como descubrirá a través de este texto, la tecnología de bases de datos hace posible representar en una forma sencilla los objetos que se encuentran en el mundo del usuario. Las formas de la figura 1-5 pueden producirse rápidamente a partir de una base de datos, ya que están almacenadas en ella las relaciones entre los registros de datos.

► DEFINICIÓN DE UNA BASE DE DATOS

El término *base de datos* tiene muchas interpretaciones diferentes. Se ha usado para referirse a un conjunto de tarjetas indexadas a los volúmenes y volúmenes de datos que un gobierno recopila acerca de sus ciudadanos. En este texto, usamos este término con un significado específico: *una base de datos es un conjunto autodescriptivo de registros integrados*. Es importante comprender plenamente esta definición.

UNA BASE DE DATOS ES AUTODESCRIPTIVA

Una base de datos es autodescriptiva: contiene, además de los datos fuente del usuario, una descripción de su propia estructura. Esta descripción se llama **diccionario de datos** (o también **directorio de datos** o **metadatos**).

En este sentido, una base de datos es similar a una biblioteca, la cual integra una colección de libros autodescriptivos. Además de los libros, la biblioteca contiene un catálogo de tarjetas que los reseñan. De la misma manera, el diccionario de datos (que es parte de la base, al igual que el catálogo de tarjetas es parte de la biblioteca) describe lo que contiene la base de datos.

¿Por qué es tan importante la característica autodescriptiva de una base de datos? En primer lugar, porque fomenta la independencia de datos/programas; es decir, hace posible determinar la estructura y el contenido de la base de datos examinándola. No es necesario adivinar qué contiene, ni necesitamos mantener documentación externa del archivo, o de los formatos de registro (como se hace en los sistemas de procesamiento de archivos).

En segundo lugar, si cambiamos la estructura de los datos en la base (por ejemplo, cuando se agregan nuevos datos a un registro existente), sólo se introduce el cambio en el diccionario de datos. Se necesita cambiar muy pocos programas, si es que realmente es muy necesario. En la mayoría de los casos sólo se debe suplir aquellos programas que procesan los datos modificados.

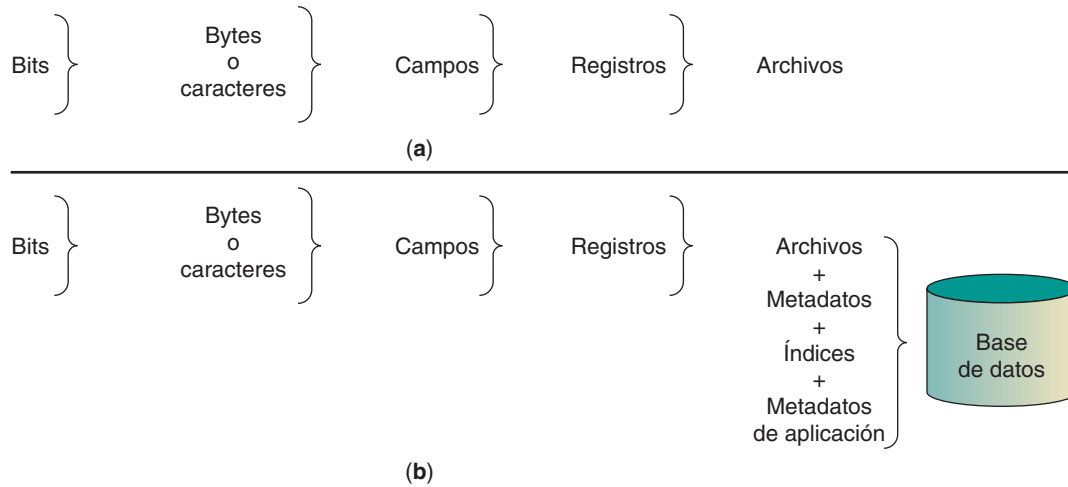
UNA BASE DE DATOS ES UN CONJUNTO DE REGISTROS INTEGRADOS

La jerarquía estándar de los datos es la siguiente: los bits se agrupan en bytes o caracteres; los caracteres se agrupan en campos; los campos integran registros, y los registros se agrupan en archivos (véase la figura 1-11 [a]). Es tentador seguir la tendencia establecida y decir qué archivos conforman las bases de datos. Aunque este enunciado es verdadero, no funciona del todo bien.

Una base de datos incluye archivos de datos del usuario, pero también más. Como mencionamos anteriormente, una base de datos contiene una descripción de sí misma en los metadatos. Además, incluye **índices** que se utilizan para representar las relaciones entre los datos, y también para mejorar el desarrollo de las aplicaciones de la base

► FIGURA 1-11

Jerarquía de los elementos de datos en: (a) sistemas de procesamiento de archivos, y (b) sistemas de base de datos



de datos. Por último, la base con frecuencia contiene datos acerca de las aplicaciones que usa. La estructura de la forma de entrada de datos, o de un reporte, a veces es parte de la base. A esta última categoría de datos se le llama **metadatos de aplicación**. Así, una base contiene los cuatro tipos de datos que se muestran en la figura 1-11(b): archivos de datos del usuario, metadatos, índices y metadatos de aplicación.

UNA BASE DE DATOS ES UN MODELO DE UN MODELO

Una base de datos es un modelo. Es tentador decir que es un modelo de la realidad, o de alguna parte de la realidad en la medida en que se relaciona con un negocio. Sin embargo, esto no es verdad. Una base de datos no es un modelo de la realidad ni de una parte de ésta, sino un modelo del *modelo del usuario*. Por ejemplo, la base de datos de Mary Richards es un modelo de la forma en que ella percibe su negocio. Para ella, su negocio tiene clientes, trabajos y referencias, así que su base de datos contiene representaciones de los hechos que relacionan esas características. Los nombres y las direcciones de los clientes, las fechas y las descripciones de sus trabajos, y los nombres de sus fuentes de referencias son medidas importantes para ella, desde el punto de vista de su negocio.

Las bases de datos varían en su nivel de detalle. Algunas son simples y burdas. Una lista de clientes y de las cantidades que éstos adeudan es una representación aproximada del modelo mental de Mary. Una representación más detallada incluye trabajos, referencias y los viajes que se efectúan para llevar a cabo cada trabajo. Una representación muy detallada contiene la cantidad y el tipo de pintura que se utilizó, cuántas brochas se necesitaron y las horas que requirió cada tarea en particular; por ejemplo, medir, pintar madera, pintar paredes, limpiar, etcétera.

El grado de detalle que debe incorporarse a una base de datos depende de la información que se requiera. Evidentemente, mientras más información se necesita, la base de datos debe ser lo más detallada posible. Decidir la cantidad apropiada de detalles es una parte importante del trabajo para el diseño de una base de datos. Como lo descubrirá, el criterio principal es el nivel de detalle que está en la mente de los usuarios.

La base de datos es un modelo dinámico porque cambia los negocios. Las personas van y vienen, los productos surgen y se descontinúan, el dinero se gana y se gasta. De la misma manera en que ocurren estos cambios, los datos que representan los negocios también deben ser modificados. De lo contrario, los datos se volverán obsoletos y representarán erróneamente al negocio.

Las **transacciones** son representaciones de sucesos. Cuando éstos ocurren, las transacciones respectivas se deben procesar en función de la base de datos. Para realizar lo anterior, alguien (un empleado que introduce datos, un vendedor, o un cajero) activa un programa de proceso de transacción e introduce los datos correspondientes. El programa llama al DBMS para modificar la base de datos. Los programas transacción-procesamiento usualmente producen representaciones o respuestas impresas, tales como las confirmaciones de órdenes o los recibos.

► HISTORIA DEL PROCESAMIENTO DE LA BASE DE DATOS

El procesamiento de la base de datos fue utilizado originalmente en corporaciones importantes y en grandes organizaciones, como la base de los sistemas de procesamiento de grandes transacciones. Un ejemplo que ya hemos considerado es la oficina de licencias y registro de vehículos. Después, conforme las microcomputadoras ganaron popularidad, la tecnología de bases de datos emigró a las micros y fue utilizada por un sólo usuario, como las aplicaciones de la base de datos personal que describimos en el caso de Mary Richards. Después, conforme se conectaron en conjunto las micros en grupos de trabajo, la tecnología de bases de datos avanzó hacia grupos de trabajo, como en el ejemplo de Treble Clef. Finalmente, ahora las bases de datos se usan para las aplicaciones de Internet e intranet.

EL CONTEXTO ORGANIZACIONAL

La aplicación inicial de la tecnología de bases de datos fue resolver problemas con el sistema de procesamiento de archivos que analizamos anteriormente. A mediados de 1960 las grandes corporaciones estaban produciendo datos con una rapidez impresionante en los sistemas de procesamiento de archivos, pero esos datos se volvían difíciles de manejar y el desarrollo de los nuevos sistemas era cada vez más complicado. Además, se requería que los procesos de administración fueran capaces de relacionar los datos de un sistema de archivo con los de otro.

Las limitaciones del procesamiento de archivos impidieron la fácil integración de los datos. Sin embargo, la tecnología de bases de datos prometió una solución a estos problemas, y las grandes compañías comenzaron a desarrollar bases de datos organizacionales. Las compañías centralizaron sus datos operativos, tales como órdenes de trabajo, inventarios y datos de contabilidad en estas bases. Las aplicaciones fueron inicialmente sistemas de procesamiento de transacción en el ámbito organizacional.

Primero, cuando la tecnología era nueva, las aplicaciones eran difíciles de desarrollar y había muchas fallas. Incluso las aplicaciones que funcionaban eran lentas y poco confiables, el hardware de la computadora no podía manejar rápidamente el volumen de las transacciones, los técnicos de desarrollo aún no habían descubierto formas más eficientes para almacenar y recuperar datos, y los programadores aún no tenían experiencia en el acceso a bases de datos, o a veces sus programas no trabajaban correctamente.

Las compañías se enfrentaron a otra desventaja del procesamiento de bases de datos: la vulnerabilidad. Si un sistema de procesamiento de archivos fallaba, sólo esa aplicación en particular era eliminada del proceso; pero si la base de datos fallaba, todas las aplicaciones dependientes serían eliminadas.

Gradualmente la situación mejoró. Los ingenieros de software y hardware aprendieron a construir sistemas lo suficientemente poderosos como para manejar muchos usuarios a la vez, con la rapidez suficiente para mantener la carga de trabajo diaria de las transacciones. Se planearon nuevas formas de controlar, proteger y respaldar las bases de datos. Evolucionaron los procedimientos normales para el procesamiento de dichas bases, y los programadores aprendieron a escribir códigos más eficientes y sostenibles. A mediados de la década de 1970, las bases de datos podían manejar eficientemente aplicaciones de procesos organizacionales confiables. La mayoría de estas aplicaciones aún funcionan, ¡después de más de 25 años de haber sido creadas!

EL MODELO RELACIONAL

En 1970, E. F. Codd publicó un artículo¹ de vanguardia en el que aplicó conceptos de una rama de las matemáticas llamada álgebra relacional al problema del almacenamiento de grandes cantidades de datos. El artículo de Codd inició un movimiento en la comunidad vinculada a bases de datos, que en muy pocos años llevó a la definición del **modelo de las bases de datos relacionales**. Éste constituye una forma particular de estructuración y procesamiento de una base de datos y se analiza detalladamente en los capítulos 5 y 9-14.

La ventaja del modelo relacional es que los datos se almacenan de tal forma que minimizan la duplicación y se eliminan ciertos tipos de errores de procesamiento que pueden ocurrir cuando se almacenan datos de otras maneras. Los datos se guardan en tablas, con renglones y columnas, como los de la figura 1-1.

De acuerdo con el modelo relacional, no todas las tablas son igualmente deseables. Utilizando un proceso llamado *normalización*, una tabla se puede cambiar por dos o más que ya existen. En el capítulo 5 aprenderá con mayor detalle el proceso de normalización.

Otra ventaja importante del modelo relacional es que las columnas contienen datos que relacionan un renglón con otro. Por ejemplo, en la figura 1-1 CUSTOMER_ID en la tabla JOB se relaciona con CUSTOMER_ID en la tabla CUSTOMER. Esto hace la relación entre los renglones visibles al usuario.

Al principio se pensaba que el modelo relacional permitiría al usuario obtener información de las bases de datos sin ayuda de los profesionales de MIS. Parte de lo razonable de esta idea era que las tablas conforman construcciones simples que son intuitivamente comprensibles. Además, puesto que las relaciones están almacenadas en los datos, los usuarios podrían combinar los renglones cuando fuera necesario. Por ejemplo, para acceder a un registro de RENTAL, un usuario de la casa Treble Clef podría combinar un renglón de la tabla CUSTOMER con los renglones de la tabla RENTAL.

Lo anterior volvió este proceso demasiado difícil para la mayoría de los usuarios. Por lo tanto, nunca se cumplió la promesa del modelo relacional como medio de acceso a la base de datos para aquellos que no son especialistas. En retrospectiva, el principal beneficio del modelo relacional era que proporcionaba una forma estándar para que los especialistas (¡como usted!) estructuraran y procesaran una base de datos.

PRODUCTOS DBMS PARA MICROCOMPUTADORAS

En 1979 una pequeña compañía llamada Ashton-Tate introdujo un producto para microcomputadora: el dBase II (que se pronuncia “dibeis dos”) al que se le llamó un DBMS relacional. Con una estrategia profesional exitosa, Ashton Tate distribuyó gratis más de 100,000 copias para aquellos que compraron la entonces nueva microcomputadora Osborne. Muchos de los que compraron esas computadoras eran pioneros en la industria de las microcomputadoras. Comenzaron a inventar aplicaciones de microcomputadoras usando dBase, y el número de aplicaciones dBase creció rápidamente. Como resultado, Ashton-Tate se convirtió en una de las primeras y principales corporaciones en la industria de la microcomputación. Después, Borland compró Ashton-Tate, el cual ahora vende la línea de productos dBase.

Sin embargo, el éxito de este producto confundió y desvirtuó el objetivo del procesamiento de la base de datos. El problema era que de acuerdo con la definición prevaleciente a finales de la década de 1970, dBase II no era un DBMS ni tampoco era relacional. En realidad, era un lenguaje de programación con capacidades generalizadas de procesamiento de archivos. Los sistemas que se desarrollaron con dBase II se parecían mucho más a los que se muestran en la figura 1-10 que a los de la figura 1-9. El millón de usuarios de dBase II pensaban que estaban usando un DBMS relacional cuando, en realidad, no era cierto.

Así, los términos *sistema de administración de base de datos* y *base de datos relacional* eran usados libremente cuando comenzó el auge de las microcomputadoras. La mayoría

¹E. F. Codd, “A relational Model of Data for Large Shared Databanks”, *Communications of the ACM*, junio de 1970, pp. 377-387.

de las personas que procesaban una base de datos en microcomputadora realmente administraban archivos y no tenían los beneficios del procesamiento de la base de datos, aunque no se daban cuenta. La situación ha cambiado a medida que el mercado de la microcomputadora se ha vuelto más maduro y sofisticado. Los productos dBase IV y dBase han avanzado, al igual que Foxpro, y son verdaderamente productos DBMS *relacionales*.

Aunque dBase fue un pionero en la aplicación de la tecnología de bases de datos en microcomputadoras, al mismo tiempo otros vendedores comenzaron a trasladar sus productos de las macrocomputadoras a las microcomputadoras. Oracle, Focus e Ingress son tres ejemplos de productos DBMS que voltearon hacia las microcomputadoras. Éstos son realmente productos DBMS y la mayoría estaría de acuerdo en que también son relacionales.

Un impacto del avance de la tecnología con respecto a las bases de datos en la microcomputadora fue la drástica mejoría en las interfaces que emplean los usuarios de los DBMS. Los usuarios de sistemas de microcomputadoras no toleran las torpes y desordenadas interfaces con las que generalmente trabajan los usuarios en los productos DBMS de macrocomputadoras. Así, conforme se inventaron productos DBMS para microcomputadoras, las interfaces de los usuarios se hicieron más fáciles de usar. Esto fue posible porque los productos DBMS operan en computadoras dedicadas y porque computadoras más poderosas están disponibles para procesar la interfaz del usuario. Actualmente los productos DBMS son mejores y más poderosos, con interfaces de usuario gráficas tales como Microsoft Windows.

La combinación de las microcomputadoras, el modelo relacional y las interfaces de usuario bastante mejoradas permitieron que la tecnología de la base de datos pasara de un contexto organizacional a uno de computadora personal. Cuando ocurrió esto, aumentó el número de lugares en los que se usaba la tecnología de las bases de datos. En 1980 había casi 10000 sitios que usaban productos DBMS en Estados Unidos; ahora hay aproximadamente ¡40 millones de sitios!

APLICACIONES DE BASES DE DATOS CLIENTE-SERVIDOR

A mediados de 1980 los usuarios comenzaron a conectar sus microcomputadoras por separado utilizando una red de área local (LAN, por sus siglas en inglés). Dichas redes permitieron a las computadoras el envío de datos de una a otra, a velocidades inimaginables. Las primeras aplicaciones de esta tecnología compartían periféricos, tales como discos rápidos de gran capacidad, impresoras costosas y graficadoras, y facilitaban la intercomunicación a través del correo electrónico. Sin embargo, con el tiempo los usuarios también querían compartir las bases de datos, lo que condujo al desarrollo de aplicaciones multiusuarios en las LAN.

La arquitectura multiusuarios basada en LAN es muy diferente a la que se usa en las bases de datos de macrocomputadoras. Con una de éstas sólo se requiere una CPU en el procesamiento de la aplicación de base de datos, pero con los sistemas LAN pueden confluir simultáneamente varias CPU. El hecho de que esta situación fuera ventajosa (mayor funcionalidad), pero problemática (coordinar las acciones independientes de las CPU), condujo a un nuevo estilo de procesamiento de bases de datos multiusuarios, que se denomina **arquitectura de base de datos cliente-servidor**.

No todo el procesamiento de bases de datos en una LAN es un proceso cliente-servidor. Una forma simple de procesamiento, aunque menos poderosa, se llama **arquitectura de archivos compartidos**. Una compañía como Treble Clef podría usar mejor esta última, puesto que su organización es pequeña y tiene requerimientos de procesamiento modestos. Sin embargo, los grandes grupos de trabajo podrían requerir un procesamiento cliente-servidor. En el capítulo 17 describiremos estos planteamientos y los analizaremos detalladamente.

BASES DE DATOS UTILIZANDO LA TECNOLOGÍA DE INTERNET

Como se mostró en el ejemplo del centro de reservaciones de la isla Calvert, la tecnología de bases de datos se usa en conjunto con la de Internet para publicar bases de da-

tos en la Web. Esta misma tecnología se usa para publicar aplicaciones sobre intranets corporativas y organizacionales. Algunos expertos creen que con el tiempo todas las aplicaciones de las bases de datos serán distribuidas con el uso de exploradores y de las tecnologías de Internet relacionadas, incluso bases de datos personales que sean “editadas” para una sola persona.

Así, son dos categorías de aplicaciones de bases de datos las que usan la tecnología Internet. Una consiste en las aplicaciones puras de la base de datos en la Web, tales como las aplicaciones de la isla Calvert; otra consiste en la base personal tradicional, grupos de trabajo, y bases de datos organizacionales que no se publican en Internet, pero que usan la tecnología de los exploradores, DHTML y XML. Debido a que es correcto referirse a esta segunda categoría como las bases de datos de Internet, este texto se referirá a ambas categorías como *bases de datos que utilizan la tecnología de Internet*.

Esta categoría maneja la carga actual de la tecnología de las bases de datos. También, como se describe en el capítulo 14, la XML en particular satisface las necesidades de las aplicaciones de las bases de datos excepcionalmente buenas, y es la base de muchos nuevos productos y servicios de base de datos.

PROCESAMIENTO DE BASES DE DATOS DISTRIBUIDAS

Antes de concluir este estudio sobre la historia del procesamiento de las bases de datos, necesitamos analizar dos aspectos que son importantes en teoría, pero que no han sido ampliamente adoptados. El primer procesamiento son las bases de datos distribuidas y el segundo, las bases de datos orientadas a objetos. Analizaremos estos temas con más detalle en los capítulos 17 y 18, respectivamente.

Las aplicaciones de bases de datos organizacionales resuelven los problemas del procesamiento de archivos y permiten un procesamiento de datos organizacionales más integrado. Los sistemas de base de datos personal y de grupos de trabajo causan que la tecnología de las bases de datos esté aún más cerca de los usuarios para permitir el acceso a la base de datos de administración local. Las **bases de datos distribuidas** combinan estos tipos de procesamientos permitiendo las personales, las de grupos de trabajo y las organizacionales para combinarse en bases de datos integradas, pero distribuidas. Así, en teoría, ofrecen aun mayor flexibilidad de acceso a los datos y al procesamiento, pero por desgracia también tienen muchos problemas sin resolver.

La esencia de las bases de datos distribuidas es que se divulgan todos los datos de la organización en muchas computadoras —microcomputadoras, servidores LAN y macrocomputadoras— que se comunican entre sí conforme efectúan los procesos. Los objetivos de los sistemas de bases de datos distribuidas son hacer parecer que cada usuario es el único que tiene acceso a los datos de la organización y proporcionar la misma consistencia, exactitud y menor tiempo del que el usuario podría ocupar si no estuviera usando la base de datos distribuida.

Entre los problemas más apremiantes de las bases de datos distribuidas están los de seguridad y control. Permitir que muchos usuarios (pueden ser cientos) tengan acceso a la base de datos y controlar lo que hacen son tareas complicadas.

Los procesos de sincronización y coordinación pueden ser difíciles. Si un grupo de usuarios baja y actualiza parte de la base de datos y después envía esos cambios a la macrocomputadora, ¿cómo puede evitar el sistema que mientras tanto otro usuario intente usar la versión de los datos que se encuentra en la macrocomputadora? Imagine que este problema involucra docenas de archivos y cientos de usuarios que usan muchísimas piezas del equipo de cómputo.

Mientras que las transiciones del procesamiento de las bases de datos, desde la organizacional hasta la personal y las de los grupos de trabajo, eran relativamente fáciles, las dificultades que enfrentaban los diseñadores de la base de datos y los ingenieros del DBMS distribuido eran monumentales. En realidad, aun cuando durante más de 25 años se ha desarrollado trabajo sobre sistemas de bases de datos distribuidas, todavía persisten problemas importantes. Microsoft ha definido y está construyendo una arquitectura de procesamiento de distribución y una serie de productos de apoyo llama-

do **servidor de transacciones Microsoft (MTS)**, por sus siglas en inglés). Aun cuando MTS lo ha prometido —y entre todas las compañías Microsoft es la que tiene los recursos para desarrollar y comercializar un sistema así— no se sabe con certeza si las bases de datos distribuidas podrán satisfacer las necesidades del proceso organizacional actual. Para más información sobre este tema vea el análisis de BD OLE en el capítulo 15.

LOS DBMS ORIENTADOS A OBJETOS (ODBMS)

A finales de la década de 1980 se comenzó a usar un nuevo estilo de programación llamado *programación orientada a objetos* (OOP, por sus siglas en inglés), el cual tiene una orientación muy diferente a la programación tradicional, como se explica en el capítulo 18. En resumen, las estructuras de datos procesados con OOP son mucho más complicadas que las desarrolladas con lenguajes tradicionales. Estas estructuras de datos también son difíciles de almacenar en los productos relacionales DBMS. Como resultado, una nueva categoría de productos DBMS llamados *sistemas de bases de datos orientadas a objetos* está evolucionando para almacenar y procesar estructuras de datos OOP.

Por muchas razones, la OOP aún no se ha usado ampliamente en los sistemas de información de negocios. En primer lugar, las aplicaciones OOP son muy difíciles de usar y es muy costoso desarrollarlas. En segundo, la mayoría de las organizaciones tienen millones o billones de bytes de datos que ya están organizados en bases de datos relacionales y están poco dispuestas a enfrentar el costo y el riesgo de convertir esas bases de datos a un formato ODBMS. Por último, la mayoría de los ODBMS que se han desarrollado apoyan aplicaciones de ingeniería y no tienen características y funciones que sean apropiadas o rápidamente adaptables a las aplicaciones de negocios de información.

Consecuentemente, para el futuro previsible, los ODBMS ocupan un nicho en las aplicaciones de sistemas de información comerciales. En el capítulo 18 analizaremos OOP, bases de datos orientadas a objetos, y una forma híbrida de Oracle llamada base de datos orientada a objetos, pero la mayor parte del análisis se enfoca en el modelo relacional, puesto que las tecnologías se refieren a lo que usted comúnmente usará durante los primeros cinco años de su carrera.

► RESUMEN

El procesamiento de bases de datos es uno de los cursos más importantes en la currícula de sistemas de información. Las habilidades y los conocimientos sobre bases de datos tienen gran demanda, no sólo en cuanto a las aplicaciones tradicionales, sino también a las aplicaciones que usan la tecnología de Internet para redes públicas y privadas.

La tecnología de bases de datos se usa en muchas aplicaciones. Algunas sirven para un usuario en una sola computadora; otras se utilizan para grupos de trabajo de 20 a 30 personas en una LAN; otras más las utilizan cientos de usuarios e implican billones de bytes de datos. Recientemente, la tecnología de bases de datos se ha combinado con la de Internet para apoyar las aplicaciones multimedia en redes públicas y privadas.

Los componentes de una aplicación de base de datos son la base de datos misma, el sistema de administración de ésta (los DBMS) y los programas de aplicación. Algunas veces dichos programas están separados por completo del DBMS; otras, algunas partes muy importantes de la aplicación se proporcionan mediante características y funciones del DBMS.

Los sistemas de procesamiento de datos almacenan información en archivos por separado, y cada uno contiene un tipo de datos diferente. Los sistemas de procesamiento de archivos tienen varias limitaciones. Con archivos por separado, es difícil combinar los datos almacenados en archivos diferentes, como los datos que con frecuencia se duplican entre los archivos, lo cual da como resultado problemas de integri-

dad. Los programas de aplicación dependen de los formatos de archivo, lo cual causa problemas de mantenimiento cuando cambian los formatos y los archivos se hacen incompatibles; por lo tanto, se requiere la conversión de archivos. También es difícil representar datos desde las perspectivas de los usuarios.

Los sistemas de procesamiento de bases de datos se desarrollaron para superar estas limitaciones. En el medio ambiente de las bases de datos, el DBMS es la interfaz entre los programas de aplicación y las bases de datos. Los datos están integrados y se reduce la duplicación. Los cambios en los formatos físicos de los datos almacenados sólo afectan al DBMS. Si se cambian, agregan o borran conjuntos de datos, pocos programas de aplicación requerirán mantenimiento. Con la tecnología de bases de datos es fácil representar objetos en el medio ambiente de los usuarios.

Una base de datos es un conjunto autodescriptivo de registros integrados. Es auto-descriptivo porque contiene una descripción de sí misma en un diccionario de datos, el cual también se conoce como directorio de datos o metadatos. Una base de datos es un conjunto de registros integrados porque la relación entre éstos se almacena en la base. Este arreglo permite al DBMS construir incluso objetos complicados mediante la combinación de datos con base en las relaciones almacenadas. Las relaciones con frecuencia se almacenan en las bases de datos importantes. Así, las tres partes que conforman una base son: la aplicación de datos, el diccionario y los datos importantes.

La tecnología de bases de datos se desarrolló en varias etapas. Las primeras bases se concentraban en el procesamiento de transacción de datos organizacionales. Después, el modelo relacional, junto con las microcomputadoras, permitió al usuario el uso de aplicaciones personalizadas. Con la llegada de las LAN, los departamentos comenzaron a implementar bases de datos cliente-servidor para grupos de trabajo. Actualmente, Internet y las aplicaciones tradicionales están siendo repartidas mediante el uso de la tecnología de Internet. El procesamiento de distribución y bases orientadas a objetos son temas importantes en el procesamiento de bases de datos. Sin embargo, hasta el momento no han tenido éxito comercial ni su uso se ha difundido ampliamente en las aplicaciones para los negocios.

► PREGUNTAS DEL GRUPO 1

- 1.1 ¿Por qué es un tema importante el procesamiento de las bases de datos?
- 1.2 Describa la naturaleza y las características de una aplicación de base de datos para un solo usuario, como la que se empleó en el caso de Mary Richards.
- 1.3 Describa la naturaleza y las características de una aplicación de bases de datos que utiliza un grupo de trabajo, como en el caso de la casa de música Treble Clef.
- 1.4 Describa la naturaleza y las características de una aplicación de bases de datos que emplea una organización como la de la oficina de licencias y registro de vehículos.
- 1.5 Describa la naturaleza y las características de una aplicación de bases de datos en una organización como la del centro de reservación de la isla Calvert.
- 1.6 Explique la naturaleza y la función de cada uno de los componentes de la figura 1-9.
- 1.7 ¿Cómo cambia con el tiempo la relación entre los programas de aplicación y el DBMS?
- 1.8 Enumere las limitaciones de los sistemas de procesamiento de archivos, como se describió en este capítulo.
- 1.9 Explique de qué manera la tecnología de la base de datos superó las limitaciones que enumeró en su respuesta de la pregunta 1.8.
- 1.10 Defina el término *base de datos*.
- 1.11 ¿Qué son los metadatos? ¿Qué son los índices? ¿Cuáles son las aplicaciones de los metadatos?

- 1.12 Explique por qué una base de datos es un modelo. Describa la diferencia entre un modelo de la realidad y un modelo de un modelo de la realidad de un usuario. ¿Por qué es importante esta diferencia?
- 1.13 Dé un ejemplo, diferente al de este capítulo, sobre una aplicación de bases de datos personales.
- 1.14 Proporcione un ejemplo, diferente al de este capítulo, referente a una aplicación de base de datos de un grupo de trabajo.
- 1.15 Mencione un ejemplo, diferente al que se menciona en este capítulo, sobre una aplicación de bases de datos en una empresa grande.
- 1.16 ¿Cuáles eran algunas de las debilidades de las primeras aplicaciones de las bases de datos?
- 1.17 ¿Cuáles son dos de las principales ventajas del modelo relacional?
- 1.18 Resuma los eventos en el desarrollo de los productos DBMS para microcomputadora.
- 1.19 ¿Cuál fue el factor principal que dio lugar a las aplicaciones de la base de datos de grupo de trabajo?
- 1.20 ¿Cuál es la diferencia entre las arquitecturas cliente-servidor y las multiusuarios?
- 1.21 ¿Cuál es la diferencia entre una aplicación de base de datos de Internet y una que usa la tecnología de Internet?
- 1.22 Explique la naturaleza general del procesamiento distribuido. ¿Cuáles son algunos de los problemas que representa?
- 1.23 Describa el propósito de una base de datos orientada a objetos. ¿Por qué estas bases de datos no han sido las más aceptadas para las aplicaciones de sistemas de información?

► PROYECTOS



A. Ingrese al sitio Web de un fabricante de computadoras, como por ejemplo Dell (www.dell.com). Use el sitio Web para determinar qué modelo de computadora portátil (*laptop*) recomendaría con un costo máximo de \$2500*. ¿Cree que se usan una o más bases de datos para manejar este sitio? Si es así, mencione cuáles funciones o características del sitio Web podrían ayudar a la tecnología de las bases de datos, considerando tanto la definición de una base de datos como las ventajas de su procesamiento.



B. Entre al sitio Web de un vendedor de libros al detalle, como por ejemplo Amazon (www.amazon.com). Use el sitio Web para localizar la biografía más reciente de William Wordsworth. Desde su punto de vista, ¿cree que se usan una o más bases de datos para este sitio? Si es así, ¿cuáles características y funciones del sitio Web cree que deberían ser de más ayuda para la tecnología de base de datos, considerando tanto las definiciones de una base de datos como las ventajas del procesamiento de base de datos?

► PREGUNTAS DEL PROYECTO FIREDUP

La compañía FiredUp es un negocio pequeño, cuyos propietarios son Curt y Julia Robard. La sede está en Brisbane, Australia. FiredUp fabrica y vende una estufa para campo ligera, llamada FiredNow. Curt, quien trabajó previamente como ingeniero aeroes-

*Tome en cuenta el lector que todas las cifras que aparecen en este libro se expresan en dólares estadounidenses, a menos que se indique lo contrario. (N. de la R.)

pacial, inventó y patentó un mechero con el que la estufa permanecerá encendida aun cuando el viento sea superior a los 160 kilómetros por hora. Julia, una diseñadora industrial muy capaz, desarrolló un elegante diseño plegadizo pequeño, ligero, fácil de instalar y muy estable. Los Robard fabrican las estufas en su cochera y las venden directamente a sus clientes a través de Internet, fax o correo.

Los propietarios de FiredUp necesitan dar seguimiento a las estufas que han vendido, con el fin de poder entrar en contacto con sus usuarios en caso de fallas del producto u otros asuntos relacionados con éste. También piensan que podrían usar su lista de clientes para comercializar otros productos que desarrollen en el futuro.

A. ¿Qué base de datos cree que sería la apropiada para que FiredUp pudiera dar seguimiento a sus estufas y dispusiera de los datos de sus clientes? Explique en qué circunstancias sería apropiada una base de datos y en cuáles no sería adecuada. Describa cuándo convendría una base de datos personal. ¿En qué casos sería conveniente una base de datos de un grupo de trabajo? ¿En qué circunstancias le convendría a FiredUp tener una base de datos en Internet?

B. Aplique el mismo problema al registro de un producto que vende la compañía Café Starbucks. Digamos, por ejemplo, que dicha compañía quiere desarrollar la capacidad de dar seguimiento a los clientes que le compran máquinas para café exprés. ¿Cómo responde a las preguntas del inciso A para este caso en particular?

Introducción al desarrollo de una base de datos

Este capítulo presenta un panorama sobre el desarrollo y la aplicación de una base de datos. Comenzamos con una descripción de los elementos que conforman una base de datos y con un análisis de las características y funciones de un DBMS. A continuación ilustramos la creación y el funcionamiento de una base de datos. Por último, analizamos las estrategias comunes del desarrollo de dicha base. El objetivo del presente capítulo es crear una perspectiva para abordar las descripciones detalladas de la tecnología en los siguientes capítulos.

► LA BASE DE DATOS

La figura 2-1 muestra los componentes principales de un sistema de base de datos. El **DBMS** procesa la **base de datos**, y lo utilizan tanto los programadores como los usuarios, quienes pueden ingresar al DBMS directa o indirectamente mediante los **programas de aplicación**. En esta sección analizaremos la base de datos y en las siguientes secciones, el DBMS y los metadatos de aplicaciones.

Como establecimos en el capítulo anterior, una base de datos contiene cuatro elementos principales: los datos del usuario, los metadatos, los índices y los metadatos de aplicaciones.

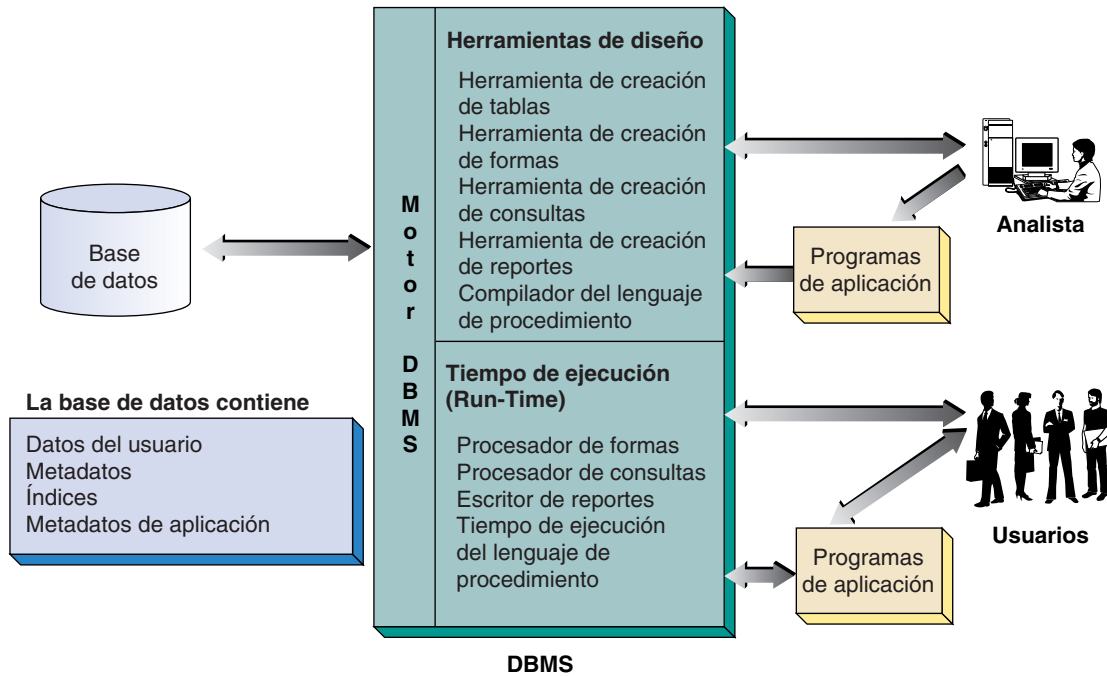
DATOS DEL USUARIO

Actualmente la mayoría de las bases de datos representan a los datos del usuario como relaciones. El término *relación* lo definiremos formalmente en el capítulo 5. Por el momento, considere una relación como una tabla de datos. Las columnas de ésta contienen campos o atributos, y los renglones, registros de entidades particulares en el campo de los negocios.

No todas las relaciones son igualmente convenientes, algunas están mejor estructuradas que otras. El capítulo 5 describe un proceso llamado normalización, que se uti-

► FIGURA 2-1

Componentes de los sistemas de bases de datos



liza para crear relaciones bien estructuradas. Para tener una idea sobre la diferencia entre las relaciones pobremente estructuradas y las bien estructuradas, considere la relación R1 (Nombre del Estudiante, Teléfono del Estudiante, Nombre del Asesor, Teléfono del Asesor) con los siguientes datos:

Nombre del Estudiante	Teléfono del Estudiante	Nombre del Asesor	Teléfono del Asesor
Baker, Rex	232-8897	Parks	236-0098
Charles, Mary	232-0099	Parks	236-0098
Johnson, Beth	232-4487	Jones	236-0110
Scott, Glenn	232-4444	Parks	236-0098
Zylog, Frita	232-5588	Jones	236-0110

El problema con esta relación es que contiene datos respecto a dos temas diferentes: estudiantes y asesores. Una relación estructurada de esta manera presenta diversos problemas cuando se actualiza. Por ejemplo, si el asesor Parks cambia su número telefónico, habrá que modificar tres renglones de datos. Por esta razón, sería mejor representar los datos mediante las dos relaciones R2 (Nombre del Estudiante, Teléfono del Estudiante, Nombre del Asesor) con los datos:

Nombre del Estudiante	Teléfono del Estudiante	Nombre del Asesor
Baker, Rex	232-8897	Parks
Charles, Mary	232-0099	Parks
Johnson, Beth	232-4487	Jones
Scott, Glenn	232-4444	Parks
Zylog, Frita	232-5588	Jones

y R3 (Nombre del Asesor, Teléfono del Asesor) con los datos:

Nombre del Asesor	Teléfono del Asesor
Parks	236-0098
Jones	236-0110

Si un asesor cambia su teléfono sólo habrá que modificar un renglón. Por supuesto, para producir un reporte que muestre los nombres de los estudiantes junto con los números telefónicos de los asesores, se tendrían que combinar los renglones de estas dos tablas. Sin embargo, almacenar por separado las relaciones y combinarlas cuando se produce un reporte resulta mucho mejor que almacenarlas como una tabla combinada.

METADATOS

Como establecimos en el capítulo 1, una base de datos se describe por sí misma, lo que significa que contiene una descripción de su estructura como parte de sí misma. Esta descripción se llama **metadatos**. Debido a que los productos DBMS están diseñados para almacenar y manejar tablas, la mayoría de los productos almacenan los metadatos en forma de tablas, a las que a veces se denomina **tablas del sistema**.

La figura 2-2 muestra un ejemplo de los metadatos guardados en las dos tablas del sistema. La primera almacena una lista de tablas que se encuentran en la base de datos, indicando cuántas columnas se encuentran en cada tabla y qué columna(s) se encuentra(n) en la llave primaria. Dicha columna es la única que identifica un renglón. La segunda tabla almacena una lista de columnas en cada tabla y el tipo y longitud de los datos de cada columna. Estas dos tablas son comunes de las tablas de sistema; otras parecidas almacenan listas de índices, llaves, procedimientos almacenados y cosas por el estilo.

► FIGURA 2-2

Ejemplos de metadatos

Tabla del sistema de tablas

Nombre de la tabla	Número de columnas	Llave primaria
Estudiante	4	Número de Estudiante
Asesor	3	Nombre del Asesor
Curso	3	Número de Referencia
Registro	3	{Número de Estudiante, Número de Referencia}

Tabla de las columnas del sistema

Nombre de la columna	Nombre de la tabla	Tipo de datos	Longitud*
Número de Estudiante	Estudiante	Entero	4
Nombre	Estudiante	Texto	20
Apellido	Estudiante	Texto	30
Especialidad	Estudiante	Texto	10
Nombre del Asesor	Asesor	Texto	25
Teléfono	Asesor	Texto	12
Departamento	Asesor	Texto	15
Número de Referencia	Curso	Entero	4
Título	Curso	Texto	10
Número de Horas	Curso	Decimal	4
Número de Estudiante	Registro	Entero	4
Número de Referencia	Registro	Entero	4
Calificación	Registro	Texto	2

* Las longitudes están en bytes, que son iguales al número de caracteres de los datos del texto.

Almacenar metadatos en tablas no sólo es eficiente para el DBMS; también conviene para los analistas debido a que pueden utilizar las mismas herramientas de consulta para los metadatos como las que se emplean para los datos del usuario. En el capítulo 9 analizaremos un lenguaje llamado SQL que se utiliza para consultar y actualizar las tablas para los metadatos y los datos del usuario.

Con el propósito de mostrarle un ejemplo de cómo puede utilizar el SQL, suponga que ha desarrollado una base de datos con 15 tablas y 200 columnas. Recuerde que muchas de las columnas contienen el tipo de datos *moneda*, pero no puede recordar cuáles. Al utilizar el SQL puede ingresar a la tabla de las columnas del sistema para descubrir cuáles son las columnas que contienen ese tipo de datos.

ÍNDICES

Un tercer tipo de datos de la base mejora el funcionamiento y el acceso a la base de datos. Estos datos, a los que a veces se les llama **datos significativos**, constan principalmente de índices, aunque algunas veces se utilizan otros tipos de estructuras de datos, tales como las listas vinculadas (véase el Apéndice A para un análisis de índices y de las listas vinculadas).

La figura 2-3 muestra una tabla de datos de estudiantes y dos índices. Para demostrar la utilidad de tener estos índices, suponga que los datos de *NúmerodeEstudiante* están almacenados en un disco en orden ascendente y que el usuario quiere imprimir un reporte de los datos de los estudiantes por apellido. Para efectuarlo, se pueden extraer todos los datos de la tabla fuente y ordenarlos; a menos que ésta sea pequeña, este proceso requiere tiempo. Alternativamente, se puede crear un índice por apellido, como el de la figura 2-3. Las entradas en este índice están ordenadas por el valor de *Apellido*, por lo que las entradas del índice se pueden leer y utilizar para ingresar los datos de los estudiantes por orden.

► FIGURA 2-3

Ejemplos de índices de una base de datos

Ejemplo de tabla ESTUDIANTE

NúmerodeEstudiante	Nombre	Apellido	Especialidad
100	James	Baker	Contaduría
200	Mary	Abernathy	Sistemas de información
300	Beth	Jackson	Contaduría
400	Eldridge	Johnson	Mercadotecnia
500	Chris	Tufte	Contaduría
600	John	Smathers	Sistemas de información
700	Michael	Johnson	Contaduría

Índice por apellido

Apellido	NúmerodeEstudiante
Abernathy	200
Baker	100
Jackson	300
Johnson	400, 700
Smathers	600
Tufte	500

Índice por especialidad

Especialidad	NúmerodeEstudiante
Contaduría	100, 300, 500, 700
Sistemas de información	200, 600
Mercadotecnia	400

Ahora suponga que los datos de los estudiantes deben imprimirse junto con la especialidad de éstos. De nuevo, se pueden extraer los datos de la tabla de referencia y clasificarlos, o se puede construir un índice como el índice por especialidad y utilizarlo como lo describimos.

Los índices no sólo se utilizan para la clasificación, sino para el rápido acceso a los datos. Por ejemplo, un usuario desea ingresar sólo aquellos estudiantes que tienen el valor de “Sistemas de información” en la especialidad. Sin un índice, habría que revisar toda la tabla de referencia; pero cuando éste existe, se puede encontrar la entrada del índice y utilizarla para localizar todos los renglones idóneos. Aunque no se necesitan índices para una tabla con pocos renglones, como la de ESTUDIANTE en la figura 2-3, considere una tabla que tenga 10000 o 20000 renglones de datos. En ese caso, la clasificación y la revisión serían muy lentas.

Los índices son útiles para las operaciones de clasificación y revisión, pero tienen un costo. Cada vez que se actualiza un renglón en la tabla ESTUDIANTE, también se deben actualizar los índices. Esto no necesariamente es malo; sólo significa que los índices no son independientes y por lo tanto deben reservarse para los casos en que sean realmente necesarios.

METADATOS DE APLICACIÓN

El cuarto y último tipo de datos en la base son los **metadatos de aplicación**, los cuales se utilizan para almacenar la estructura y el formato de formas del usuario, reportes, consultas y otros componentes de aplicación. No todos los productos DBMS sustentan componentes de aplicación, y en aquellos productos que lo hacen no todos almacenan la estructura de dichos componentes como metadatos de aplicación en la base de datos. Sin embargo, la mayoría de los productos DBMS modernos sí almacenan estos datos como parte de la base. En general, ni los analistas ni los usuarios ingresan directamente a los metadatos de aplicación, sino que utilizan herramientas del DBMS para procesarlos.

► DBMS

Los productos DBMS varían considerablemente en cuanto a características y funciones. Los primeros de estos productos se desarrollaron en la década de 1960 para ser usados en macrocomputadoras y tenían características muy primitivas. Desde entonces, los productos DBMS han sido continuamente perfeccionados y mejorados no sólo para procesar mejor los datos de las bases, sino también para incorporar características que faciliten la creación de las aplicaciones de dichas bases.

En este capítulo utilizaremos Microsoft Access 2002 para ilustrar las capacidades de los productos DBMS. Esto obedece a que Access 2002 tiene las características y funciones de un DBMS moderno. Sin embargo, no es el único DBMS y su elección en este caso no significa una preferencia especial por encima de otros productos similares, como el Lotus Approach.

Como se muestra en la figura 2-1, las características y funciones de un DBMS pueden dividirse en tres subsistemas: herramientas de diseño, tiempo de ejecución y motor DBMS.

EL SUBSISTEMA DE HERRAMIENTAS DE DISEÑO

El subsistema de herramientas de diseño consta de un conjunto de implementos que facilitan el diseño y la creación de la base de datos y de sus aplicaciones. Por lo general incluyen herramientas para la creación de tablas, formas, consultas y reportes. Los productos DBMS también proporcionan lenguajes e interfaces de programación para programar con

lenguajes. Por ejemplo, Access tiene dos lenguajes: un macrolenguaje que no requiere conocimientos profundos sobre programación y una versión de BASIC llamada Visual Basic.

SUBSISTEMA DEL RUN-TIME (TIEMPO DE EJECUCIÓN)

El subsistema run-time¹ procesa los componentes de aplicación que se desarrollan al utilizar las herramientas de diseño. Por ejemplo, Access 2002 tiene una facilidad de run-time que materializa las formas y conecta los elementos de la forma con los datos de la tabla. Suponga que se ha descrito una forma que incluye un cuadro de texto, el cual expone el valor de `NúmerodeEstudiante` de la tabla `ESTUDIANTE`. Durante la ejecución, cuando se abre la forma, el procesador run-time de ésta extrae el valor del `NúmerodeEstudiante` del renglón actual en la tabla `ESTUDIANTE` y lo despliega en la forma. Todo esto es automático; una vez que se crea la forma, ni el usuario ni el analista necesitan hacer nada. Otros procesadores run-time responden a consultas e imprimen reportes. Además, hay un componente del run-time que procesa los requerimientos del programa de aplicación para leer y escribir los datos de la base.

Aunque no se muestra en la figura 2-1, también los productos DBMS deben proporcionar una interfaz para los lenguajes estándares tales como el C++ y el Java. Usted aprenderá más acerca de esto en los capítulos 15 y 16.

EL MOTOR DBMS

El tercer componente del DBMS es el motor DBMS, el cual es el intermediario entre las herramientas de diseño y los subsistemas run-time y los datos. El motor DBMS recibe los requerimientos de los otros dos componentes —planteados en términos de tablas, renglones y columnas— y traduce los requerimientos en órdenes dirigidas al sistema operativo para leer y escribir datos en medios físicos.

El motor DBMS también está involucrado en el manejo de transacciones, bloqueo, respaldo y recuperación. Como se muestra en el capítulo 11, las acciones en contra de la base de datos con frecuencia deben hacerse como un entero. Cuando se procesa una orden, por ejemplo los cambios en las tablas `CLIENTE`, `PEDIDO` e `INVENTARIO` deben realizarse en conjunto; o se hace en todas o en ninguna. El motor DBMS ayuda a coordinar las actividades para asegurarse de que se aplicó todo o nada del grupo.

Microsoft proporciona dos motores distintos para Access 2002: el motor Jet y el SQL Server; el primero se utiliza para pequeñas bases de datos personales y de trabajo en grupo. El SQL Server, el cual es un producto independiente de Microsoft, se utiliza para grandes bases de datos departamentales y organizacionales de tamaños pequeño a mediano. Cuando usted crea una base de datos utilizando las capacidades originales para generación de tablas de Access 2002 (bases de datos almacenadas con el sufijo `.mdb`), está utilizando el motor Jet (Jet Engine). Cuando crea proyectos de Access 2002 (con sufijo `.adp`) está creando una interfaz de aplicación para el motor del SQL Server.

► CREACIÓN DE LA BASE DE DATOS

El *esquema de una base de datos* define la estructura de ésta, sus tablas, relaciones, dominios y reglas del negocio. El esquema de una base de datos es un diseño, la base sobre la cual se construyen ésta y las aplicaciones.

¹No confunda el término *subsistema run-time* con el término *producto de run-time*. Algunos vendedores utilizan el término *producto de run-time* para referirse a aquel que incluye los componentes run-time y el motor DBMS, pero no al subsistema de las herramientas de diseño. Dicho producto puede utilizarse para procesar una aplicación que ya ha sido desarrollada. El propósito de los productos run-time es reducir el costo de la aplicación al usuario final. Por lo general, el producto run-time es menos costoso (a veces gratuito) que el DBMS completo. De ahí que sólo el analista compre el producto completo, que incluye el subsistema de las herramientas de diseño; los usuarios finales sólo compran el producto run-time.

EJEMPLO DE UN ESQUEMA

Para ilustrar un esquema y saber por qué es importante, considere un ejemplo. El Colegio Highline es una pequeña escuela que imparte actividades físicas. Su departamento de actividades estudiantiles patrocina ligas locales de atletismo, y el gran problema consiste en no perder de vista al equipo de atletismo, que ha sido asignado a varios capitanes de equipo. Para este sistema se utilizan los siguientes componentes del esquema:

TABLAS La base de datos contiene dos tablas:²

CAPTAIN (CaptainName, Phone, Street, City, State, Zip)
(CAPITÁN [Nombre del Capitán, Teléfono, Calle, Ciudad, Estado, Código Postal])

y

ITEM (Quantity, Description, DateOut, DateIn)
(ARTÍCULO [Cantidad, Descripción, Fecha de Salida, Fecha de Entrada])

donde los nombres de las tablas se muestran fuera de los paréntesis y los de las columnas, entre paréntesis.

Ni CaptainName ni Description son necesariamente un nombre único; dos capitanes podrían fácilmente llamarse “Mary Smith”, o seguramente hay muchos artículos denominados “Soccer Balls”. Para asegurarse de que cada renglón puede ser identificado (la importancia de esto se aclarará en capítulos posteriores), se agregarán dos columnas de números únicos a estas tablas, como se muestra a continuación:

CAPTAIN (CAPTAIN_ID, CaptainName, Phone, Street, City, State, Zip)
(CAPITÁN [CAPITÁN_ID, Nombre del Capitán, Teléfono, Calle, Ciudad, Estado, Código Postal])

ITEM (ITEM_ID, Quantity, Description, DateOut, DateIn)
(ARTÍCULO [ARTÍCULO_ID, Cantidad, Descripción, Fecha de Salida, Fecha de Entrada])

RELACIONES. La relación entre estas dos tablas es como sigue: un renglón de CAPTAIN se relaciona con muchos renglones de ITEM; sin embargo, un renglón ITEM se relaciona con un renglón CAPTAIN. La notación para una relación como ésta es **1:N** y se pronuncia “uno a N” o “uno a muchos”. El término **1:N** significa que un renglón de la primera tabla se relaciona con muchos renglones de la segunda.

Para las tablas que se muestran aquí no hay manera de distinguir qué renglón de CAPTAIN se relaciona con otros renglones de ITEM. De ahí que, para mostrar dicha relación, se le agrega CAPTAIN_ID al renglón ITEM. La estructura completa de las dos tablas es la siguiente:

CAPTAIN (CAPTAIN_ID, CaptainName, Phone, Street, City, State, Zip)
ITEM (ITEM_ID, Quantity, Description, DateOut, DateIn, CAPTAIN_ID)

Con esta estructura es fácil determinar qué capitán ha tomado prestado determinado ITEM. Por ejemplo, para descubrir quién pidió el artículo 1234, se examina el renglón correspondiente y se encuentra el valor de CAPTAIN_ID almacenado en tal renglón. Entonces se puede utilizar dicho valor para almacenar el nombre y el número telefónico de ese capitán.

DOMINIOS. Un dominio³ es un conjunto de valores que puede tener una columna. Considere los dominios de las columnas de la tabla ITEM. Suponga que tanto CAPTAIN_ID como Quantity son números enteros, que Description es un texto de longitud 25, que

² Como se muestra en los capítulos 3 al 7, la tarea más importante y difícil para el desarrollo de una base de datos es diseñar la estructura de la tabla. Al iniciar este ejemplo con las tablas ya definidas se ha pasado por alto una gran parte del proyecto.

³ Este análisis se simplifica considerablemente al concentrarse en los componentes del sistema de una base de datos. En el capítulo 4 encontrará un análisis más completo sobre los dominios.

DateOut y DateIn tienen el dominio de la fecha, y que también CAPTAIN_ID tiene el dominio de números enteros. Además del formato físico, también se necesita decidir si alguno de los dominios será único para la tabla. En este ejemplo, quisimos que ITEM-ID fuera único, y por lo tanto se debe especificar su dominio de esta manera. Puesto que un capitán puede haber tomado más de un artículo, CAPTAIN_ID no es único para la tabla ITEM.

También se deben especificar los dominios de las columnas CAPTAIN. CAPTAIN_ID es entero y el resto de las columnas son textos de diferentes longitudes. CAPTAIN_ID debe ser único en la tabla CAPTAIN.

REGLAS DEL NEGOCIO. El último elemento del esquema de una base de datos son las reglas del negocio, las cuales son restricciones en las actividades del negocio y necesitan reflejarse en la base de datos al igual que en sus aplicaciones. Los siguientes son ejemplos de reglas del negocio del Colegio Highline:

1. Para pedir prestado cualquier artículo, un capitán debe tener un número telefónico local.
2. En ningún momento ningún capitán tendrá en préstamo más de siete balones de fútbol.
3. Los capitanes deben regresar todos los artículos cinco días después de que termine el semestre.
4. Ningún capitán podrá pedir más equipo si ha retrasado la entrega de algún artículo.

Las reglas del negocio son una parte muy importante del esquema, debido a que especifican las restricciones de los valores de datos permitidos que deben imponerse, sin importar la manera en que los cambios de los datos afecten al motor DBMS. Independientemente de que la petición de un cambio de datos provenga del usuario de una forma, de una consulta-requerimiento actualizada, o de un programa de aplicación, el DBMS debe asegurarse de que el cambio no viole ninguna regla.

Por desgracia, las reglas del negocio son impuestas de diferentes maneras por distintos productos DBMS. Con Access 2002 pueden definirse ciertas reglas en el esquema y ser puestas en práctica automáticamente. Con productos como el SQL Server y ORACLE, se imponen reglas del negocio adicionales mediante un dispositivo denominado *procedimientos almacenados* (*stored procedures*). En algunos casos, el producto DBMS no tiene capacidad para imponer las reglas del negocio necesarias y éstas deben codificarse como programas de aplicación. Analizaremos este tema con mayor detalle en el capítulo 10.

CREACIÓN DE TABLAS

Una vez que se ha diseñado el esquema, el siguiente paso es crear tablas de bases de datos utilizando las herramientas para la creación de tablas del DBMS. La figura 2-4 muestra la forma que se utiliza con Microsoft Access para crear la tabla ITEM. El nombre de cada columna de la tabla está representado en la columna Field Name, y la representación de los datos de la columna se especifica en la columna Data Type. La columna Description se utiliza para registrar la documentación opcional y los comentarios con respecto a las columnas de la tabla. Los datos adicionales —como longitud del texto, formato del campo, título y otros— se especifican en los campos de entrada en la columna de la forma inferior a mano izquierda.

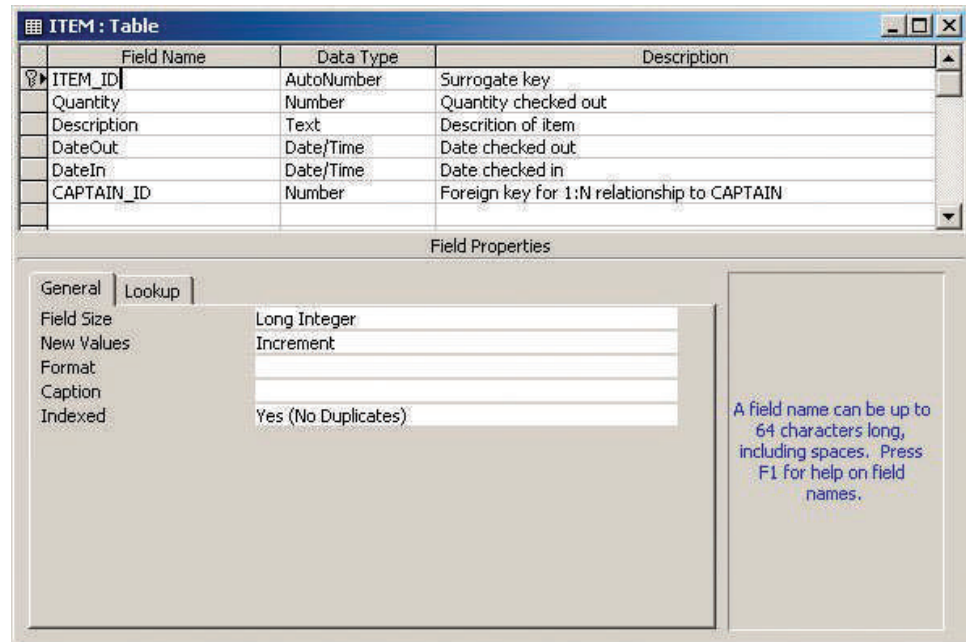
En la figura 2-4 la información importante se encuentra en la columna ITEM_ID. Observe que la propiedad indexada en la parte inferior de la forma ha sido programada como Yes (No Duplicates), lo que significa que un índice de valores únicos se creará para la columna ITEM_ID. Para completar la definición de la base de datos, la tabla CAPTAIN se crea de una manera similar.

DEFINICIÓN DE RELACIONES

La relación entre CAPTAIN e ITEM es 1:N, lo cual se representa en el esquema mediante la colocación de la llave de CAPTAIN en la tabla ITEM. En la figura 2-4 colocamos CAPTAIN_ID en la tabla ITEM. Una columna como la de CAPTAIN_ID en la tabla ITEM a veces se denomina **llave externa** debido a que es la llave de una tabla que es externa a la tabla en la cual reside. Cuando se crean formas, consultas y reportes, el DBMS puede pro-

► FIGURA 2-4

Creación de una
tabla con Microsoft
Access 2002



Field Name	Data Type	Description
ITEM_ID	AutoNumber	Surrogate key
Quantity	Number	Quantity checked out
Description	Text	Description of item
DateOut	Date/Time	Date checked out
DateIn	Date/Time	Date checked in
CAPTAIN_ID	Number	Foreign key for 1:N relationship to CAPTAIN

Field Properties

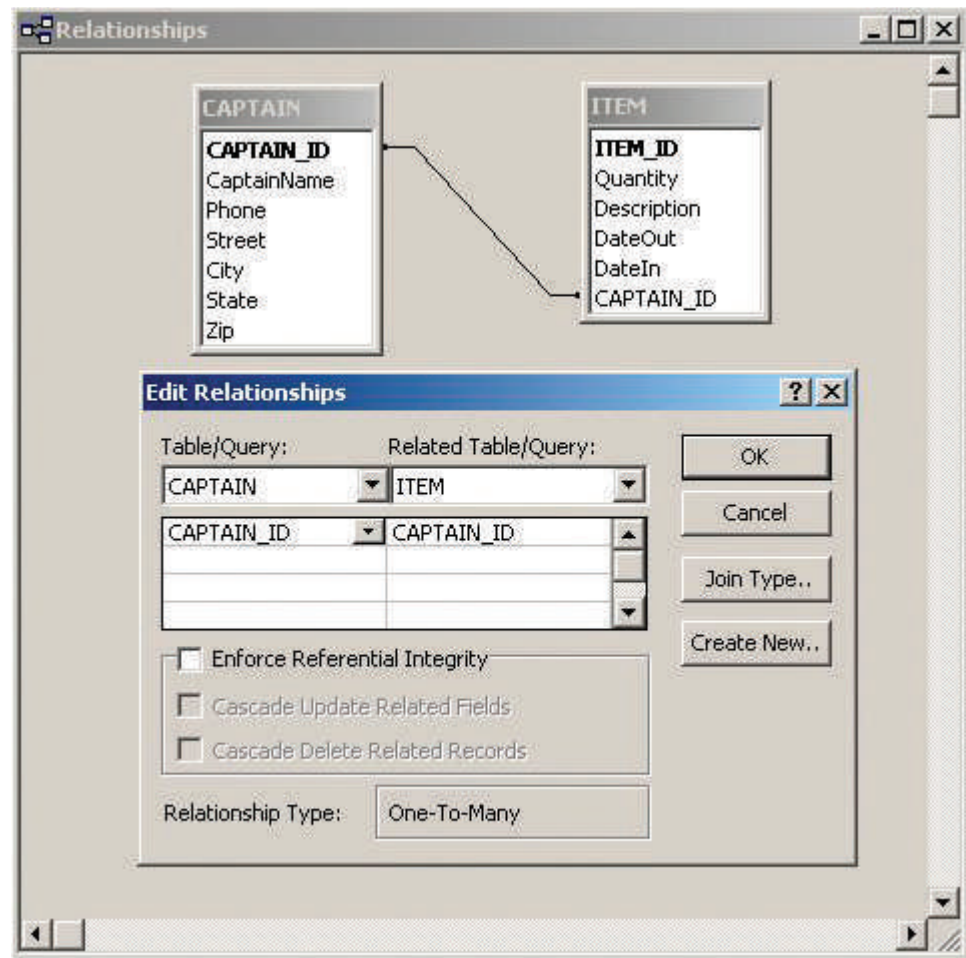
General | Lookup

Field Size: Long Integer
New Values: Increment
Format:
Caption:
Indexed: Yes (No Duplicates)

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

► FIGURA 2-5

Declaración de una
relación con
Microsoft Access
2002



Relationships

CAPTAIN

CAPTAIN_ID

CaptainName
Phone
Street
City
State
Zip

ITEM

ITEM_ID

Quantity
Description
DateOut
DateIn
CAPTAIN_ID

Edit Relationships

Table/Query: CAPTAIN Related Table/Query: ITEM

CAPTAIN_ID CAPTAIN_ID

☐ Enforce Referential Integrity
☐ Cascade Update Related Fields
☐ Cascade Delete Related Records

Relationship Type: One-To-Many

OK
Cancel
Join Type..
Create New..

porcionar más servicios y ayudar al analista si tiene conocimiento sobre qué CAPTAIN_ID en la tabla ITEM es una llave externa de CAPTAIN.

Los productos DBMS varían en la manera en que declaran esta condición. Con Microsoft Access la declaración se hace al establecer la relación entre la llave y la llave externa, como se muestra en la figura 2-5. CAPTAIN_ID de la tabla principal (CAPTAIN) está programada para igualarse con CAPTAIN_ID de la tabla relacionada (ITEM).

Una de las ventajas de establecer una relación al DBMS es que siempre que se coloquen las columnas de dos tablas en una forma, consulta, o reporte, el DBMS sabrá cómo relacionar los renglones de las tablas. Aunque esto puede establecerse cada vez para cada forma, consulta o reporte, hacerlo una vez ahorra tiempo y reduce la posibilidad de cometer errores. Por ahora, ignore otros elementos en la ventana de Edición de Relaciones (Edit Relationships). Usted aprenderá acerca de ellos a medida que avancemos. Una vez que se hayan definido las tablas, columnas y relaciones, el siguiente paso será construir los componentes de las aplicaciones.

► COMPONENTES DE APLICACIONES

Una aplicación de base de datos consta de formas, consultas, reportes, menús y programas de aplicación. Como se muestra en la figura 2-1, las formas, consultas y reportes pueden ser definidos utilizando las herramientas que trae el DBMS. Los programas de aplicación deben escribirse ya sea en un lenguaje que sea parte del DBMS, o en uno estandarizado y conectado a la base de datos mediante el DBMS.

FORMAS

La figura 2-6 muestra tres presentaciones diferentes de los datos en las tablas CAPTAIN y ITEM. En la figura 2-6(a) se muestran los datos en una hoja de cálculo o formato tabular. El usuario puede hacer “clic” en el signo que está al inicio de cada renglón para desplegar los registros de ITEM que se relacionan con un renglón específico de CAPTAIN. Esto se ha realizado para los datos de Abernathy en el segundo renglón de esta figura. Observe que dos de los renglones de ITEM se relacionaron con Abernathy.

La figura 2-6(b) ilustra una segunda presentación mediante una forma de datos de entrada, la cual muestra los datos para un solo capitán a la vez. A los usuarios sin experiencia les parecerá más fácil de usar que el formato tabular.

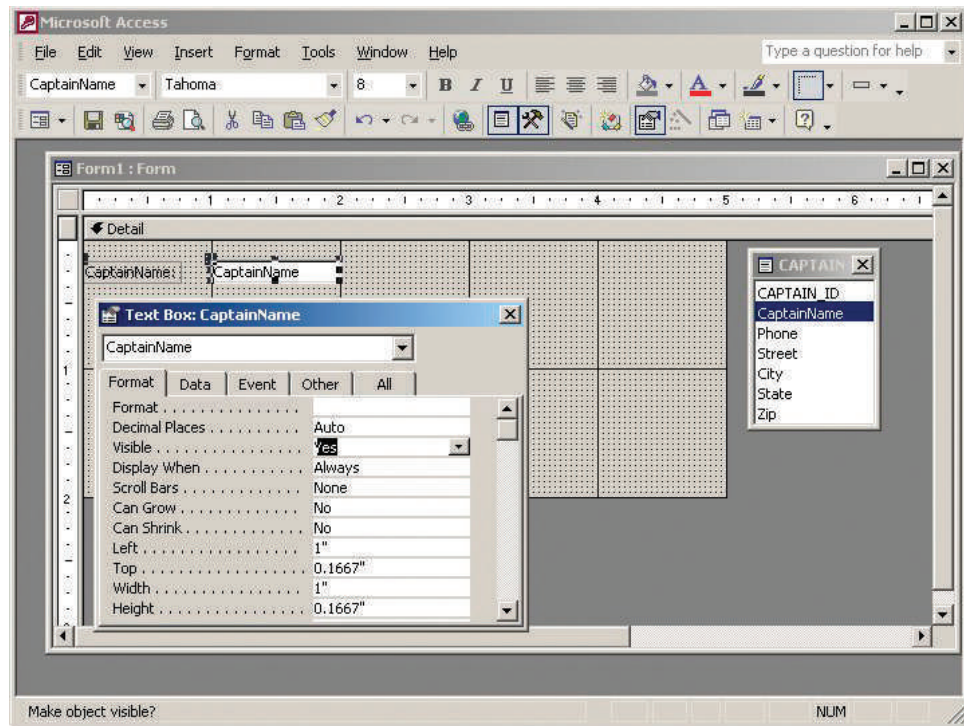
La página Captain Registration (Registro de Capitán) que se muestra en la figura 2-6(c) se puede utilizar en Internet o en la intranet de la universidad e ingresarla por medio del Internet Explorer de Microsoft. Tal uso requerirá que esté almacenada en un servidor Web, tal como el Servidor de Información de Internet. Aprenderá más sobre esto en los capítulos del 14 al 16. Por ahora, sólo observe que la herramienta de formas de Access 2002 puede utilizarse para crearlas.

Access 2002 genera automáticamente la forma tabular para cada tabla definida en la base de datos. Sin embargo, las formas de entrada de los datos se deben crear utilizando las herramientas para generarlas. La figura 2-7 muestra una manera de crearla. La fuente de datos de dicha forma ha sido programada para la tabla CAPTAIN (que no se muestra en la figura 2-7). Access muestra una ventana denominada *field list* (*lista de campo*), la cual despliega las columnas de la tabla CAPTAIN. En esta figura, el usuario ha arrastrado el nombre del capitán de la lista de campos en la forma. Como respuesta, Access crea una etiqueta de control para CaptainName y un control del cuadro de texto que se utilizará para ingresar y desplegar los valores para CaptainName. En este punto, se dice que el cuadro de texto está *limitado a* la columna CaptainName de la tabla CAPTAIN. Otras columnas están limitadas de la misma manera; las de la tabla ITEM se limitan a la forma mediante un dispositivo llamado subforma. Access también tiene una herramienta experta para el diseño de formas, que se puede utilizar para crear algunas como la de la figura 2-6(b).

Ninguna de estas formas despliega las columnas CAPTAIN_ID o ITEM_ID. Dichos ID han estado escondidos del usuario. Sin embargo, tras bambalinas, el DBMS automá-

► FIGURA 2-7

Creación de una forma con Microsoft Access 2002



de nuevo la figura 2-4. La representación de los datos de ITEM_ID ha sido puesta en un AutoNumber (AutoNúmero). Esto instruye a Access para que asigne valores a ITEM_ID cuando se crean nuevos renglones. Cuando se creó la tabla CAPTAIN (la cual no se muestra), se estableció una programación similar a CAPTAIN_ID. Sin embargo, observe que para CAPTAIN_ID en la tabla ITEM no se estableció un AutoNumber. Esto se debe a que el valor de CAPTAIN_ID se creó al mismo tiempo que el renglón CAPTAIN; así que ese valor se copia en el campo CAPTAIN_ID en la tabla ITEM cuando se conecta un renglón ITEM a un capitán en particular.

¿Por qué se ocultan estos ID? La razón es que no tienen ningún significado para los usuarios. El Colegio Highline no asigna estos ID a los capitanes o a determinados artículos que son prestados. Si fuera así, entonces los ID se utilizarían y serían visibles. En su lugar, éstos han sido creados sólo para que cada renglón de cada tabla sea únicamente identificable para Access. Puesto que estos ID no significan nada para el usuario, están ocultos. Los identificadores como éste se llaman **llaves sustitutas**.

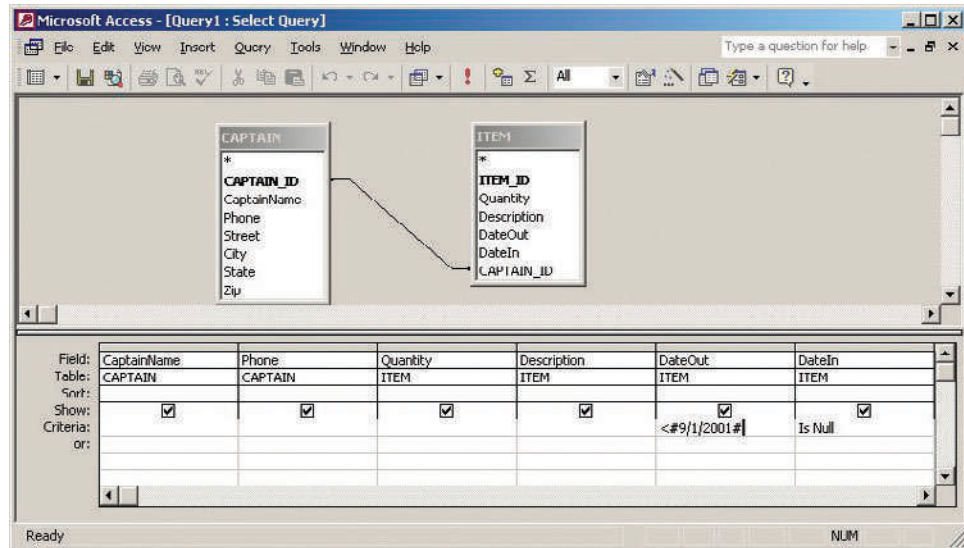
CONSULTAS

De vez en cuando los usuarios desean consultar los datos para responder consultas o identificar determinados problemas o situaciones. Por ejemplo, suponga que al inicio del semestre de otoño del 2001, uno de los usuarios desea saber si algún equipo que fue revisado antes del primero de septiembre del 2001, no se ha vuelto a revisar. De ser así, deseará conocer el nombre del capitán, su número telefónico y la cantidad y descripción de los artículos que tiene prestados.

Existen un sinnúmero de maneras para expresar dicha consulta. Una es utilizar el lenguaje de acceso a datos SQL, el cual se explica en el capítulo 9; otra es usar la **consulta mediante ejemplo (query by example, QBE)**. La figura 2-8 muestra la creación de esta consulta por medio de QBE de Microsoft Access. Para crearla, el usuario coloca en la ventana de consulta los nombres de las tablas que serán consultadas. Esto ya se ha hecho en la sección anterior de la forma en que se muestra en la figura 2-8. Puesto que la relación entre CAPTAIN e ITEM ya ha sido definida para Access (en la figura 2-5), Access sabe que las dos tablas están unidas por CAPTAIN_ID, como lo muestra la línea dibujada entre CAPTAIN_ID en los dos cuadros de texto en la figura 2-8.

FIGURA 2-8

Creación de una consulta con Microsoft Access 2002



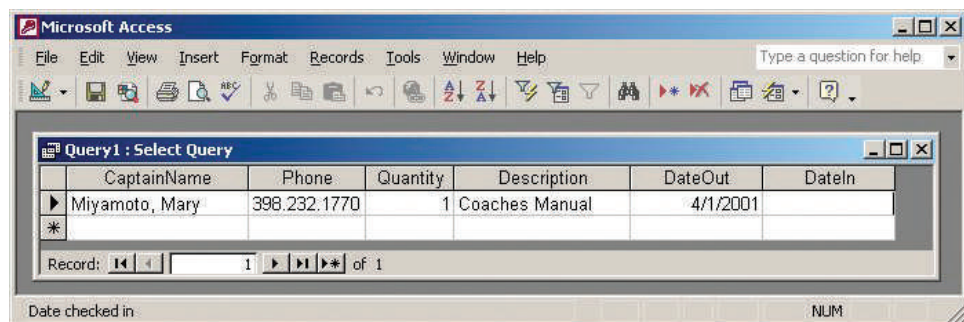
Posteriormente, el creador de la consulta indica cuáles columnas de datos regresarán mediante la consulta. Con Access, esto se realiza cuando se arrastran los nombres de las columnas de los cuadros de texto en la cuadrícula y se posicionan en la entrada principal inferior de la forma. En la figura 2-8 las columnas CaptainName, Phone, Quantity, Description, DateOut, y DateIn se han colocado en la consulta. Los criterios de la consulta se especifican en el renglón llamado Criterios los cuales son los datos que deben estar antes de (<) 9/1/2001. (En Access, los datos están entre signos de número [#].) Asimismo, el valor de DateIn será nulo, lo que significa que no se ha especificado ningún valor para DateIn. El resultado de esta consulta sobre ejemplos de datos se muestra en la figura 2-9. Observe que todo el equipo que se muestra se prestó antes del 9/1/2001, como quedó establecido en la definición de consulta.

Con Access, y con la mayoría de los productos DBMS, las consultas pueden guardarse como parte de la aplicación, de manera que puedan regresarse cuando sea necesario. Además, se pueden parametrizar, lo cual significa que pueden construirse de manera que se acepten los valores de los criterios al mismo tiempo en que se ejecutan. Por ejemplo, la consulta en la figura 2-8 puede parametrizarse de modo que el usuario ingrese el valor de DateOut cuando la consulta se ejecute. Aparecerán cualesquiera de los artículos que fueron prestados antes de dicha fecha, pero que no se han vuelto a prestar.

Un tercer tipo de consulta, que es más fácil para los usuarios, se denomina **consulta por forma (query by form, QBF)**. Con ésta, el usuario escribe las restricciones para la consulta en una forma de entrada de datos y presiona el botón de búsqueda. El DBMS encuentra todas las instancias que concuerdan con las restricciones específicas. Para la consulta del ejemplo, en el formato de la figura 2-6, el usuario ingresaría <#9/1/2001# en el campo DateOut e Is Null en el campo DateIn, y presionaría el botón de Query_by_Form. El DBMS encontraría los registros de todos los capitanes que coincidieran con las restricciones de la consulta. Query by form es una manera más

FIGURA 2-9

Ejemplo del resultado de la consulta de la figura 2-8



► FIGURA 2-10

Ejemplo del reporte
del Colegio Highline

Captain Equipment Report

CaptainName	Miyamoto, Mary			
Phone	398.232.1770			
DateOut	Quantity	Description	DateIn	
4/1/2001	1	Coaches Manual		
4/1/2001	7	Soccer Balls	6/10/2001	
4/10/2001	25	Blue Soccer Shirts	6/10/2001	

CaptainName	Abernathy, Mary Jayne			
Phone	223.768.3378			
DateOut	Quantity	Description	DateIn	
3/22/2001	5	Soccer Balls	6/3/2001	
3/22/2001	14	Soccer Shirts	6/3/2001	

nueva y moderna en comparación con la consulta por ejemplo, y es probable que la reemplace en muchas aplicaciones.

REPORTES

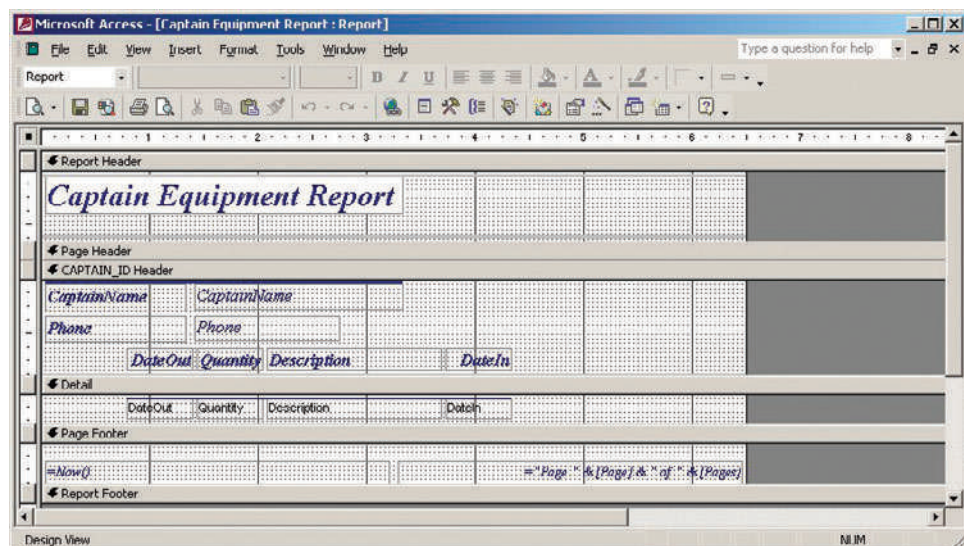
Un reporte es una representación con formato de la información que se encuentra en una base de datos. El reporte de la figura 2-10 contiene una sección para cada capitán de equipo y una lista de los artículos que se les han prestado a cada uno. Por ejemplo, los artículos a cargo del capitán “Miyamoto Mary” se muestran en la primera sección del reporte.

El desarrollo de un reporte es similar a preparar una forma de entrada de datos, aunque en ocasiones es más fácil debido a que un reporte sólo se utiliza para mostrar datos. Otras veces, la elaboración de un reporte es más difícil debido a que frecuentemente su estructura es más complicada que la de las formas.

La figura 2-11 muestra la definición de los reportes de Access 2002 para el reporte que se incluye en la figura 2-10. Éste es un ejemplo de un **escritor de reportes agrupados**; se llama así debido a que el reporte se divide en secciones o grupos. Como se

► FIGURA 2-11

Desarrollo de un
reporte con Microsoft
Access 2002



muestra, existen grupos superiores, un grupo de detalle y grupos inferiores. El grupo superior muestra el nombre del reporte y el grupo inferior, la fecha en que se imprimió, el número de página y el número total de páginas. El encabezado de la página está vacío, pero el encabezado de CAPTAIN_ID tiene los datos y las etiquetas para la sección de detalles, la cual muestra los artículos prestados a un capitán.

Hay un problema cuando los dos campos de datos de este reporte están por separado. Para que sea eficaz, es mejor colocar DateOut entre Description y DateIn. Esto se puede hacer rápidamente arrastrando y posicionando las etiquetas y las cajas de texto, y es representativo del tipo de cambios que se realizan cuando se usan estas herramientas.

En general, los reportes pueden tener muchas secciones. En un ejemplo más complicado, es decir, matrícula curso-alumno, un reporte puede ser agrupado por COLEGIO, DEPARTAMENTO, CURSO y ESTUDIANTE. En este caso, el reporte tendría tres secciones de encabezados y una línea de detalles.

MENÚES

Los menús se utilizan para organizar componentes de la aplicación, de tal manera que sean más accesibles para el usuario final y proporcionen control sobre las actividades de éste. La figura 2-12 muestra un menú de ejemplo para la aplicación de Highline. La línea en la parte superior de la forma presenta las opciones de más alto nivel: File, Forms, Queries, Reports y Help (Archivo, Formas, Consultas, Reportes y Ayuda). Las letras subrayadas representan las teclas de acceso rápido. Si el usuario oprime <Alt> y la letra subrayada, se desplegará el submenú de opciones.

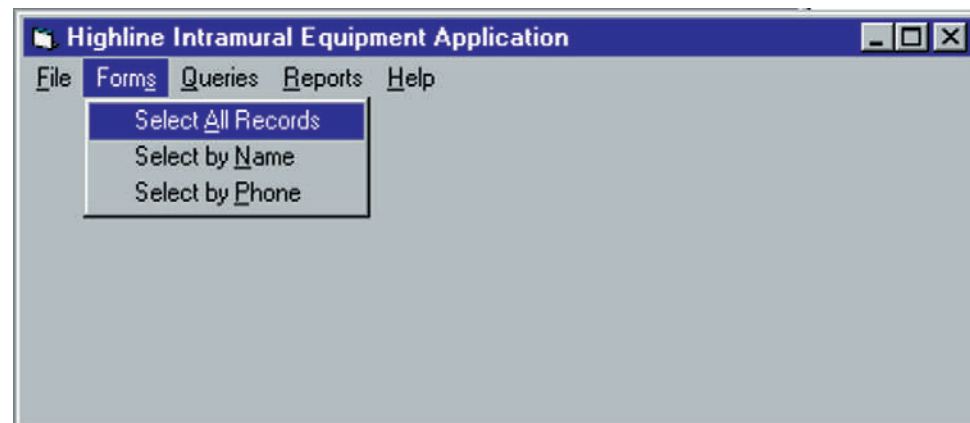
En la figura 2-12 el usuario ha tecleado <Alt> más la letra S. Las opciones del submenú son Select All Records, Select by Name y Select by Phone. De nuevo, las teclas de acceso rápido pueden utilizarse para seleccionar una de estas partes al teclear (Alt) y la letra subrayada de la opción. Los menús hacen que la aplicación sea más accesible para el usuario mostrando cuáles son las opciones disponibles y ayudándole a seleccionar las que quiere realizar. Asimismo, se pueden utilizar los menús para controlar el acceso del usuario a formas, reportes y programas. Algunas aplicaciones sacan ventaja de esto al cambiar dinámicamente las partes del menú después de que el usuario ingresa.

PROGRAMAS DE APLICACIÓN

El componente final de una aplicación de una base de datos radica en los programas de aplicación. Como se mencionó anteriormente, dichos programas pueden escribirse en un lenguaje específico para el DBMS o en un lenguaje estándar que se interconecta con

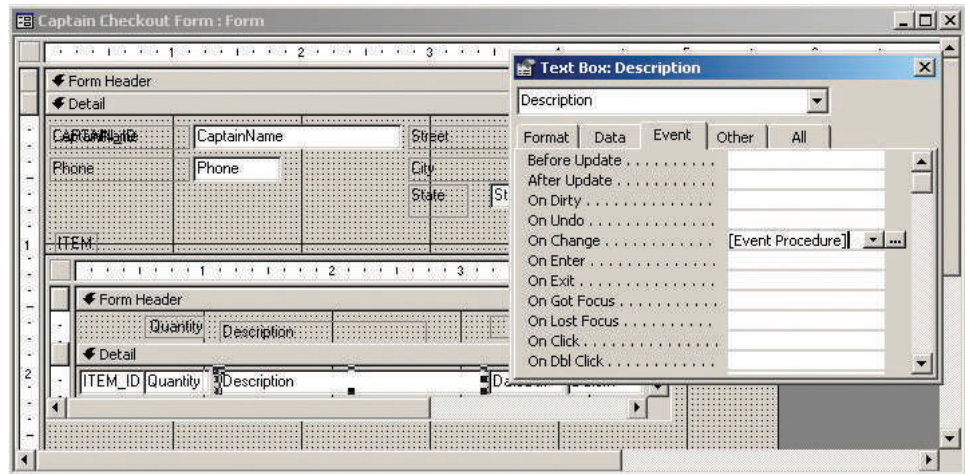
► FIGURA 2-12

Ejemplo de menú



► FIGURA 2-13

Captación de un evento en la forma de captura



el DBMS a través de un programa de interfaz predefinido. Aquí se utilizarán Microsoft Visual Basic y Access 2002.

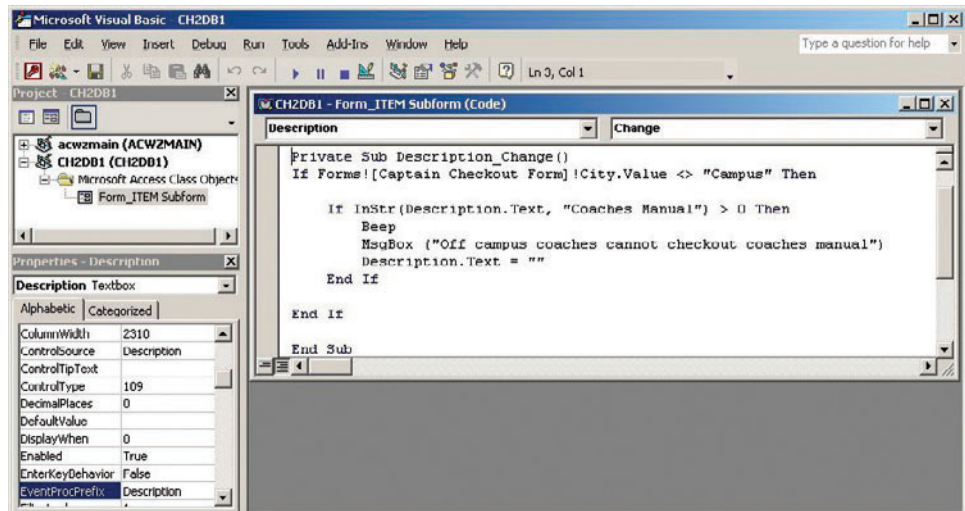
Suponga que Highline tiene una regla de negocio en la que los capitanes que viven fuera del campus no pueden tomar prestado el manual del entrenador. No se sabe la razón de esta regla; probablemente se deba a que contiene instrucciones secretas que proporcionan a los equipos de Highline una ventaja competitiva, y el colegio Highline está tratando de controlar la distribución de los manuales. Quizá son caros y si se pierden los estudiantes que viven en el campus tendrán que pagarlos. Cualquiera que sea la razón, Highline desea la aplicación de una base de datos para hacer que esta regla se cumpla.

Existen muchas maneras de hacerlo. Una de ellas, que se muestra en la figura 2-13, se refiere a captar el cambio en el cuadro Text Box: Description en la subforma de ITEM de Captain Checkout Form, e insertar un código para poner en vigor la regla cada vez que Description cambie. En la figura 2-13 el analista ha abierto las propiedades para el cuadro de texto de Description, y ha establecido On Change en [Event Procedure]. Cuando esto se llevó a cabo, Access abrió la ventana codificada que se muestra en la figura 2-14.

El analista escribió una sección de Visual Basic que se ejecutará cuando se presente el cambio. Primero, el código determina si el valor de City (Ciudad) no es igual a la condición "Campus". Si es así, entonces se requerirá la función InStr para determinar si los contenidos del cuadro de texto (Description.Text) contienen la condición de "Coaches Manual" (Manual del Entrenador). Si es así, se escucha un sonido, se genera un cuadro de mensaje y el cuadro de texto de Description se pone en blanco.

► FIGURA 2-14

Código de Visual Basic para poner en vigor la regla del negocio



Este código se ejecuta cada vez que el usuario cambia el valor del cuadro de texto de Description. Existen otros medios más eficaces de imponer esta regla, pero el punto aquí es sólo mostrarle cómo se puede integrar el código de aplicación con las formas de la base de datos.

Por supuesto, también es posible escribir un código que se ejecute independientemente de cualquier forma o reporte. Los programas pueden ser escritos para lectura y actualización de los datos de la base, de la misma manera en que pueden leer y actualizar otro tipo de archivos. Verá ejemplos de esto más adelante, especialmente en el capítulo 15 donde se muestra el uso de Microsoft ADO para leer y actualizar los datos de un Servidor Web, así como en el capítulo 16 en el que se utilizará Java con el mismo propósito.

► PROCESOS PARA EL DESARROLLO DE UNA BASE DE DATOS

Se han escrito volúmenes y volúmenes sobre el desarrollo de los sistemas de información en general, y de manera particular con respecto al desarrollo de las aplicaciones de una base de datos. Por consiguiente, no es necesario analizar a profundidad los procesos del desarrollo de sistemas; sin embargo, concluiremos este capítulo con un panorama de los procesos que se utilizan para desarrollar bases de datos y aplicaciones de éstas.

ESTRATEGIAS GENERALES

Una base de datos es un modelo del modelo de los usuarios referente a las actividades de su negocio. Por lo tanto, con el fin de construir una base de datos eficaz, así como las aplicaciones relacionadas, el equipo analista debe entender plenamente el modelo de los usuarios. Para hacerlo, el equipo construye un modelo de datos que identifica las cosas que se almacenarán en la base de datos y define su estructura y las relaciones entre éstas. Tal entendimiento debe existir antes que se realice el proceso de desarrollo, mediante entrevistas con los usuarios y la construcción de un documento de requerimientos. La mayoría de este tipo de esquemas incluyen el uso de **prototipos**, que son bases de datos y aplicaciones de ejemplos que representan diversos aspectos del sistema que será creado.

Existen dos estrategias generales para desarrollar una base de datos: arriba-abajo y abajo-arriba. El **desarrollo de lo general a lo particular** deriva de lo general a lo específico. Comienza con el estudio de las metas estratégicas de la organización, los medios por los cuales se pueden lograr estas metas, así como los requerimientos de información que se deben satisfacer para alcanzar las metas, y los sistemas que se requieren para proporcionar dicha información. A partir de tal estudio se construye un modelo de datos abstractos.

Mediante este modelo de alto nivel, el equipo analista trabaja progresivamente descendiendo hacia descripciones y modelos más y más detallados. Los modelos de nivel intermedio se expanden también con mayor detalle hasta que las bases de datos particulares y las aplicaciones pueden ser identificadas. Se selecciona entonces una o más de estas aplicaciones para el desarrollo. Con el tiempo, todo el modelo de datos de alto nivel se transforma en modelos de nivel inferior y se crean todos los sistemas indicados, las bases de datos y las aplicaciones.

El **desarrollo de lo particular a lo general** opera en el orden contrario de la abstracción al comenzar con la necesidad de desarrollar un sistema específico. Los medios de selección del primer sistema varían de una empresa a otra. En algunas, el comité directivo elige la aplicación; en otras, los usuarios son quienes la seleccionan; en cambio en otras, la voz con mayor autoridad en la oficina ejecutiva es la que tiene el poder de decisión.

Por cualquier medio, se selecciona un sistema específico para el desarrollo. El equipo analista obtiene entonces las declaraciones y los requerimientos al considerar las salidas y entradas de cualquier sistema con una base computacional, mediante el análisis de las formas y los reportes para los sistemas manuales, así como mediante la entrevista-

ta con los usuarios para descubrir sus necesidades respecto a nuevos reportes, formas, consultas y otros requerimientos. A partir de todo esto, el equipo desarrolla el sistema de información. Si éste involucra una base de datos, el equipo utiliza las especificaciones de los requerimientos para construir un modelo de datos, y a partir de éste diseña e implementa la base de datos. Cuando se concluye este sistema, comienzan otros proyectos con el fin de construir sistemas de información adicionales.

Los partidarios del enfoque de lo general a lo particular afirman que es superior al enfoque que va de lo particular a lo general, debido a que los modelos de datos (y sistemas subsecuentes) se construyen con una perspectiva global. Aseguran que dichos sistemas poseen mejores interfaces que ningún otro, que son más consistentes y requieren mucho menos repetición de trabajo y modificaciones.

Los partidarios del enfoque de lo particular a lo general sostienen que es superior al enfoque de lo general a lo particular debido a que es más rápido y menos riesgoso. Afirman que el modelo de lo general a lo particular da como resultado muchos estudios que son difíciles de completar y que el proceso de planeación a menudo resulta en la obstrucción del análisis. Aunque el modelo de lo particular a lo general no necesariamente da como resultado el mejor conjunto de sistemas, sí produce rápidamente sistemas útiles. Los beneficios de éstos comienzan a ser más rápidos que con el modelo de lo general a lo particular y pueden compensar mucho más en cualquier repetición o modificación que se necesite realizar para ajustar el sistema a una perspectiva global.

Este texto explica las herramientas y técnicas que se pueden utilizar con cualquier estilo de desarrollo de sistemas. Por ejemplo, aunque el modelo entidad-relación (capítulo 3) y el modelo de objeto semántico (capítulo 4) trabajan tanto con el desarrollo de lo general a lo particular como con el de lo particular a lo general, el enfoque entidad-relación es particularmente eficaz con el desarrollo de lo general a lo particular, y el enfoque de objeto semántico es especialmente eficaz con el desarrollo de lo particular a lo general.

MODELACIÓN DE DATOS

Como ya establecimos, la meta más importante de la fase de requerimientos es la creación de un modelo de datos de los usuarios. Ya sea que se realice en el estilo de lo general a lo particular o en el de lo particular a lo general, involucra la entrevista con los usuarios, documentación de los requerimientos y, a partir de éstos, la construcción del modelo de datos y prototipos. Dicho modelo identifica lo que se almacenará en la base de datos y define su estructura e interrelación.

Por ejemplo, considere la figura 2-15(a) en la que se muestra una lista de órdenes que hizo un vendedor durante un periodo específico. Para que este reporte se produzca mediante la aplicación de una base de datos, ésta debe contener los datos mostrados; por ende, los analistas de la base de datos necesitan examinar el reporte y trabajar en los datos que deben almacenarse en la base. En este caso, deben existir datos con respecto a los vendedores (nombre y región) y acerca de los pedidos u órdenes (empresa, fecha del pedido y cantidad).

El desarrollo de una base de datos es complicado por el hecho de que no sólo existe un requerimiento sino muchos, los cuales a menudo se traslapan. El reporte en la figura 2-15(b) se refiere también a vendedores, pero, en lugar de pedidos, emite cheques de comisión. A partir de este reporte se puede suponer que existen diferentes tipos de pedidos y que cada uno tiene una tasa de comisión diferente.

Los pedidos que están en el reporte de la figura 2-15(b) se relacionan de alguna manera con los de la figura 2-15(a), pero el modo en que lo hacen no es del todo claro. El equipo analista debe determinar esta relación mediante la inducción de reportes y formas, entrevistas con los usuarios, conocimiento del equipo en cuestión, así como otras fuentes.

MODELACIÓN DE DATOS BASADA EN REFERENCIAS. Cuando los usuarios dicen que necesitan formas y reportes con datos y estructuras específicas, su petición implica un modelo basado en las cosas que forman parte de su mundo. Sin embargo, no pueden describir exactamente qué modelo es ése. Si un analista le preguntara a un

► FIGURA 2-15

Ejemplos de dos reportes relacionados: (a) ejemplo del reporte de VENTAS y (b) ejemplo del reporte de COMISIONES

Lista de pedidos del vendedor				
03-Oct-2001				
Nombre	Región	NombredelaCompañía	FechadelPedido	Cantidad
Kevin Dougherty	Oeste	Cabo Controls	9/12/2001	\$2,349.88
				\$2,349.88
Mary B. Wu	Oeste	Ajax Electric	9/17/2001	\$23,445.00
		American Maxell	9/24/2001	\$17,339.44
				\$40,784.44
Gran total:				\$43,134.32

(a)

Reporte del cheque de comisión del vendedor				
03-Oct-2001				
Nombre	NúmeroLocal	Fecha del Cheque	TipodeCheque	Cantidad del Cheque
Kevin Dougherty	232-9988	9/30/2001	XZ	\$487.38
				\$487.38
Mary B. Wu	232-9987	9/30/2001	C	\$237.44
		9/30/2001	A	\$1,785.39
				\$2,022.83
				\$2,510.21
Gran total:				\$2,510.21

(b)

usuario común y corriente: “¿Cuál es su estructura mental del modelo de datos con respecto a los vendedores?”, el usuario, en el mejor de los casos, se sorprendería y estaría alerta; esto se debe a que muchos usuarios no piensan de esta manera.

En lugar de lo anterior, los analistas deben inferir, a partir de las declaraciones de los usuarios sobre las formas y los reportes, la estructura y las relaciones de las cosas que se almacenarán en la base de datos. Deberán entonces registrar esas inferencias en un modelo de datos que se transforme en el diseño de una base de datos, y dicho diseño se implementará por medio de un DBMS. Así es como se construyen las aplicaciones que producen reportes y formas para los usuarios.

La construcción de un modelo de datos es un proceso por inferencia. Los reportes y las formas son como las sombras: se proyectan en un muro. Los usuarios pueden describir las sombras, pero no las formas que dan origen a esas sombras. De ahí que los analistas deban deducir, trabajar en diferentes sentidos, revertir o replantear la ingeniería de las estructuras y de las relaciones de dichas formas, a partir de las sombras.

Por desgracia, este proceso por inferencia es más arte que ciencia. Es posible aprenderse las herramientas y las técnicas para la modelación de datos; de hecho dichas herra-

mientas y técnicas son el tema de los dos siguientes capítulos, pero el uso de éstas es un arte que requiere de la experiencia guiada por la intuición.

La calidad del modelo es importante. Si el modelo de datos documentados refleja exactamente el modelo de datos que hay en la mente de los usuarios, existe una excelente oportunidad de que las aplicaciones que resulten se acerquen más a las necesidades de los usuarios. Pero si el modelo de datos documentados refleja incorrectamente dicho modelo, es poco probable que la aplicación se acerque a lo que ellos realmente quieren.

MODELADO EN SISTEMAS MULTIUSUARIO. El proceso de modelación de datos se vuelve mucho más complejo en el caso de un grupo de trabajo multiusuarios y bases de datos organizacionales, debido a que muchos usuarios pueden imaginarse diferentes modelos de datos. Ocasionalmente éstos son inconsistentes, aunque la mayoría de las veces dichas inconsistencias pueden solucionarse. Por ejemplo, los usuarios pueden emplear el mismo término para diferentes cosas, o diferentes términos para las mismas cosas.

Sin embargo, a veces las diferencias no pueden ser conciliadas. En esos casos, el analista de la base de datos debe documentar las diferencias y ayudar a los usuarios a resolverlas, y esto por lo general significa que ciertas personas deben cambiar la manera en la cual ven al mundo que les rodea.

Un reto aún mayor está en los grandes sistemas, en los que ningún usuario tiene un modelo de la estructura completa. Cada usuario comprende parte del modelo de datos del grupo de trabajo o de la organización, pero ninguno lo entiende completamente. En tales casos, la base de datos se convierte en la unión lógica de las piezas del modelo del grupo de trabajo o de la organización, y los analistas deben documentar esa unión lógica en el modelo de datos. Lo anterior puede ser muy difícil.

CONFUSIÓN ACERCA DEL TÉRMINO *MODELO*. Los capítulos 3 y 4 presentan dos herramientas alternativas para la construcción de modelos de datos: el de entidad-relación y el modelo de objeto semántico. Ambos son estructuras que describen y documentan los requerimientos de los datos de los usuarios. Para evitar confusión, observe los diferentes usos del término *modelo*. El equipo analiza los requerimientos y construye un *modelo de los datos de los usuarios*, o un *modelo de los datos de los requerimientos*. Este modelo es una representación de la estructura y de las relaciones de lo que se necesita en la base de datos para sustentar los requerimientos de los usuarios. Con el fin de crear el modelo de datos de los usuarios, el equipo analista utiliza herramientas denominadas modelos de datos entidad-relación y de objeto semántico, las cuales consisten en estándares de lenguaje y diagramas para representar el modelo de los datos de los usuarios. Su papel en el desarrollo de la base de datos es similar al del diagrama de flujos y del pseudocódigo en la programación.

► RESUMEN

Los componentes de un sistema de una base de datos son la propia base de datos, el DBMS y los programas de aplicación, los cuales los utilizan analistas y usuarios. Una base de datos contiene datos, metadatos, índices y metadatos de aplicación. Actualmente la mayoría de las bases representan a los datos como relaciones o tablas, aunque no todas las relaciones son igualmente deseables. Las relaciones indeseables pueden mejorarse a través de un proceso denominado normalización. Los metadatos a menudo son almacenados en tablas llamadas tablas de sistema.

Las funciones y características de un DBMS pueden agruparse en tres subsistemas. El subsistema de las herramientas de diseño define la base de datos y la estructura de las aplicaciones, o de los componentes de éstas. Las funciones del subsistema run-time son la materialización de las formas, reportes y consultas mediante la lectura y escritura de los datos de una base. El motor DBMS es el intermediario entre los otros dos subsistemas y el sistema operativo. Recibe requerimientos establecidos en términos de tablas, renglones y columnas y los traduce en lectura y escritura.

Un esquema es una descripción de la estructura de una base de datos e incluye descripciones de tablas, relaciones, dominios y reglas del negocio. Los renglones de una tabla pueden relacionarse con los de otras. Este capítulo ilustró una relación 1:N entre los renglones de las tablas; asimismo, existen otros tipos de relaciones, como veremos en el siguiente capítulo.

Un dominio es un conjunto de los valores que una columna puede tener. Se debe especificar un dominio para cada columna de cada tabla.

Por último, las reglas del negocio son restricciones de las actividades del negocio que deben reflejarse en la base de datos y en las aplicaciones de ésta.

Las facilidades del DBMS se utilizan para crear estructuras de tablas, definir relaciones y crear formas, consultas, reportes y menús. Los productos DBMS también incluyen dispositivos para interactuar con programas de aplicación por escrito, ya sea en lenguajes específicos del DBMS o en lenguajes estándares como Java.

Puesto que la base de datos es un modelo del negocio de los usuarios, el desarrollo de una base de datos empieza con el aprendizaje y registro de dicho modelo. Algunas veces esto se expresa en prototipos de la aplicación, o en los componentes de la aplicación que será construida.

Los dos estilos generales de desarrollo son de lo general a lo particular, que deriva de lo general a lo específico, y el desarrollo de lo particular a lo general, que procede de lo específico a lo general. Con el desarrollo de lo general a lo particular las aplicaciones se llevan a cabo desde una perspectiva global; con el de lo particular a lo general, las aplicaciones se desarrollan más rápido. Algunas veces se utiliza una combinación de ambos enfoques.

Los modelos de datos se construyen mediante un proceso que implica la inferencia a partir de las declaraciones de los usuarios. Se reúnen las formas, los reportes y las consultas, y los analistas trabajan en dirección inversa para deducir las estructuras que los usuarios se imaginan. Esto es necesario debido a que la mayoría de los usuarios no pueden describir exactamente sus modelos de datos. La modelación de datos puede ser especialmente difícil y desafiante en las aplicaciones multiusuarios, en las cuales los propósitos de los usuarios pueden contraponerse y por lo tanto ninguno puede visualizar el propósito integral de la actividad del negocio.

El término *modelo de datos* se utiliza de dos maneras: para referirse a un modelo del propósito de los usuarios con respecto a sus datos, y para denotar las herramientas que se utilizan con el fin de definir el propósito de los usuarios en cuanto a sus datos.

► PREGUNTAS DEL GRUPO I

- 2.1 Nombre los componentes principales del sistema de una base de datos y explique brevemente la función de cada uno.
- 2.2 Dé un ejemplo, aparte del ya mencionado en este capítulo, sobre una relación que tienda a presentar problemas cuando se actualiza. Utilice la relación R1 como base, referencia.
- 2.3 Transforme la relación de su respuesta en la pregunta 2.2 en dos o más relaciones que no tengan problemas de actualización. Utilice las relaciones R2 y R3 como ejemplos.
- 2.4 Explique las funciones de los metadatos y de las tablas de sistema.
- 2.5 ¿Cuál es la función de los índices? ¿En qué momento son deseables y cuál es su costo?
- 2.6 ¿Cuál es la función de los metadatos de aplicación? ¿Cómo difieren de los metadatos?
- 2.7 Explique las características y funciones del subsistema de las herramientas de diseño de un DBMS.
- 2.8 Describa las características y funciones del subsistema run-time de un DBMS.

- 2.9 Explique las características y funciones del motor DBMS.
- 2.10 ¿De qué consta el esquema de una base de datos? Enumere sus componentes.
- 2.11 ¿Cómo se representan las relaciones en el diseño de una base de datos relacional? Proporcione un ejemplo sobre dos tablas con una relación 1:N y explique cómo se expresa la relación en los datos.
- 2.12 ¿Qué es un dominio y por qué es importante?
- 2.13 ¿Qué son las reglas del negocio? Dé un ejemplo sobre posibles reglas del negocio para las relaciones en su respuesta a la pregunta 2.11.
- 2.14 ¿Qué es una llave externa? ¿Cuáles columnas en su respuesta a la pregunta 2.11 representan una llave externa?
- 2.15 Explique el propósito de las formas, consultas y menús.
- 2.16 Explique la diferencia entre la consulta mediante ejemplo y la consulta por forma (QBF).
- 2.17 ¿Cuál es la primera tarea importante en el desarrollo de una base de datos y aplicaciones relacionadas?
- 2.18 ¿Cuál es la función de un prototipo?
- 2.19 Describa el desarrollo de lo general a lo particular. ¿Cuáles son sus ventajas y desventajas?
- 2.20 Describa el desarrollo de lo particular a lo general. ¿Cuáles son sus ventajas y desventajas?
- 2.21 Explique los dos significados diferentes del término *modelo de datos*.

► PREGUNTAS DEL GRUPO II

- 2.22 Implemente una base de datos con las relaciones CAPTAIN e ITEM, en cualquier DBMS al que pueda tener acceso. Utilice uno de los productos DBMS para ingresar los datos a cada una de estas relaciones. Cree y procese una consulta para utilizar el dispositivo del DBMS y procesar una consulta que identifique aquellos artículos que fueron revisados antes del 1 de septiembre de 2001, así como los que aún no han sido revisados. Imprima el nombre del capitán, su teléfono y la cantidad y descripción de cualesquiera de dichos artículos.
- 2.23 Entreviste a un analista profesional con respecto a la aplicación de una base de datos, e investigue el proceso que él utiliza para desarrollarla. ¿Ese desarrollo es de lo general a lo particular, de lo particular a lo general, o usa alguna otra estrategia? ¿Cómo construye modelos de datos este analista y con qué herramientas? ¿Cuáles son los problemas principales que por lo general surgen al desarrollar una base de datos?
- 2.24 Considere la declaración: “Una base de datos es un modelo del modelo de la realidad de los usuarios.” ¿En qué difiere de la declaración: “Una base de datos es un modelo de la realidad”? Suponga que dos analistas están en desacuerdo sobre un modelo de datos, y uno de ellos afirma: “Mi modelo es una mejor representación de la realidad.” ¿Qué quiere decir esa persona? ¿Qué diferencias podrían surgir si un analista cree más en la primera declaración que en la segunda?

► PREGUNTAS DEL PROYECTO FIREDUP

Considere la situación de la compañía FiredUp, cuyo caso se presentó al final del capítulo 1. Cada una de las estufas viene con un formato adjunto de registro del producto que incluye los siguientes datos:

NombredelComprador, Dirección, NúmeroDepartamento, Ciudad, Estado/Provincia, CódigoPostal, País, CorreoElectrónico, NúmeroTelefónico, FechadeCompra, y NúmeroSerie

Suponga que FiredUp decide crear una base de datos con las siguientes tablas:

CLIENTE (Nombre, Dirección, NúmeroDepartamento, Ciudad, Estado/Provincia, CódigoPostal, País, CorreoElectrónico y NúmeroTelefónico)

y

COMPRA (FechadeCompra y NúmeroSerie)

A. Construya una tabla de datos a manera de ejemplo que esté basada en la estructura de CLIENTE. Incluya por lo menos cuatro renglones en su tabla. Para las consultas de la A a la G sólo enumere los datos utilizando un procesador de texto.

B. ¿Cuál de las columnas de la tabla CLIENTE puede utilizarse para identificar un renglón único de la tabla? Tal columna a veces se denomina *llave primaria*, como lo aprenderá posteriormente en este texto.

C. Construya una tabla de datos acorde a la estructura de COMPRA. Incluya por lo menos cuatro renglones en su tabla.

D. ¿Cuáles de las columnas de la tabla COMPRA pueden utilizarse como la llave primaria de COMPRA?

E. Cuando se utilizan las tablas anteriormente definidas no hay manera de relacionar a un cliente en particular con su estufa. Una manera de hacerlo sería añadir NúmeroSerie de COMPRA a CLIENTE. La tabla CLIENTE aparecerá entonces como:

CLIENTE (Nombre, Dirección, NúmeroDepartamento, Ciudad, Estado/Provincia, CódigoPostal, País, CorreoElectrónico, NúmeroTelefónico y NúmeroSerie)

Copie los datos de su ejemplo CLIENTE y agréguele la columna NúmeroSerie. Nombre a esta nueva tabla CLIENTE1.

F. Una técnica alternativa para representar la relación de dos tablas sería colocar CorreoElectrónico de CLIENTE en COMPRA. La tabla COMPRA aparecerá entonces como:

COMPRA (FechadeCompra, NúmeroSerie, CorreoElectrónico) Copie su ejemplo Datos de COMPRA y agréguele la columna CorreoElectrónico. Nombre a esta nueva tabla COMPRA1.

G. Ahora tiene tres posibles estructuras de bases de datos.

BD1: CLIENTE1 con COMPRA

BD2: CLIENTE con COMPRA1, y

BD3: CLIENTE1 con COMPRA1

¿En qué circunstancias recomendaría utilizar la estructura BD1?

¿Cuándo recomendaría específicamente el uso de la estructura BD2?

H. ¿En qué casos aconsejaría utilizar la estructura BD3?

MODELACIÓN DE DATOS

La modelación de datos es el proceso de crear una representación lógica de la estructura de una base de datos. Para que sea correcta, la modelación debe considerar todas las perspectivas de los usuarios acerca de los datos. La modelación es la tarea más importante en el desarrollo de las bases y sus aplicaciones. Si la modelación de datos representa erróneamente los puntos de vista del usuario, las aplicaciones le resultarán muy difíciles de usar, incompletas y se sentirá frustrado. La modelación de datos es la base de todo el trabajo posterior del desarrollo de bases de datos y sus aplicaciones.

La parte II describe dos herramientas diferentes de la modelación de datos; el capítulo 3 considera el modelo entidad-relación (E-R), que tiene una gran cantidad de seguidores entre los profesionales de las bases de datos, y en el capítulo 4 se describe el modelo del objeto semántico, que tiene menos adeptos pero se le considera más completo y fácil de usar que el modelo E-R.

Ambos modelos proporcionan un lenguaje que expresa la estructura de los datos y las relaciones en el medio ambiente del usuario. La modelación de datos expresa un diseño lógico de datos de la misma manera en que un diagrama de flujo expresa el diseño lógico de un programa.

El modelo entidad-relación

Este capítulo describe e ilustra el uso del **modelo entidad-relación (modelo E-R)**, que presentó Peter Chen en 1976.¹ En su artículo, Chen estableció los fundamentos del modelo, los cuales a partir de entonces ha ampliado y modificado él mismo, así como muchos otros.^{2,3} Además, el modelo E-R ha formado parte de varias herramientas CASE, que lo han modificado. Actualmente no existe un solo modelo estándar E-R que sea comúnmente aceptado, sino que hay una serie de construcciones comunes de las cuales derivan la mayoría de las variantes E-R. Este capítulo describe estas construcciones comunes y muestra cómo se utilizan. Los símbolos que se emplean para expresar el modelo E-R difieren considerablemente. No sólo analizaremos los símbolos tradicionales, sino también aquellos que se usan en el Lenguaje de modelación unificado (UML, por sus siglas en inglés), el cual constituye una herramienta de diseño que se está expandiendo rápidamente entre los programadores orientados a objetos, y que incorpora al modelo E-R.

► ELEMENTOS DEL MODELO ENTIDAD-RELACIÓN

Los elementos clave del modelo E-R son entidades, atributos, identificadores y relaciones. Consideremos cada uno de éstos.

¹ P. P. Chen, "The Entity-Relationship Model—Towards a Unified View of Data", *ACM Transactions on Database Systems*, enero de 1976, pp. 9-36.

² T. J. Teorey, D. Yang, y J. P. Fry, "A Logical Design Methodology for Relational Databases Using the Extended Entity-Relationship Model", *ACM Computing Surveys*, junio de 1986, pp. 197-222.

³ Thomas A. Bruce, *Designing Quality Databases with IDEF1X Information Models* (Nueva York; Casa de Publicidad Dorset, 1992).

ENTIDADES

Una **entidad** es algo que se puede identificar en el medio ambiente de trabajo de los usuarios; es decir, aquello a lo cual los usuarios quieren dar seguimiento. Algunos ejemplos de entidades son EMPLEADO (EMPLOYEE) Mary Doe, CLIENTE 12345, ORDEN-VENTA (SALES-ORDER) 1000, VENDEDOR (SALESPERSON) John Smith, y PRODUCTO (PRODUCT) A4200. Las entidades de determinado tipo se agrupan en **clases de entidades**. Así, la clase de entidad EMPLEADO es un conjunto de todas las entidades EMPLEADO. En este texto, las clases de entidades están impresas con letras mayúsculas.

Es importante comprender las diferencias entre una clase de entidad y una instancia de entidad. Una *clase de entidad* es un conjunto de entidades y se describe mediante la estructura o formato de las entidades en esa clase. Una *instancia* de entidad es la representación de una entidad en particular, tal como CLIENTE 12345; se describe mediante los valores de los atributos de ésta. Por lo general existen muchas instancias de entidad en una clase de entidad. Por ejemplo, dentro de la clase CLIENTE, existen muchas instancias, una para cada cliente representado en la base de datos. Una clase de entidad y dos de sus instancias se muestran en la figura 3-1.

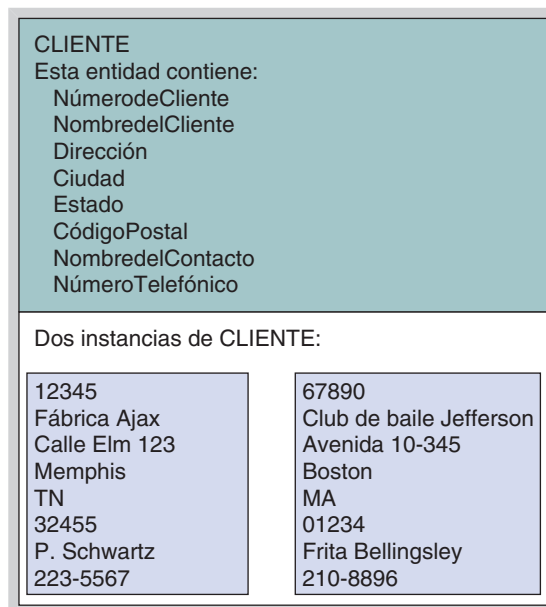
ATRIBUTOS

Las entidades tienen **atributos**, o **propiedades**, como a veces se les llama, que describen las características de la entidad. Algunos ejemplos de atributos son: NombredelEmpleado, FechadeContrato y CódigodeAptitudesdeTrabajo. En este texto, los atributos están impresos en mayúsculas y minúsculas. El modelo E-R supone que todas las instancias de cierta clase de entidad tienen los mismos atributos.

La definición original del modelo E-R incluye atributos multivalor y compuesto. Un ejemplo de un atributo compuesto es la Dirección, la cual consta del grupo de atributos {Calle, Ciudad, Estado/Provincia, CP}. Un ejemplo de un atributo multivalor es NombredelContacto en CLIENTE, donde más de un nombre de una persona está asociado con un cliente específico. Un atributo puede ser tanto multivalor como compuesto; por ejemplo, el atributo compuesto Teléfono {CódigodeÁrea, Número} podría ser multivalor para permitir una gran cantidad de números telefónicos. Muchas implementaciones del modelo E-R ignoran los atributos compuestos de un solo valor. Requieren atributos multivalor (compuestos o no) para ser transformadas en entidades, como se muestra a continuación.

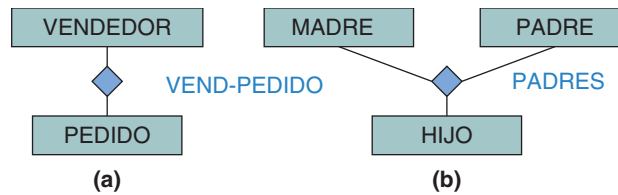
► FIGURA 3-1

CLIENTE: un
ejemplo de entidad



► FIGURA 3-2

Relaciones de grados diferentes:
(a) ejemplo de relación de grado 2, y
(b) ejemplo de relación de grado 3



IDENTIFICADORES

Las instancias de una entidad tienen **identificadores**, los cuales son atributos que nombran, o identifican, las instancias de una entidad. Por ejemplo, las instancias EMPLEADO podrían ser identificadas mediante el NúmerodeSeguroSocial (SocialSecurityNumber), el NúmerodeEmpleado (EmployeeNumber), o el NombredeEmpleado (EmployeeName). Las instancias EMPLEADO no parecen ser identificadas por atributos tales como Salario o FechadelContrato, porque normalmente no se usan en una función de selección. De igual manera, los CLIENTES podrían ser identificados por NúmerodeCliente o NombredeCliente, y PEDIDOS por el NúmerodePedido.

El identificador de una instancia de entidad consta de uno o más de los atributos de ésta. Un identificador puede ser **único** o **no único**. Si es único, su valor identificará solamente un ejemplo de entidad. Si no lo es, el valor identificará una serie de instancias. El NúmerodeEmpleado es más parecido a un identificador único, mientras que NombredeEmpleado es similar a un identificador no único (por ejemplo, puede haber varios John Smith).

Los identificadores que constan de dos o más atributos se llaman **identificadores compuestos**. Algunos ejemplos son {Código de Área, Número Local}, {Nombre de Proyecto, Nombre de la Tarea}, y {Nombre, Apellido, Extensión Telefónica}.

RELACIONES

Las entidades pueden asociarse con otras mediante **relaciones**. El modelo E-R contiene tanto clases de relaciones como instancias de relaciones.⁴ Las *clases de relaciones* son asociaciones entre las clases de entidad, y las *instancias de relaciones* son asociaciones entre las instancias de entidad. Las relaciones pueden tener atributos.

Una clase de relación puede involucrar muchas clases de entidades. El número de éstas en la relación es el **grado** de ésta. En la figura 3-2(a) la relación VEND-PEDIDO es de grado 2 porque involucra dos clases de entidades: VENDEDOR y PEDIDO. La relación PADRES en la figura 3-2(b) es de grado 3, porque involucra tres clases de entidades: MADRE, PADRE e HIJO. Las relaciones de grado 2 son muy comunes y con frecuencia se denominan **relaciones binarias**.

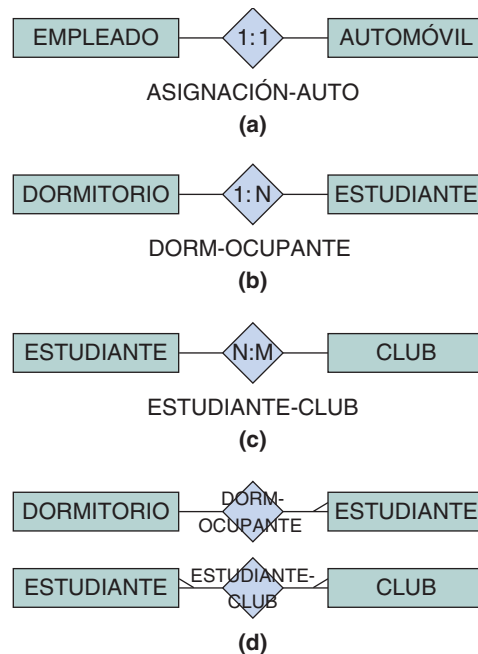
TRES TIPOS DE RELACIONES BINARIAS. La figura 3-3 muestra tres tipos de relaciones binarias. En una relación 1:1 (léase “uno a uno”), una instancia de entidad de un tipo se relaciona con una instancia de una sola entidad de otro tipo. En la figura 3-3(a), la relación de ASIGNACIÓN-AUTO (AUTO-ASSIGNMENT) asocia a un único EMPLEADO con un solo AUTO. De acuerdo con este diagrama, ningún empleado tiene más de un automóvil asignado y ningún automóvil se asigna a más de un empleado.

La figura 3-3(b) muestra el segundo tipo de relación, 1:N (léase “uno a N” o “uno a muchos”). En esta relación, denominada DORM-OCUPANTE (DORM-OCCUPANT),

⁴ Con el fin de abreviar, algunas veces eliminaremos la palabra *instancia* cuando el contexto indique claramente que se refiere a una instancia, en lugar de a una clase de entidad.

► FIGURA 3-3

Tres tipos de relaciones binarias:
(a) relación binaria 1:1, (b) relación binaria 1:N, (c) relación binaria N:M, y (d) representación de relación con pata de gallo



una sola instancia de DORMITORIO relaciona a muchas instancias de ESTUDIANTE. De acuerdo con este esquema, un dormitorio tiene muchos estudiantes, pero un estudiante tiene sólo un dormitorio.

Las posiciones del 1 y de la N son significativas. El 1 está cerca de la línea que conecta con DORMITORIO, lo cual significa que el 1 se refiere al lado DORMITORIO de la relación, y la N está cerca de la línea que conecta con ESTUDIANTE, lo cual significa que la N se refiere a la parte ESTUDIANTE de la relación. Si el 1 y la N se invirtieran y la relación se escribiera N:1, un DORMITORIO tendría un ESTUDIANTE y un ESTUDIANTE tendría muchos DORMITORIOS. Éste, por supuesto, no es el caso.

La figura 3-3(c) muestra el tercer tipo de relación binaria, N:M (léase "N a M" o "muchos a muchos"). Esta relación se llama ESTUDIANTE-CLUB y relaciona las instancias de ESTUDIANTE con las de CLUB. Un estudiante puede reunirse en más de un club, y un club puede tener muchos estudiantes.

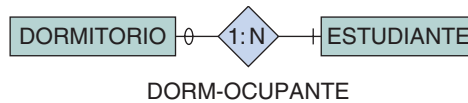
Los números dentro del diamante de la relación muestran la cantidad máxima de entidades que puede haber en un lado de la relación. Estas restricciones se llaman **cardinalidad máxima** de la relación. La relación en la figura 3-3(b), por ejemplo, se dice que tiene cardinalidad máxima de 1:N. Pero las cardinalidades no están limitadas a los valores que se muestran aquí. Es posible, por ejemplo, que la cardinalidad máxima sea distinta de 1 y N. La relación entre EQUIPO-DE-BÁSQUETBOL y JUGADOR, por ejemplo, podría ser 1:5, lo cual indicaría que un equipo de básquetbol tiene más de cinco jugadores.

Las relaciones de los tipos que se muestran en la figura 3-3 a veces se denominan **relaciones TIENE-UN**. Este término se utiliza porque una entidad *tiene una* relación con otra entidad. Por ejemplo, un EMPLEADO tiene un AUTO; un ESTUDIANTE tiene un DORMITORIO; y un CLUB tiene ESTUDIANTES.

DIAGRAMAS DE ENTIDAD-RELACIÓN. Las representaciones de la figura 3-3 se llaman **diagramas entidad-relación** o **diagramas E-R**. Éstos, aunque están estandarizados, se construyen libremente. De acuerdo con este estándar, las clases de entidad se muestran con rectángulos; las relaciones, con diamantes, y la cardinalidad máxima

► FIGURA 3-4

Relación que muestra la cardinalidad mínima



de la relación se indica dentro del diamante;⁵ el nombre de la entidad, dentro del rectángulo, y el nombre de la relación está cerca del diamante.

Aunque en algunos diagramas E-R el nombre de la relación está dentro del diamante, esto puede hacer que el diagrama parezca difícil, ya que los diamantes pueden ser grandes y estar fuera de escala con el fin de incluir el nombre de la relación. Para evitar esto, los nombres de las relaciones a veces se escriben sobre el diamante. Cuando el nombre se coloca dentro o en la punta del diamante, la cardinalidad de la relación se muestra por medio de “patas de gallo” en las líneas que se conectan a la(s) entidad(es) en la mayoría de los lados de la relación. En la figura 3-3(d) se muestran las relaciones DORM-OCUPANTE y ESTUDIANTE-CLUB con esas “patas de gallo”.

Como ya lo establecimos, la cardinalidad máxima indica el número máximo de entidades que pueden estar involucradas en una relación. Los diagramas no indican la mínima. Por ejemplo, la figura 3-3(b) muestra que un estudiante está relacionado, como máximo, con un dormitorio, pero no muestra si un estudiante *debe estar* relacionado con una instancia de dormitorio.

Se utilizan varias formas diferentes para mostrar la **cardinalidad mínima**. Una de ellas, que se ilustra en la figura 3-4, es poner una marca a lo largo de la línea de relación para indicar que debe existir una entidad en la relación, y colocar un óvalo para señalar que ahí puede o no haber una entidad en la relación. En tal sentido, la figura 3-4 muestra que un DORMITORIO debe tener una relación cuando menos con un ESTUDIANTE, pero que un ESTUDIANTE no necesariamente debe estar relacionado con un DORMITORIO. La relación completa de restricciones indica que un DORMITORIO tiene una cardinalidad mínima de uno y una cardinalidad máxima de muchas entidades de ESTUDIANTE. Un ESTUDIANTE tiene una cardinalidad mínima de cero y una cardinalidad máxima de una entidad DORMITORIO.

Puede existir una relación entre entidades de la misma clase. Por ejemplo, la relación HABITACIONES-CON (ROOMS-WITH) se podría definir en la entidad ESTUDIANTE. La figura 3-5(a) muestra esta relación, y la figura 3-5(b) ejemplifica las instancias de entidades que conforman esta relación. A las relaciones entre entidades de una sola clase a veces se les llama **relaciones recursivas**.

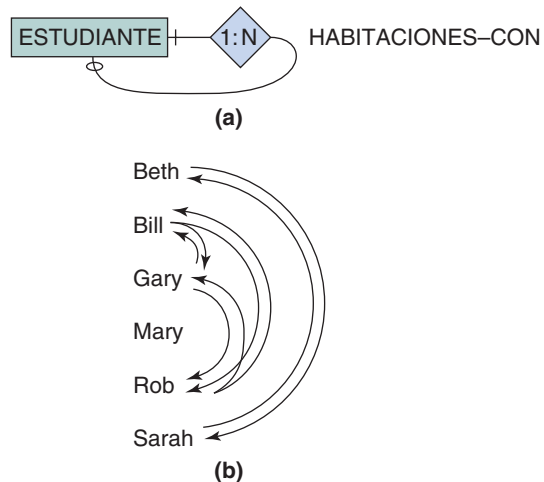
MUESTRAS DE ATRIBUTOS EN LOS DIAGRAMAS ENTIDAD-RELACIÓN. En algunas versiones de los diagramas E-R los atributos se muestran en óvalos, que están conectados a la entidad o a la relación que pertenecen. La figura 3-6(a) ejemplifica las entidades ESTUDIANTE y DORMITORIO y la relación DORM-OCUPANTE con los atributos. Como se muestra, DORMITORIO tiene los atributos NombredelDormitorio, Ubicación, y NúmerodeHabitaciones, y ESTUDIANTE tiene los atributos NúmerodeEstudiante, NombredelEstudiante y Año delEstudiante. La relación DORM-OCUPANTE tiene el atributo Renta, el cual muestra la cantidad que paga un estudiante en particular por determinado dormitorio.

Si una entidad tiene muchos atributos, enumerarlos de esta manera en el diagrama E-R puede ocasionar desorden y dificultad para interpretarlos. En estas instancias, los atributos de entidad se listan por separado, como se muestra en la figura 3-6(b). Muchas herramientas CASE muestran estos atributos en Windows pop-up.

⁵ Los símbolos gráficos que surgieron con el modelo (los cuales se describen aquí) no son los más indicados para mostrar un modelo en un sistema de Interfaz Gráfica de Usuario (GUI, por sus siglas en inglés), como Macintosh o como Microsoft Windows. En realidad, el modelo E-R fue desarrollado antes de que fuera popular cualquier sistema GUI. Los símbolos UML que se muestran posteriormente en este capítulo se utilizan con más facilidad en un medio ambiente gráfico.

► FIGURA 3-5

Relación recursiva



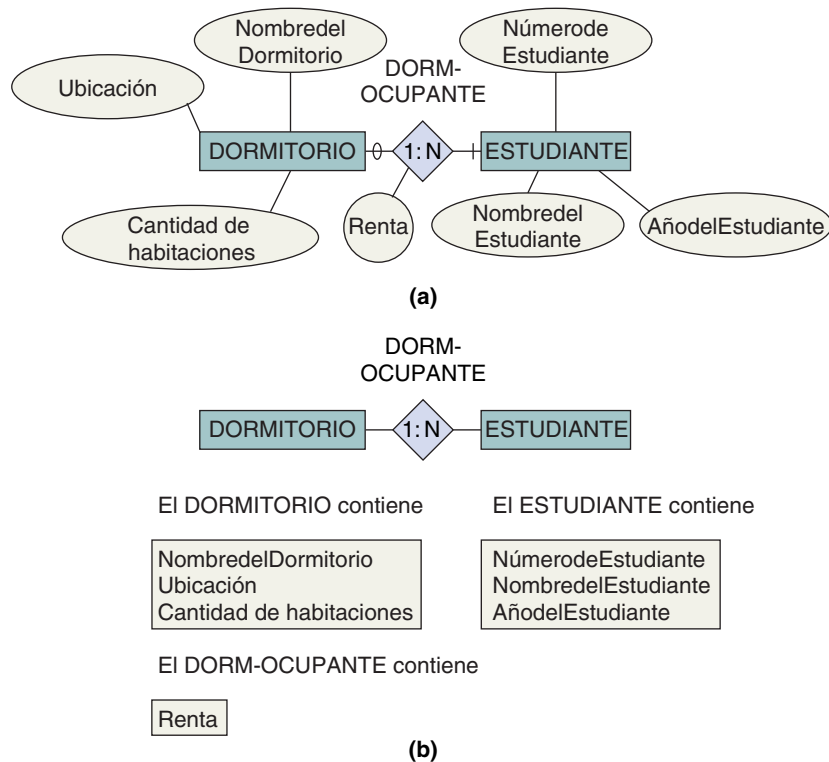
ENTIDADES DÉBILES

El modelo entidad-relación define un tipo especial de entidad llamada **entidad débil**. Ésta no puede existir en la base de datos, a menos que también haya otro tipo de entidad en la base de datos. Una entidad que no es débil se llama **entidad fuerte**.

Para entender a las entidades débiles considere una base de datos humana, con clases de entidades EMPLEADO y ENCARGADO. Suponga que el negocio tiene una regla referente a que una instancia EMPLEADO *puede existir* sin tener una relación con cualquier entidad ENCARGADO, pero una entidad ENCARGADO *no puede existir* sin tener una relación con una entidad particular EMPLEADO. En este caso, ENCARGADO es una entidad débil. Esto significa que los datos de ENCARGADO se pueden almacenar en la base de datos únicamente si ENCARGADO tiene una relación con una entidad EMPLEADO.

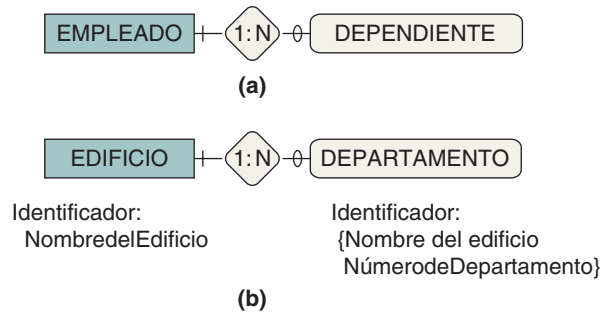
► FIGURA 3-6

Propiedades que se muestran en los diagramas entidad-relación:
(a) diagrama entidad-relación que muestra las propiedades, y
(b) diagrama entidad-relación con las propiedades listadas por separado



► FIGURA 3-7

Entidades débiles:
(a) ejemplo de una
entidad débil, y
(b) entidad
dependiente de ID



Como se muestra en la figura 3-7(a), las entidades débiles se representan mediante las esquinas redondeadas del rectángulo de la entidad. Además, la relación de la cual depende la existencia de la entidad se muestra en un diamante que también tiene las esquinas redondeadas. Alternativamente, en algunos diagramas E-R (que no se muestran aquí), las entidades débiles se representan usando una línea doble para delimitar el rectángulo de la entidad débil, y diamantes dobles para denotar la relación de la cual depende.

El modelo E-R incluye un tipo especial de entidad débil llamada **entidad dependiente de un identificador**. En ésta, el identificador de una entidad incluye al de otra. Considere las entidades EDIFICIO (BUILDING) y DEPARTAMENTO (APARTMENT). Suponga que el identificador EDIFICIO es NombredelEdificio, y que el identificador de DEPARTAMENTO es el identificador compuesto {NombredelEdificio, NúmerodeDepartamento}. Puesto que el identificador de DEPARTAMENTO contiene al de EDIFICIO (NombredelEdificio), entonces DEPARTAMENTO es dependiente del identificador en EDIFICIO. Compare la figura 3-7(b) con la figura 3-7(a). Otra manera de pensar esto es que, tanto lógica como físicamente, un DEPARTAMENTO no puede existir a menos que haya un EDIFICIO.

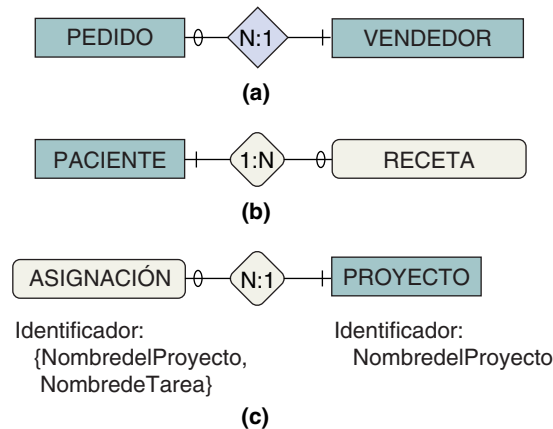
Las entidades dependientes de un identificador son comunes. Otro ejemplo es la entidad VERSIÓN en la relación entre PRODUCTO y VERSIÓN, donde PRODUCTO es un software y VERSIÓN es la salida de ese producto de software. El identificador de PRODUCTO es NombredelProducto y el identificador de VERSIÓN es {NombredelProducto, NúmerodeEdición}. Un tercer ejemplo es EDICIÓN en la relación entre LIBRO DETEXTO y EDICIÓN. El identificador de LIBRO DETEXTO es el título y el identificador de EDICIÓN es {Título, NúmerodeEdición}.

Por desgracia, existe una ambigüedad oculta en la definición de entidad débil, y los diseñadores de bases de datos (al igual que algunos autores de libros de texto) la interpretan de manera diferente. La ambigüedad, en un sentido estricto, radica en lo siguiente: si una entidad débil es definida como cualquier entidad cuya presencia en la base de datos depende de otra entidad, entonces cualquier entidad que participe en una relación con una cardinalidad mínima de 1 en una segunda entidad, es una entidad débil. De esta manera, en una base de datos académica, si ESTUDIANTE debe tener un ASESOR, entonces ESTUDIANTE es una entidad débil, porque una entidad ESTUDIANTE no puede ser almacenada sin un ASESOR. Sin embargo, esta interpretación también parece amplia para algunas personas. Un ESTUDIANTE no es físicamente dependiente de un ASESOR (a diferencia de DEPARTAMENTOS y EDIFICIOS), y un ESTUDIANTE no es lógicamente dependiente de un ASESOR (¡a pesar de lo que le pueda parecer a cualquiera de los dos!); por lo tanto, ESTUDIANTE debe ser considerada una entidad fuerte.

Para evitar estas situaciones, algunas personas interpretan la definición de entidad débil de manera más restringida. Para que sea una entidad débil debe depender *lógicamente* de otra. De acuerdo con esta definición, tanto ENCARGADO como DEPARTAMENTO podrían ser consideradas entidades débiles; pero ESTUDIANTE, no. Un ENCARGADO no puede ser un encargado a menos que algo dependa de él, y un DEPARTAMENTO no puede existir sin un EDIFICIO del cual formar parte. Sin embargo, ESTUDIANTE puede existir lógicamente sin ASESOR, aunque lo requiera una regla de negocio.

► FIGURA 3-8

Ejemplos de entidades requeridas



Para ilustrar esta interpretación, consideremos varios ejemplos. Suponga que un modelo de datos incluye la relación entre PEDIDO y VENDEDOR (figura 3-8[a]). Aun cuando podemos decir que un PEDIDO debe tener un VENDEDOR, no necesariamente requiere uno para existir (el PEDIDO puede ser una venta en efectivo en la que no se registra al vendedor). De ahí que la cardinalidad mínima de 1 derive de una regla de negocios, no de una necesidad lógica. Así, PEDIDO requiere un VENDEDOR pero su existencia no depende de él, y PEDIDO puede ser considerado una entidad fuerte.

Ahora, considere la relación entre PACIENTE (PATIENT) y RECETA (PRESCRIPTION) de la figura 3-8(b). Aquí, una RECETA no puede existir lógicamente sin un PACIENTE, así que no sólo tiene la cardinalidad mínima 1, sino que depende de la existencia de PACIENTE. Así, RECETA es una entidad débil. Por último, considere ASIGNACIÓN (ASSIGNMENT) en la figura 3-8(c), donde el identificador de ASIGNACIÓN contiene al identificador de PROYECTO. Aquí, no sólo ASIGNACIÓN tiene una cardinalidad mínima de 1, y no sólo ASIGNACIÓN es dependiente de la existencia de un PROYECTO, sino también ID es dependiente de PROYECTO, puesto que su llave incluye la llave de otra entidad. De esta manera, ASIGNACIÓN es una entidad débil.

En este texto definimos a las entidades débiles como aquellas que deben depender lógicamente de otra. Por lo tanto, no todas las entidades que tienen una cardinalidad mínima de 1 en su relación con otra entidad son débiles. Sólo aquellas que son lógicamente dependientes se denominan débiles. Esta definición también implica que todas las entidades dependientes de un identificador son débiles. Además, cada entidad débil tiene una cardinalidad mínima de 1 en la entidad de la cual depende; pero cada entidad que tiene una cardinalidad mínima de 1 no necesariamente debe ser débil.⁶

REPRESENTACIÓN DE ATRIBUTOS MULTIVALOR CON ENTIDADES DÉBILES.

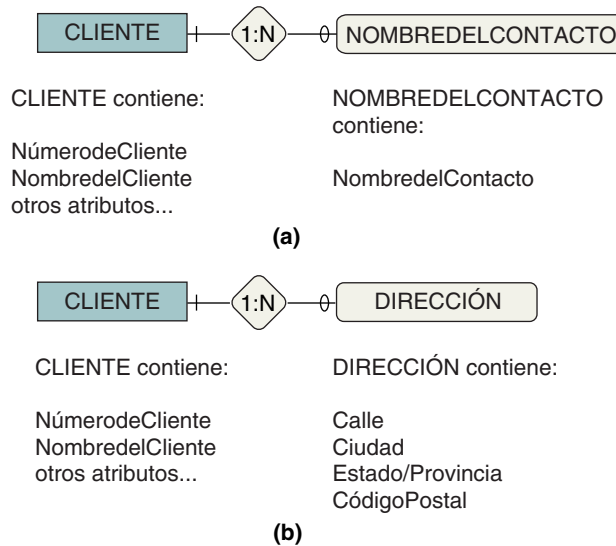
Los atributos multivalor se representan en un diagrama E-R con el fin de crear una nueva entidad débil para representar el atributo multivalor y construir una relación uno a muchos. Por ejemplo, la figura 3-9(a) muestra la representación del atributo multivalor Nombre del Contacto en CLIENTE. Una nueva entidad débil llamada NOMBRE DEL CONTACTO se crea con un solo atributo Nombre del Contacto. La relación entre CLIENTE y NOMBRE DEL CONTACTO es uno a muchos. La entidad construida debe ser débil porque es lógicamente dependiente de la entidad que tiene el atributo multivalor.

La figura 3-9(b) muestra la representación del atributo compuesto multivalor Dirección. La nueva entidad débil DIRECCIÓN contiene todos los atributos del compuesto: Calle, Ciudad, Estado/Provincia, CP.

⁶ Este análisis omite las instancias en las que la cardinalidad mínima es mayor a 1. La lógica es similar, pero la entidad depende de un grupo de entidades.

FIGURA 3-9

Representación de
atributos multivalor
con entidades débiles



ENTIDADES SUBTIPO

Algunas entidades contienen conjuntos opcionales de atributos; éstas a menudo se representan utilizando subtipos.⁷ Considere, por ejemplo, CLIENTE, con atributos NúmerodeCliente, NombredeCliente y CantidadqueAdeuda. Suponga que un CLIENTE puede ser una persona, una sociedad o una empresa y que los datos adicionales que se pueden almacenar dependen de qué tipo sean. Suponga que los datos adicionales son los siguientes:

CLIENTE-PERSONA

Dirección, NúmerodeSeguroSocial

CLIENTE-SOCIEDAD

NombredeSocioAdministrador, Dirección, NúmerodeIdentificaciónFiscal

CLIENTE-EMPRESA

PersonadeContacto, Teléfono, NúmerodeIdentificaciónFiscal

Una posibilidad es asignar todos estos atributos a la entidad CLIENTE, como se muestra en la figura 3-10(a). En este caso, algunos de los atributos no son aplicables. NombredeSocioAdministrador no tiene significado para un cliente o una empresa, y por lo tanto, no puede tener ningún valor.

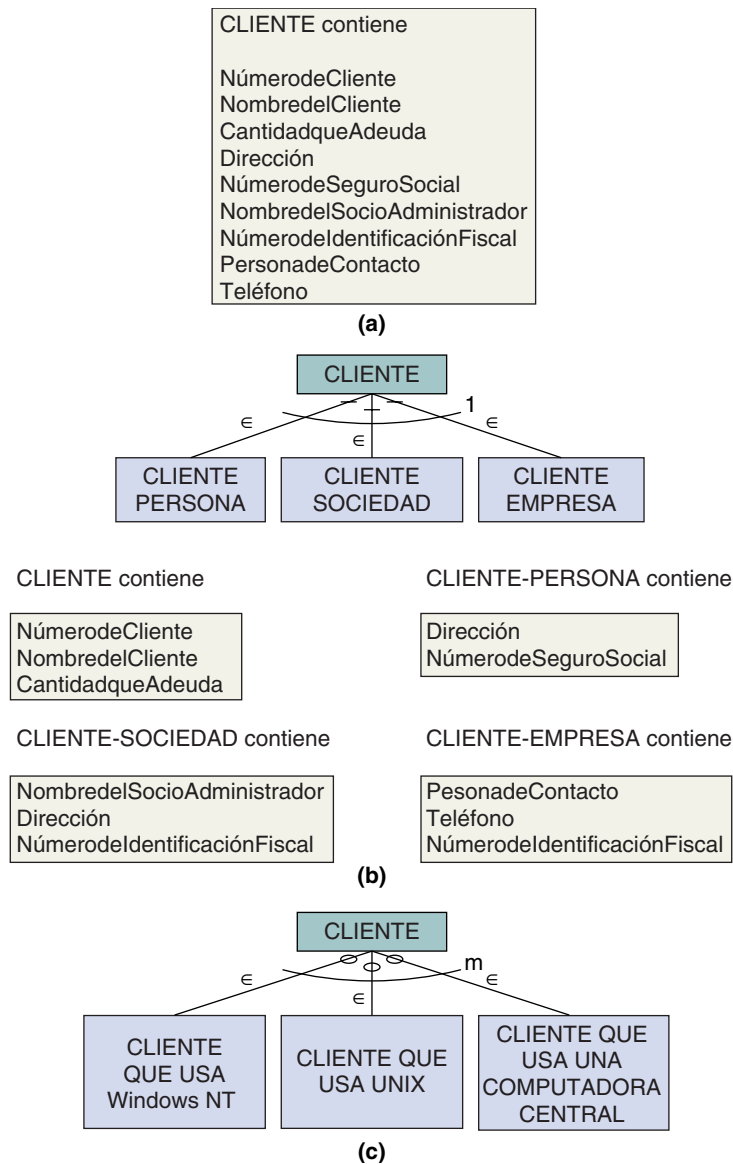
Un modelo más preciso definiría tres entidades subtipo, como se muestra en la figura 3-10(b). Aquí las entidades CLIENTE-PERSONA, CLIENTE-SOCIEDAD y CLIENTE-EMPRESA se muestran como **subtipos** de CLIENTE. A su vez, CLIENTE es un **supertipo** de las entidades CLIENTE-PERSONA, CLIENTE-SOCIEDAD y CLIENTE-EMPRESA.

El símbolo $\in$ junto a las líneas de relación indica que CLIENTE-PERSONA, CLIENTE-SOCIEDAD y CLIENTE-EMPRESA son subtipos de CLIENTE. La línea curva con un 1 próximo indica que la entidad CLIENTE debe pertenecer únicamente a un subtipo. Significa que los subtipos son excluyentes y que se requiere uno de ellos.

⁷ Los subtipos se agregaron al modelo E-R después de la publicación del artículo original de Chen, y son parte de lo que se llama el *modelo E-R extendido*.

► FIGURA 3-10

Entidades subtipo:
 (a) *CLIENTE* sin
 entidades subtipo,
 (b) *CLIENTE* con
 entidades subtipo, y
 (c) subtipos no
 excluyentes con
 supertipo opcional



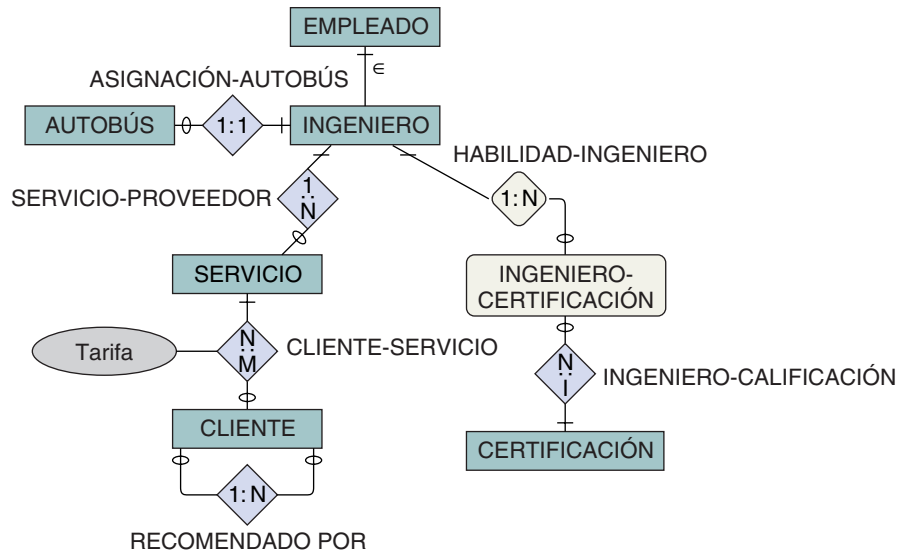
Las entidades con una relación *ES-UN* deberían tener el mismo identificador, puesto que representan diferentes aspectos de una misma cosa. En este caso, el identificador es Número de Cliente. Esta situación contrasta con la relación *TIENE-UN* que se muestra en la figura 3-3, donde las entidades representan aspectos de diferentes cosas.

Las jerarquías de generalización tienen una característica especial llamada **herencia**, lo cual significa que las entidades en subtipos heredan atributos de la clase de entidad supertipo. Por ejemplo, *CLIENTE-SOCIEDAD*, hereda de *CLIENTE* *Nombre del Cliente* y *Cantidad que Adeuda*.

Las razones para utilizar subtipos en la modelación de datos difieren de las razones para usarlos en programación orientada a objetos. De hecho, la única razón para usarlos en el modelo de datos es evitar situaciones en las cuales se requieran algunos atributos que sean nulos. Por ejemplo, en la figura 3-10(a), si el *Número de Seguro Social* tiene un valor, entonces los últimos cuatro atributos deben ser nulos. La situación puede ser más obvia en aplicaciones médicas; por ejemplo, el hecho de preguntarle a un hombre enfermo cuántos embarazos ha tenido. Los valores nulos se analizan con mayor detalle en el capítulo 6.

FIGURA 3-11

Ejemplo de un diagrama entidad-relación



EJEMPLO DE DIAGRAMA E-R

La figura 3-11 es un ejemplo de diagrama E-R con todos los elementos del modelo E-R que hemos estudiado. Muestra las entidades y las relaciones para una compañía consultora de ingeniería que analiza la construcción y estado de casas y de otros edificios e instalaciones.

Hay un tipo de entidad para los empleados de la compañía. Debido a que algunos EMPLEADOS son INGENIEROS, hay un subtipo de relación entre EMPLEADO e INGENIERO. Cada INGENIERO debe ser un EMPLEADO; el INGENIERO tiene una relación de 1:1 con AUTOBÚS, y cada AUTOBÚS debe estar asignado a un INGENIERO, pero no todos los INGENIEROS tienen un AUTOBÚS.

Los INGENIEROS proporcionan SERVICIOS a los CLIENTES. Un INGENIERO puede proporcionar desde cero hasta muchos servicios, pero un SERVICIO debe ser proporcionado sólo por determinado INGENIERO, y ese servicio únicamente lo puede realizar ese ingeniero en particular. Los CLIENTES tienen muchos SERVICIOS, y un SERVICIO puede ser solicitado por muchos CLIENTES. Un CLIENTE debe haber comprado por lo menos un SERVICIO, pero un SERVICIO no necesariamente tiene que haber tenido CLIENTES. La relación CLIENTE-SERVICIO tiene un atributo Tarifa, el cual muestra la cantidad que un cliente en particular paga por un SERVICIO determinado. (En este diagrama no se muestran otros atributos de entidades y relaciones.)

Algunas veces unos CLIENTES envían a otros, lo cual se indica mediante la relación recursiva RECOMENDADO-POR. Un CLIENTE puede recomendar a uno o a varios CLIENTES más. Un CLIENTE puede o no haber sido recomendado por otro cliente, pero un CLIENTE puede ser recomendado sólo por otro CLIENTE.

La entidad INGENIERO-CERTIFICACIÓN muestra que un INGENIERO ha concluido su educación y ha obtenido la aprobación necesaria para tener un certificado. Un INGENIERO puede obtener varias CERTIFICACIONES. La existencia de INGENIERO-CERTIFICACIÓN depende del INGENIERO a través de la relación HABILIDAD-INGENIERO. La CERTIFICACIÓN es la entidad que describe determinada certificación.

DOCUMENTACIÓN DE REGLAS DEL NEGOCIO

En el capítulo 2 definimos un esquema de base de datos que consta de tablas, relaciones, dominios y reglas del negocio. Podemos obtener o inferir los primeros tres a partir de un modelo E-R, pero no podemos obtener reglas del negocio de ese modelo. Así, algunas veces estas reglas se agregan al modelo E-R durante la etapa de la modelación de datos.

El modelo E-R se desarrolla a partir de un análisis de requerimientos de los usuarios, durante el cual con frecuencia surgen las reglas del negocio; de hecho, los analistas de sistemas deben poner mucha atención en las preguntas que hacen con respecto a los requerimientos.

Considere las entidades AUTOBÚS e INGENIERO de la figura 3-11. ¿La empresa tiene reglas para determinar a quién se le asigna un AUTOBÚS? Si no hay suficientes AUTOBUSES para que se le asigne uno a cada INGENIERO, ¿con qué regla se determina a quién se le asigna uno? Podría ser que la aplicación de la base de datos sea asignar autobuses dependiendo de cuál INGENIERO tenga que prestar el mayor número de SERVICIOS fuera de la oficina durante cierto periodo, o alguna otra norma similar.

Otro ejemplo se refiere a la asignación de INGENIEROS a SERVICIOS. Probablemente hay reglas relacionadas con el tipo de INGENIERO-CERTIFICACIÓN que un INGENIERO debe tener para que se le asignen determinados tipos de SERVICIOS. Por ejemplo, para inspeccionar un edificio de departamentos, el INGENIERO puede necesitar una licencia de INGENIERO profesional. Incluso, si no hay una ley que dicte esta regla, la política de la compañía puede ser que este requisito se cumpla.

Las reglas del negocio se pueden o no aplicar a través del DBMS, o por medio del programa de aplicación. Algunas veces las reglas del negocio están escritas en un manual de procedimientos que deben seguir los usuarios de la base de datos. En este punto no es importante la manera en que habrán de imponerse las reglas; lo que interesa es documentarlas para que sean parte de los requerimientos del sistema.

EL MODELO ENTIDAD-RELACIÓN Y LAS HERRAMIENTAS CASE

Desarrollar un modelo de datos utilizando el modelo entidad-relación ha sido más fácil en años recientes, ya que las herramientas para construir los diagramas E-R están incluidas en muchos productos CASE que son muy populares. Productos tales como IEW, IEF, DEFT, ER-WIN y Visio tienen facilidades de dibujo y diagramación para crear diagramas E-R. Estos productos también integran entidades con las relaciones de la base de datos que los representan, los cuales pueden facilitar la administración, el manejo y el mantenimiento de la base de datos.

Para nuestro análisis no consideramos el uso de las herramientas CASE. Pero si en la universidad a la que usted asiste hay dicha herramienta, úsela para crear los diagramas E-R en los ejercicios que se le asignen. Los diagramas E-R creados usando estas herramientas por lo general son visualmente más agradables, y muy fáciles de cambiar y adaptar.

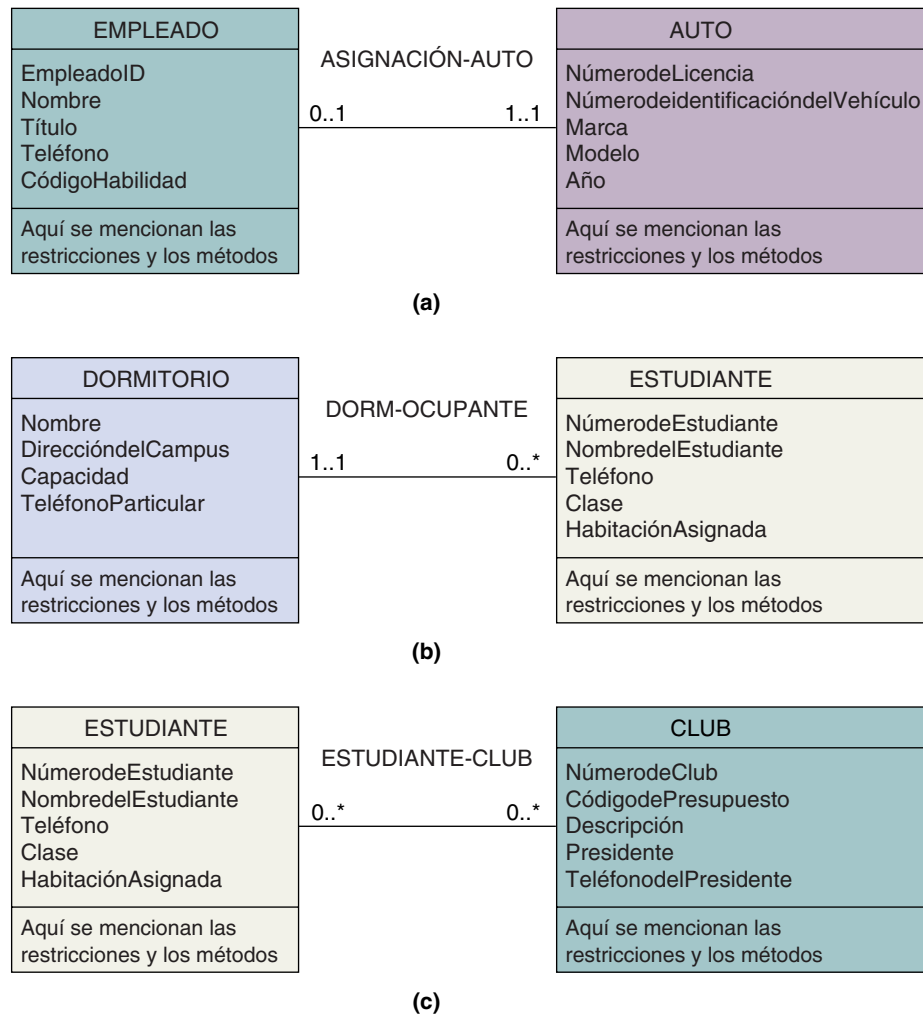
► DIAGRAMAS DE ENTIDAD-RELACIÓN DE ESTILO UML

El Lenguaje de modelación unificado (UML, por sus siglas en inglés) es un conjunto de estructuras y técnicas para la modelación y el diseño de programas orientados a objetos (OOP) y aplicaciones. El UML es tanto una metodología para el desarrollo de sistemas OOP como un conjunto de herramientas para apoyar el desarrollo de dichos sistemas. El UML ha adquirido importancia en el Grupo de administración de objetos (Object Management Group), organización que ha estado desarrollando modelos OOP, tecnología y estándares desde la década de 1980. También ha comenzado a tener un amplio uso entre los profesionales de OOP. El UML es la base de las herramientas de diseño orientadas a objetos de Rational Systems.

Debido a que es una metodología de aplicación de desarrollo, UML es tema para un curso de desarrollo de sistemas, y en este caso no es de mucho interés para nosotros. Sin embargo, usted puede encontrar diagramas entidad-relación de estilo UML, así que será mejor que se familiarice con dicho estilo. Conviene tener en cuenta que en el caso del diseño de la base de datos se trata de los mismos diagramas tradicionales de entidad-relación.

► FIGURA 3-12

Representación
(a) del UML de una
relación 1:1,
(b) representación
UML de una relación
1:N, y
(c) representación
UML de una relación
N:M



ENTIDADES Y RELACIONES UML

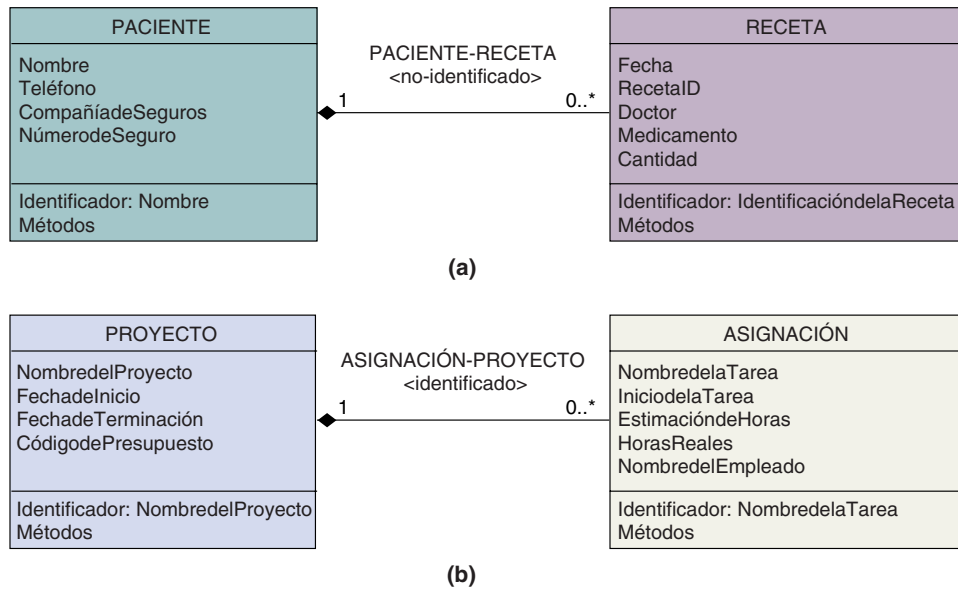
La figura 3-12 muestra la representación UML de los diseños en la figura 3-3. Cada entidad está representada por una **clase de entidad**, la cual se muestra como un rectángulo con tres segmentos. El segmento superior muestra el nombre de la entidad y otros datos que analizaremos. El segundo segmento enumera los nombres de los atributos de la entidad, y el tercero documenta las restricciones y enumera los métodos (procedimientos del programa) que corresponden a la entidad.

Las relaciones se muestran con una línea entre dos entidades. Las cardinalidades se representan en el formato $x..y$, donde x es lo mínimo que se requiere y y es lo máximo permitido. Así, $0..1$ significa que no se requiere una entidad y y que cuando mucho se permite una. Un asterisco significa que se permite un número ilimitado. Así, $1..*$ quiere decir que se requiere una y se permite un número ilimitado. Examine las figuras 3-12(a)-(c) referentes a los ejemplos de relaciones de cardinalidad máxima 1:1, 1:N y N:M.

REPRESENTACIÓN DE ENTIDADES DÉBILES. La figura 3-13 muestra la representación UML de entidades débiles. Un diamante lleno se coloca en la línea padre de la entidad débil (la entidad de la cual depende ésta). En la figura 3-13(a), RECETA es la entidad débil y PACIENTE es la entidad padre. Todas las entidades débiles tienen una entidad padre, y de esta forma la cardinalidad al lado de la entidad débil es siempre

► FIGURA 3-13

Representación UML de entidades débiles. (a) Entidad débil no dependiente de un identificador, y (b) entidad débil dependiente de un identificador



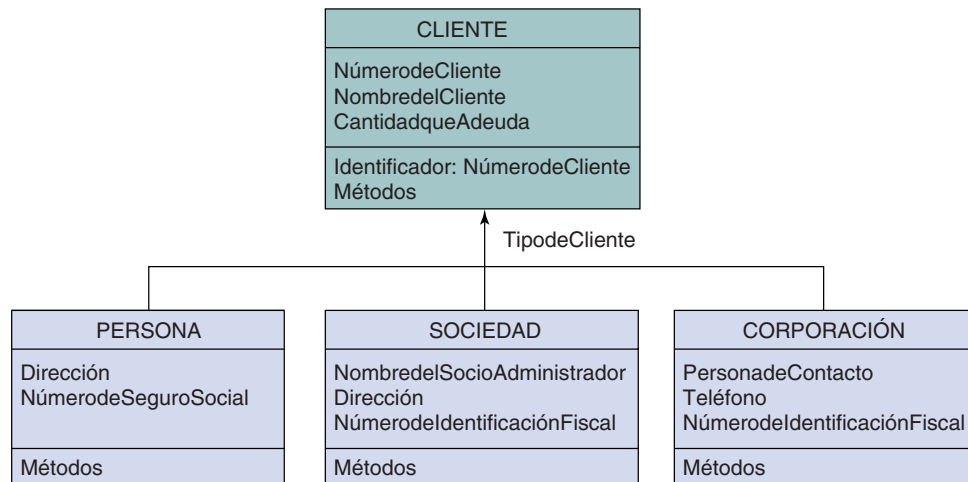
una relación 1..1. Ya que esto es así, la cardinalidad en la entidad padre se muestra simplemente como 1.

La figura 3-13(a) muestra una entidad débil que no es una entidad dependiente de un identificador. Esto se denota por la expresión <no-identificado> en la relación PACIENTE-RECETA. La figura 3-13(b) muestra una entidad débil que es dependiente de un identificador. Esto se especifica con la etiqueta <identificado>.

REPRESENTACIÓN DE SUBTIPOS. El UML representa los subtipos como se muestra en la figura 3-14. En ésta, PERSONA, SOCIEDAD y CORPORACIÓN, subtipos de CLIENTE, se permiten. De acuerdo con esta figura, determinado CLIENTE podría ser uno, dos o tres de estos subtipos. En una instancia así, esto no tiene sentido; un CLIENTE debería ser uno y solamente uno de estos tipos. La versión actual de UML no proporciona un significado al documento exclusivamente. Sin embargo, esta notación podría ser agregada al diagrama UML.

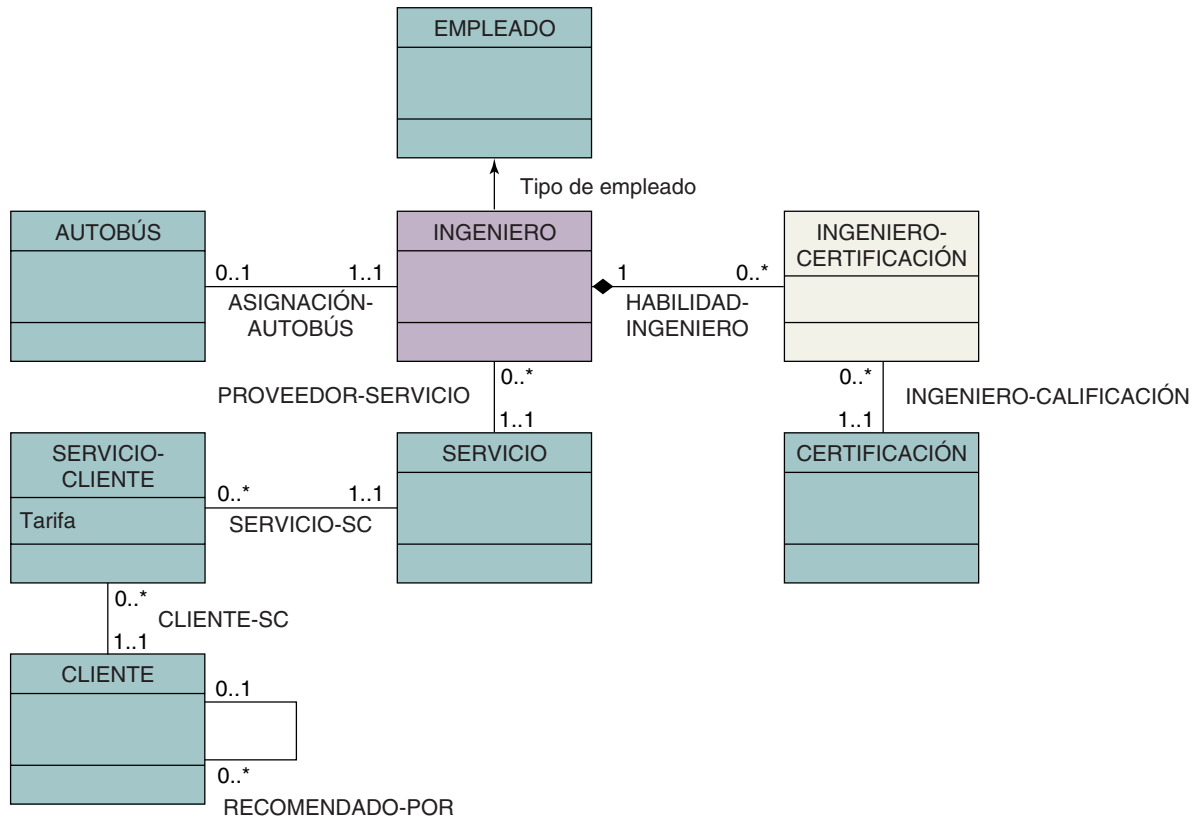
► FIGURA 3-14

Representación de subtipos UML



► FIGURA 3-15

Versión UML del diagrama E-R de la figura 3-11



La figura 3-15 representa una versión UML del diagrama entidad-relación que se mostró en la figura 3-11. Debido a que la relación entre SERVICIO y CLIENTE tiene un atributo —Tarifa—, una entidad por separado CLIENTE-SERVICIO ha sido definida para cumplir este atributo. Ésta es una práctica estándar cuando se utilizan las herramientas UML. También observe la representación de la relación recursiva, RECOMENDADO-POR.

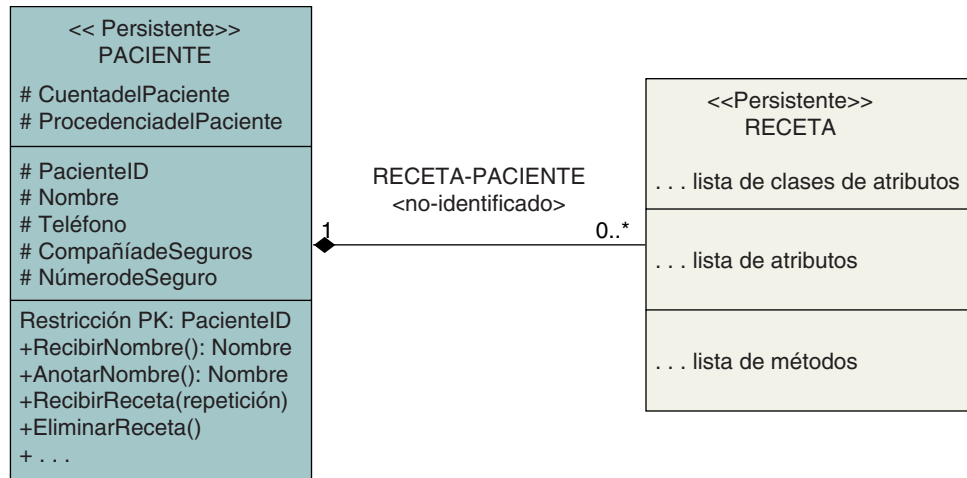
CONSTRUCCIONES OOP INTRODUCIDAS POR MEDIO DE UML

Debido a que UML es una tecnología orientada a objetos, varias construcciones OOP han sido agregadas a las clases de entidad UML. Aquí abordaremos estas ideas y las desarrollaremos en el capítulo 18. Primero, las clases de todas las entidades que serán almacenadas en la base de datos están etiquetadas con la palabra clave <Persistente>. Esto significa que los datos deberán seguir existiendo aun cuando el objeto que se procese esté destruido. En términos simples, significa que la clase de entidad va a ser almacenada en la base de datos.

A continuación, las clases de entidad UML son tomadas en cuenta para las **clases de atributos**, que difieren de los atributos de entidad porque se relacionan con las clases de todas las entidades de un tipo determinado. Así, la figura 3-16, CuentadelPaciente de PACIENTE es un atributo del conjunto de todos los PACIENTEs en la base de datos. ProcedenciadelPaciente es un atributo que documenta la procedencia de todos los PACIENTEs en la base de datos.

► FIGURA 3-16

Clases de entidad
UML con
construcciones OOP



Como verá, estas clases de atributos no tienen razón de ser cuando se utiliza el modelo relacional. En lugar de esto, en algunos casos, atributos tales como CuentadelPaciente no se almacenan en la base de datos, sino que son calculados en el tiempo de ejecución, o run-time. En otros casos, se ingresa una nueva entidad que contenga las clases de atributos. Para la entidad de la figura 3-16 una nueva entidad, llamada PROCEDENCIA-PACIENTE, podría ser definida para contener los atributos CuentadelPaciente y ProcedenciadelPaciente. En este caso todas las entidades en PACIENTE estarán conectadas a PROCEDENCIA-PACIENTE.

Una tercera característica es que UML utiliza la notación orientada a objetos para la visibilidad de los atributos y los métodos. Los atributos que están precedidos por un + son públicos; los precedidos por un # son protegidos, y aquellos con un – son privados. En la figura 3-16, Nombre en PACIENTE es un atributo protegido.

Estos términos surgen de la disciplina de la programación orientada a objetos. Se puede tener acceso a un atributo *público* y cambiar mediante algún método de cualquier objeto. Se puede invocar un método público con cualquier método de cualquier objeto. *Protegido* quiere decir que el atributo o método sólo es accesible mediante métodos de esta clase o de sus subclases, y *privado* significa que es accesible sólo mediante métodos de esta clase.

Por último, están las entidades UML con restricciones específicas y métodos en el tercer segmento de las clases de entidades. En la figura 3-16 se coloca una restricción de llave primaria limitada en PacienteID. Esto simplemente significa que PacienteID es un identificador único. Además, la figura 3-16 documenta que RecibirNombre() se crea para proporcionar acceso público (observe el + al frente de RecibirNombre) para el atributo Nombre; AnotarNombre() se usa para colocar su valor y el método RecibirReceta() se puede usar para intercambiar en el conjunto de entidades Receta con respecto a esta entidad PACIENTE.

EL PAPEL ACTUAL DEL UML EN EL PROCESAMIENTO DE BASES DE DATOS

Las ideas que se ilustran en la figura 3-16 se encuentran en aguas oscuras, donde se combinan el procesamiento de la base de datos y el pensamiento de objetos. Esta notación orientada a objetos no encaja en las prácticas y procedimientos del procesamiento de bases de datos comerciales. El concepto de que un atributo de entidad puede estar oculto en un objeto no tiene sentido, a menos que únicamente los programas orientados a objetos se procesen en la base de datos y, aun entonces, dichos programas deben procesar los datos de acuerdo con esa política. Con excepción de los productos y aplicaciones DBMS orientados a objetos (OODBMS, por sus siglas en inglés), esto nunca se hace.

En cambio, la mayoría de los productos comerciales DBMS tiene características que permiten a todos los tipos de programas acceder a la base de datos y procesar cualquier dato, por lo que tienen autoridad de seguridad. Es más, con dispositivos como el

generador de consultas de Microsoft Access 2002 (figura 2-8) no hay forma de limitar el acceso a los valores de los atributos a un solo objeto.

Así, usted debe saber cómo interpretar diagramas entidad-relación estilo UML. Éstos se pueden utilizar para el diseño de bases de datos, tal como se puede hacer con los diagramas tradicionales de estilo E-R. Sin embargo, actualmente la notación orientada a objetos que introducen tiene un valor práctico limitado. Vea el capítulo 18 para mayor información acerca de este tema.

► EJEMPLOS

La mejor forma que hay para obtener capacidad con cualquier herramienta de modelación es estudiar ejemplos y utilizarla para hacer sus propios modelos. El resto de este capítulo presenta dos instancias de aplicaciones para ayudarle con la primera tarea. Después de estudiar dichos ejemplos, puede trabajar los problemas que están al final del capítulo.

EJEMPLO 1: CLUB DE BAILE JEFFERSON

El club Jefferson imparte clases de baile de salón y ofrece lecciones privadas y en grupo; cobra \$45 la hora por estudiante (o pareja) cuando se trata de una lección privada, y \$6 la hora por estudiante en el caso de una lección en grupo. Las lecciones privadas se imparten a cualquier hora, desde el medio día hasta las 10:00 p.m. seis días a la semana. Las lecciones en grupo se imparten sólo en las tardes.

El club de baile emplea dos tipos de instructores: asalariados de tiempo completo, e instructores de medio tiempo. La remuneración de los instructores de tiempo completo consiste en una cantidad fija por semana y a los de tiempo parcial se les paga cierta cantidad por una tarde, o por impartir sólo una clase.

Además de las lecciones, el club patrocina a la semana dos presentaciones de baile social con música grabada. La cuota de admisión es de \$5 por persona. El baile de los viernes por la noche es el más popular y en promedio se reúnen 80 personas; el baile del domingo por la noche atrae aproximadamente a 30. El propósito de los bailes es proporcionar a los estudiantes un lugar para que practiquen sus habilidades. No se sirven alimentos ni bebidas.

Al club de baile Jefferson le gustaría desarrollar un sistema de información para dar seguimiento a los estudiantes y a las clases que éstos han tomado. Al administrador también le gustaría saber cuántas lecciones y de qué tipo ha impartido cada maestro, y poder calcular el costo promedio por lección de cada uno de sus instructores.

ENTIDADES

La mejor manera para comenzar un modelo entidad-relación es determinar las entidades potenciales. Las entidades normalmente se representan con nombres (lugares, personas, conceptos, sucesos, equipos, etc.) en documentos o en entrevistas. Una búsqueda de ejemplos previos sobre nombres importantes que relacionan al sistema de información revela la siguiente lista:

- Lección privada
- Lección en grupo
- Maestro
- Maestro de tiempo completo
- Maestro de tiempo parcial
- Baile
- Cliente

Evidentemente, los nombres *lección privada* y *lección en grupo* tienen algo en común, al igual que los nombres *maestro*, *maestro de tiempo completo* y *maestro de tiempo*

parcial. Una solución es definir una entidad LECCIÓN, con subtipos LECCIÓN-PRIVADA y LECCIÓN-EN-GRUPO y otra entidad MAESTRO, con subtipos MAESTRO-DE-TIEMPO-COMPLETO y MAESTRO-DE-TIEMPO-PARCIAL. Las entidades adicionales son BAILE y CLIENTE.

Como se estableció en el capítulo 2, la modelación de datos es arte y ciencia. La solución antes descrita es sólo una de las soluciones factibles. Otra posibilidad sería eliminar LECCIÓN y MAESTRO de la lista del párrafo anterior y suprimir todos los subtipos. Una tercera opción sería eliminar LECCIÓN (puesto que lección nunca se mencionó como un nombre), pero mantener MAESTRO y sus subtipos. Aquí elegimos el tercer caso porque parece ser el más adecuado considerando la información que tenemos. Así, la lista de entidades es LECCIÓN-PRIVADA, LECCIÓN-EN-GRUPO, MAESTRO, MAESTRO-DE-TIEMPO-COMPLETO, MAESTRO-DE-TIEMPO-PARCIAL, BAILE y CLIENTE.

Elegir entre estas alternativas requiere analizar los requerimientos y considerar las implicaciones de diseño de cada una. A veces ayuda considerar los atributos de las entidades. Si, por ejemplo, la entidad LECCIÓN no tiene más atributos que su identificador, entonces no es necesaria.

RELACIONES

Para empezar, MAESTRO tiene dos entidades subtipo: MAESTRO-DE-TIEMPO-COMPLETO y MAESTRO-DE-TIEMPO-PARCIAL. Un profesor determinado debe corresponder a uno u otro, así que los subtipos son mutuamente excluyentes.

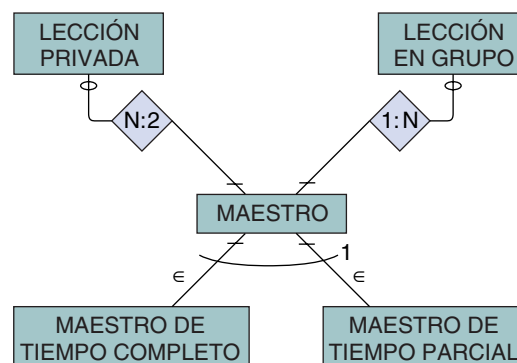
A continuación considere la relación entre MAESTRO y LECCIÓN-PRIVADA y LECCIÓN-EN-GRUPO. Un MAESTRO puede impartir muchas LECCIONES PRIVADAS y normalmente una LECCIÓN-PRIVADA la imparte un solo MAESTRO. Sin embargo, una plática posterior con el administrador reveló que para bailarines avanzados, especialmente los que se preparan para competencias, a veces participan dos maestros en una lección privada. Por lo tanto, la relación entre MAESTRO y LECCIÓN-PRIVADA debe ser de muchos a muchos. Supongamos que sólo un profesor es asignado a una lección en grupo. Las relaciones antes descritas se muestran en la figura 3-17.

Los CLIENTES deben tomar LECCIONES-PRIVADAS o LECCIONES-EN-GRUPO. Algunas veces una persona toma una lección y otras veces una pareja. Hay dos formas en que se puede modelar esta situación. Una entidad PAREJA puede ser definida como una relación uno a dos con CLIENTE, o CLIENTE o PAREJA pueden tener una relación con LECCIÓN-PRIVADA. Supongamos que las parejas no toman lecciones en grupo, o que si lo hacen no es importante almacenar esa información en la base de datos. Esta alternativa se muestra en la figura 3-18(a).

La existencia de LECCIONES PRIVADAS depende de CLIENTE o PAREJA. Esto es, una lección no puede existir a menos que se imparta a cualquiera, a un CLIENTE o a una PAREJA. El 1 en el extremo derecho de la línea horizontal, debajo de CLIENTE y PAREJA, in-

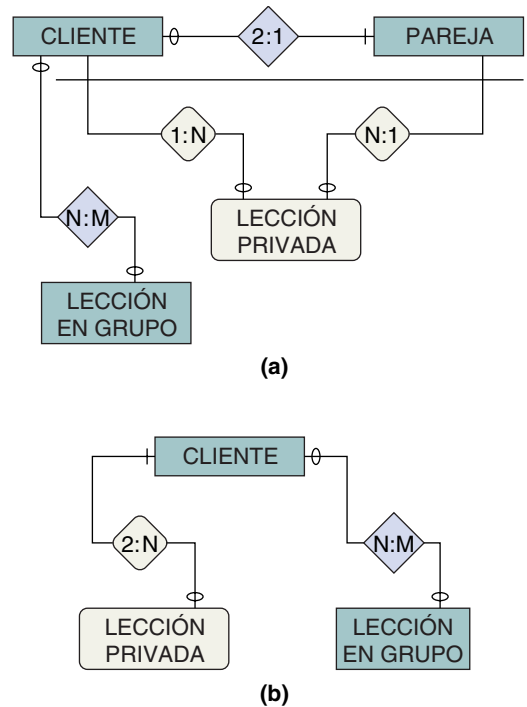
► FIGURA 3-17

Diagrama E-R inicial
para el club de baile
Jefferson



► FIGURA 3-18

Alternativas para la representación de CLIENTE:
 (a) diagrama E-R que muestra la entidad PAREJA, y
 (b) diagrama E-R sin parejas



dica que la LECCIÓN-PRIVADA debe tener cuando menos un CLIENTE o una PAREJA, lo cual tiene sentido puesto que LECCIÓN-PRIVADA depende de ellos.

Otra alternativa es no representar a las parejas, sino modelar la relación entre CLIENTE y LECCIÓN-PRIVADA como muchos a muchos. Específicamente, esta relación es uno o dos a muchos y se muestra en la figura 3-18(b). Aunque el modelo no se detalla como el de la figura 3-18(a), puede ser suficiente para los propósitos del club de baile.

La última relación posible es entre BAILE y otras entidades. Tanto los clientes como los profesores asisten a los bailes, y los analistas deben decidir si es importante o no almacenar estas relaciones. ¿Realmente Jefferson necesita saber qué clientes asisten a cuáles bailes? ¿El administrador del club en realidad quiere que quede registrado en el sistema de información computarizado cuando los clientes atraviesan la puerta? ¿Los clientes también desean que ese hecho quede registrado? Probablemente ésta no es una relación que necesite o deba ser almacenada en la base de datos.

La situación entre BAILE y MAESTRO es diferente. A Jefferson le gusta que sus maestros asistan a todos los bailes. Con el fin de hacer que este requerimiento sea equitativo, el administrador ha programado la asistencia de los maestros a los bailes. Desarrollar y registrar este programa requiere que la base de datos contenga la relación BAI-LE-MAESTRO, la cual es muchos a muchos.

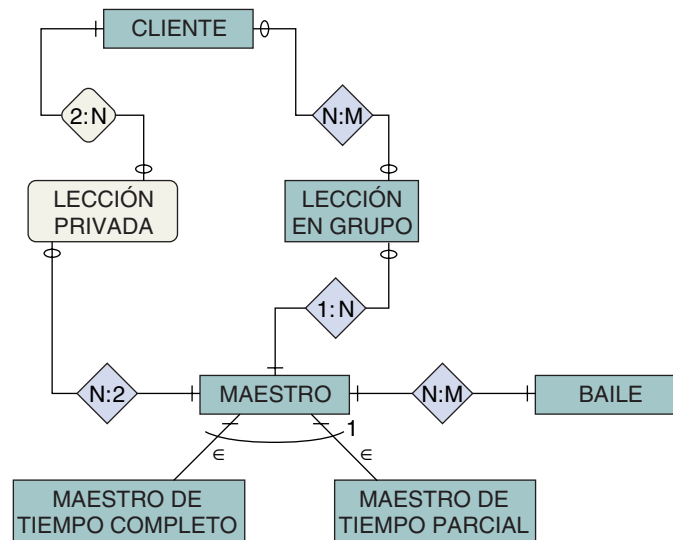
DIAGRAMA E-R FINAL PARA EL CLUB DE BAILE JEFFERSON

La figura 3-19 muestra un diagrama E-R para el modelo descrito en esta sección. No hemos puesto nombre a las relaciones en este diagrama. Aunque al hacerlo los diagramas resultarían como era de esperarse, para nuestros datos, nombrar las relaciones no aportaría nada.

La existencia de LECCIONES PRIVADAS depende de CLIENTE, pero las LECCIONES-EN-GRUPO no, debido a que las lecciones en grupo son programadas mucho antes de que cualquier cliente se inscriba, y a que se seguirán realizando aunque los clientes no asistieran. Sin embargo, esta situación no es verdad en cuanto a las lecciones privadas porque sólo se programan a solicitud del cliente. También observe que este modelo no representa parejas.

► FIGURA 3-19

Diagrama final E-R para el club de baile Jefferson



Una vez que se ha desarrollado un modelo como éste, se debe verificar su precisión y que los requerimientos hayan sido satisfechos. Por lo general esto se hace con los usuarios.

EVALUACIÓN DEL MODELO DE DATOS E-R

Es más fácil y más barato corregir errores al principio del desarrollo de la base de datos que al final. Por ejemplo, cambiar la cardinalidad máxima de una relación de 1:N a N:M en la etapa de modelación de datos sólo es cuestión de registrar el cambio en el diagrama E-R. Pero una vez que se ha diseñado y cargado la base de datos con información y los programas de aplicación escritos para procesarla, hacer ese cambio requerirá muchísimo trabajo y posiblemente semanas para realizarlo. Por lo tanto, es importante evaluar el modelo de datos antes del diseño.

Una técnica de evaluación es considerar el modelo de datos E-R en el contexto de las posibles consultas que se podrían plantear a una base de datos con la estructura que implica el modelo. Por ejemplo, observe el diagrama de la figura 3-19. ¿Qué preguntas podrían contestarse con una base de datos que se implementara de acuerdo con este diseño?

- ¿A quiénes se impartieron las lecciones privadas?
- ¿Cuáles clientes han tomado una lección privada con Jack?
- ¿Quiénes son los maestros de tiempo completo?
- ¿Quiénes son los maestros que están programados para asistir al baile del viernes?

Cuando se evalúe el modelo de datos E-R, usted puede formular estas preguntas y pedir a los usuarios que redacten su propia lista de preguntas. Esas preguntas pueden ser comparadas con el diseño para comprobar su pertinencia. Por ejemplo, suponga que los usuarios preguntan cuáles clientes asistieron al baile del viernes pasado. Los diseñadores del modelo de datos en la figura 3-19 concluirían que su diseño no fue el correcto, porque no es posible contestar esta pregunta utilizando su modelo E-R. Si se debe responder esa pregunta, entonces hay que construir una relación entre CLIENTE y BAILE.

Evidentemente, este proceso informal y libremente estructurado no se puede utilizar para *probar* que un diseño es correcto. Sin embargo, ésta es una técnica pragmática que se puede utilizar para verificar el correcto potencial de un diseño. ¡Y es mejor esto que no hacer ninguna evaluación!

EJEMPLO 2: PASEOS EN BARCO SAN JUAN

Paseos en barco San Juan es una agencia que alquila embarcaciones a sus clientes mediante una tarifa determinada. Dicha agencia no tiene embarcaciones propias, sino que las rentas en nombre de los propietarios que desean obtener ingresos cuando no usan sus botes. San Juan cobra una cuota por sus servicios. Se especializa en alquileres de embarcaciones que se pueden utilizar para pasear durante varios días o semanas. El barco más pequeño en su inventario es de 28 pies de largo y el más grande es de 51 pies.

Cada embarcación está completamente equipada cuando se alquila. La mayor parte del equipo lo proporcionan los dueños, pero San Juan agrega una parte. El dueño proporciona aditamentos fijos tales como radios, brújulas, indicadores de profundidad, así como estufas y refrigeradores. Otro equipo que proporciona el propietario no forma parte del barco, por ejemplo velas, cuerdas, anclas, lanchas, salvavidas, y, en la cabina, platos, cubiertos de plata, utensilios de cocina, ropa de cama y similares. La empresa pone equipo que también podría considerarse como provisiones: diagramas, libros de navegación, tablas de mareas y corrientes, jabón, toallas para trastos, papel higiénico y artículos similares.

Parte importante de las responsabilidades de San Juan es registrar el equipo del barco. Suele ser costoso; en particular, el que no está fijo se puede perder fácilmente o ser robado. Los clientes son responsables de todo el equipo durante el tiempo que alquilan el barco.

A San Juan le gusta conservar los registros exactos de sus clientes y de los alquileres no sólo con fines de mercadotecnia, sino también para registrar los viajes que los clientes han hecho. Algunos itinerarios y condiciones climatológicas son más peligrosos que otros, y le gusta saber cuál es la experiencia de sus clientes.

El negocio principal de San Juan es únicamente el alquiler de los barcos, lo que significa que no prestan servicios de tripulación, tales como marineros u otros ayudantes. Sin embargo, en algunos casos los clientes solicitan los servicios de algún ayudante y entonces la casa contrata a ese personal por horas.

Los barcos de paseo con frecuencia necesitan mantenimiento. Los contratos que hace San Juan con los propietarios incluyen registros precisos de todas las actividades y costos de mantenimiento, incluyendo actividades normales tales como limpieza o cambios de aceite del motor, así como reparaciones no programadas. En algunos casos durante un paseo se necesitan algunas reparaciones. Por ejemplo, el motor de un barco puede fallar cuando éste está lejos de las instalaciones de San Juan. En ese caso, los clientes se comunican por radio y la empresa determina la mejor alternativa de reparación y envía personal al barco descompuesto. Para tomar estas decisiones, los despachadores necesitan información acerca de las opciones de reparación, así como antecedentes en cuanto a calidad y costos.

Antes de que continúe leyendo, intente usted mismo producir un diagrama entidad-relación. Examine los enunciados anteriores y busque los nombres que parezcan importantes para el diseño. Después compruebe las posibles relaciones entre las entidades. Por último, enumere los atributos comunes para cada entidad o relación.

ENTIDADES

El modelo de datos que requiere la casa de alquiler de embarcaciones San Juan es más complicado que el del club de baile Jefferson. Las entidades potenciales se muestran en la figura 3-20(a).

Primero considere las entidades que se relacionan con el equipo. Existen varias clases diferentes de equipo y este dato sugiere la posibilidad de modelar los subtipos. Sin embargo, ¿por qué San Juan tiene a su cargo el cuidado del equipo? En realidad no quiere conocer las características de cada artículo, por ejemplo la longitud de la cadena de cada ancla. Su objetivo es registrar los artículos y de qué tipo son, de tal manera que pueda determinar si falta alguno o si está dañado. Esto se puede hacer sin conservar los registros detallados de los subtipos particulares del equipo. Así, para este diseño, colocamos todos los tipos de equipo en la entidad EQUIPO.

► FIGURA 3-20

*Entidades para
Paseos en barco
San Juan*

ARRENDAMIENTO
BARCO
CLIENTE
PROPIETARIO
EQUIPO
EQUIPO PROPORCIONADO POR EL PROPIETARIO
EQUIPO FIJO DEL PROPIETARIO
EQUIPO REMOVIBLE DEL PROPIETARIO
EQUIPO PROPORCIONADO POR SAN JUAN
ITINERARIO-CLIMA
PASEO
TRIPULACIÓN POR HORAS
MANTENIMIENTO PROGRAMADO
MANTENIMIENTO NO PROGRAMADO
REPARACIONES
INSTALACIONES PARA REPARACIONES

(a) Entidades posibles para Paseos en barco San Juan

ARRENDAMIENTO o PASEO(sinónimos)
BARCO
CLIENTE
PROPIETARIO
EQUIPO
ITINERARIO CLIMA
TRIPULACIÓN POR HORAS
MANTENIMIENTO PROGRAMADO
REPARACIONES o MANTENIMIENTO NO PROGRAMADO(sinónimos)
INSTALACIONES PARA REPARACIÓN

(b) Entidades seleccionadas para el diseño E-R

La propiedad del equipo se establece definiendo una relación entre EQUIPO y PROPIETARIO. Si se le permite a Paseos en barco San Juan ser una instancia de PROPIETARIO, todo el equipo que posee puede ser incluido en esta relación. De manera similar, como en el caso de la descripción, no hay motivo para definir de manera diferente al equipo que está fijo del que no lo está. Se puede elaborar una lista de datos sin esta división. La figura 3-20(b) muestra la lista final de entidades. Observe que ARRENDAMIENTO y PASEO son sinónimos, pues se refieren a la misma transacción. Aquí incluimos ambos para que puedan ser relacionados al caso de la descripción.

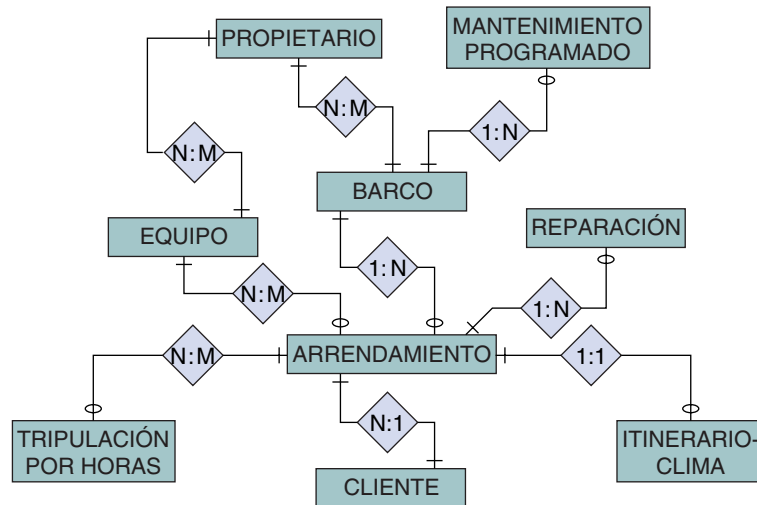
Es posible que MANTENIMIENTO PROGRAMADO deba combinarse con MANTENIMIENTO NO PROGRAMADO. Una forma para decidirlo, es examinar los atributos de cada una de estas entidades. Si son iguales, ambas deben unirse. También observe que REPARACIÓN y MANTENIMIENTO-NO-PROGRAMADO se definen como sinónimos.

RELACIONES

La figura 3-21 es un diagrama entidad-relación de Paseos en barco San Juan. En su mayoría, las relaciones en este diagrama son claras, pero la relación entre EQUIPO y ARRENDAMIENTO es discutible. Uno puede decir que EQUIPO debería estar relacionado con BARCO y no con ARRENDAMIENTO, o que EQUIPO debería relacionarse con BARCO (el equipo que forma parte del barco) y el demás equipo debería estar relacionado con ARRENDAMIENTO. Estos cambios serían alternativas factibles para el diseño que se muestra en la figura 3-21.

► FIGURA 3-21

Diagrama E-R para
Paseos en barco
San Juan



Observe también que MANTENIMIENTO PROGRAMADO se relaciona con BARCO, pero que REPARACIÓN (MANTENIMIENTO NO PROGRAMADO) se relaciona con ARRENDAMIENTO. Esto implica que no se requiere ninguna acción de reparación cuando el barco no está rentado. Quizás esto no sea realista.

Finalmente, ARRENDAMIENTO y ITINERARIO-CLIMA tienen una relación 1:1 y también poseen los mismos atributos de identificación. Por lo tanto, sería posible, e incluso preferible, combinarlos en una clase de entidad.

► BASES DE DATOS COMO MODELOS DE MODELOS

Como puede ver, hay diferentes formas de modelar una situación de negocios, y la variedad es aún más grande conforme la aplicación se vuelve más compleja. Con frecuencia son factibles docenas de modelos y puede ser difícil elegir entre ellos.

A veces, cuando evalúan alternativas, los analistas del proyecto estudian y discuten qué modelo de datos representa mejor al mundo real. Estas discusiones están equivocadas. Las bases de datos no modelan al mundo real, aunque éste es un error común. Las bases de datos son modelos de los modelos del mundo de los usuarios (o, para ser más precisos, de su mundo de negocios). La pregunta que debe hacerse al evaluar modelos de datos alternativos no es: “¿Este diseño representa exactamente al mundo real?”, sino “¿Este diseño representa exactamente el modelo del medio ambiente de los usuarios?” El objetivo es desarrollar un diseño que corresponda a la concepción mental de los usuarios.

De hecho, Immanuel Kant y otros filósofos declararon que es imposible que los seres humanos construyan modelos de lo que realmente existe, y afirmaron que la esencia de las cosas es eternamente desconocida para los humanos.⁸ Extendiendo estos argumentos a los sistemas computacionales, Winograd y Flores declararon que, en las sociedades, los humanos construyen sistemas de prueba que les permiten operar con éxito en el mundo. Un conjunto de pruebas no es un modelo al infinito de la realidad,

⁸ En su libro *Prolegómenos para cualquier metafísica futura*, Kant afirmó que no podemos ir más allá de toda experiencia posible, ni formar un concepto definitivo de lo que las cosas en sí mismas pueden ser. Declaró que aún no estamos en libertad de abstenernos completamente de investigarlas, porque la experiencia nunca satisface por completo a la razón; pero al contestar las preguntas, nos remitimos más y más al pasado y no quedamos satisfechos de la solución total. Immanuel Kant, *Prolegomena to Any Future Metaphysics* (Indianápolis: Bobbs-Merril, 1950), pág. 100.

sino, más bien, sólo un sistema social que permite a los usuarios coordinar sus actividades con éxito, y no se puede decir nada más.⁹

Por lo tanto, los sistemas de computación necesitan modelar y representar las comunicaciones de sus usuarios con otras personas. No modelan nada más que el sistema de pruebas y comunicaciones. Así que aprenda a preguntarse: “¿Este modelo refleja exactamente las percepciones de los usuarios y los modelos mentales de su mundo? ¿Los ayudará a responderse consistente y eficazmente el uno al otro, así como a sus clientes?” No tiene sentido hacer un análisis para reivindicar que su modelo es una mejor representación de la realidad. Lo importante es desarrollar uno que represente bien el modelo del medio ambiente de negocio del usuario.

► RESUMEN

El modelo entidad-relación fue desarrollado por Peter Chen. Con ese modelo, las entidades se definen como cosas identificables y de importancia para los usuarios. Todas las entidades de cierto tipo forman una clase de entidad. Una entidad particular se denomina instancia. Las entidades tienen atributos que describen sus características, y uno o más atributos identifican a una entidad.

Las relaciones son asociaciones entre entidades. El modelo E-R explícitamente define relaciones; cada relación tiene un nombre, y existen clases de relaciones así como entidades de relaciones. Las relaciones pueden tener atributos.

El grado de la relación es el número de entidades que participan en ésta. La mayoría son binarias. Los tres tipos de relaciones binarias son 1:1, 1:N y N:M.

En los diagramas entidad-relación las entidades se muestran en rectángulos y las relaciones en diamantes. La cardinalidad máxima se muestra dentro del diamante; la mínima se indica por medio de una línea cortada o de un óvalo. Las relaciones que conectan ejemplos de entidades de la misma clase son recursivas. Los atributos se pueden mostrar en un diagrama E-R en elipses, o en una tabla por separado.

Una entidad débil es la que depende de la existencia de otra; una entidad que no es débil se llama entidad fuerte. Las entidades débiles se muestran en rectángulos con las esquinas redondeadas, y la relación de la cual depende la entidad se indica por medio de un diamante con las esquinas redondeadas. En este texto definimos a una entidad débil como aquella que depende lógicamente de otra. Una entidad puede tener una cardinalidad mínima de 1 en una relación con otra entidad y no necesariamente es una entidad débil. Los atributos multivalor se representan con entidades débiles.

Algunas entidades tienen subtipos que definen subconjuntos de entidades similares. Los subtipos heredan atributos de su entidad padre, el supertipo. Las relaciones TIE-NE-UN conectan entidades de diferentes tipos y los identificadores de las entidades son diferentes. Las relaciones ES-UN son subtipos y los identificadores de las entidades son los mismos.

Cuando se ha desarrollado un modelo de datos, los diseñadores deben considerar las reglas del negocio que pueden limitar el procesamiento contra las entidades. Cada entidad en el modelo debería ser evaluada a la luz de posibles adiciones, cambios y eliminaciones de datos. En particular, con frecuencia las eliminaciones son fuente de importantes restricciones para el procesamiento. Cuando se descubren las reglas del negocio deben ser documentadas en el modelo de datos.

El modelo E-R es una parte importante de muchos productos CASE, los cuales proporcionan herramientas para la construcción y el almacenamiento de diagramas E-R. Algunas herramientas CASE integran las construcciones E-R con estructuras de datos en los repositorios CASE. El Lenguaje de modelación unificado (UML, por sus siglas en inglés) ha definido un nuevo estilo de diagramas entidad-relación. Le convendría familiarizarse con los diagramas de ese estilo; pero también debe darse cuenta de que cuan-

⁹ Terry Winograd y F. Flores, *Understanding Computers and Cognition* (Reading, MA: Addison-Wesley, 1986).

do se crea el diseño de una base de datos no hay una diferencia fundamental entre el estilo tradicional y el UML.

Una vez terminados, los modelos E-R deben ser evaluados. Una técnica es enumerar las consultas que se podrían responder utilizando el modelo de datos. Después, mostrar esta lista a los usuarios, a los cuales se les pedirá que piensen en preguntas adicionales. Entonces, se evalúa el diseño contra esas preguntas para asegurarse de que el modelo pueda responderlas.

Las bases de datos no modelan el mundo real, sino el modelo del mundo de negocios de los usuarios. El criterio apropiado para juzgar un modelo de datos es si corresponde o no al modelo de los usuarios. No tiene caso discutir cuál es el que se adapta mejor al mundo real.

► PREGUNTAS DEL GRUPO I

- 3.1 Defina *entidad* y dé un ejemplo.
- 3.2 Explique la diferencia entre una clase de entidad y una instancia de entidad.
- 3.3 Defina *atributo* y dé ejemplos para la entidad que describió en la pregunta 3.1.
- 3.4 Explique qué es un atributo compuesto y dé un ejemplo.
- 3.5 En su respuesta a la pregunta 3.3, ¿qué atributo identifica a la entidad?
- 3.6 Defina *relación* y dé un ejemplo.
- 3.7 Explique la diferencia entre una clase de relación y una entidad de relación.
- 3.8 Defina *grado de relación*. Proporcione un ejemplo diferente al de este texto con respecto a una relación con un grado mayor que 2.
- 3.9 Enumere y dé un ejemplo sobre los tres tipos de relaciones binarias. Dibuje un diagrama E-R para cada uno.
- 3.10 Defina los términos *cardinalidad máxima* y *cardinalidad mínima*.
- 3.11 Nombre y dibuje los símbolos utilizados en los diagramas entidad-relación para: (a) entidad, (b) relación, (c) entidad débil y su relación, (d) relación recursiva, y (e) entidad subtipo.
- 3.12 Mencione un ejemplo de diagrama E-R para las entidades DEPARTAMENTO y EMPLEADO, las cuales tengan una relación 1:N. Suponga que un DEPARTAMENTO no necesita tener algún EMPLEADO, pero que cada EMPLEADO debe tener un DEPARTAMENTO.
- 3.13 Dé un ejemplo referente a una relación recursiva y muéstrelo en un diagrama E-R.
- 3.14 Muestre ejemplos de atributos para DEPARTAMENTO y EMPLEADO (con base en la pregunta 3.12). Utilice los símbolos de estilo UML.
- 3.15 Defina el término *entidad débil* y proporcione un ejemplo sobre alguno que no se mencione en este libro.
- 3.16 Explique la ambigüedad en la definición del término *entidad débil*. Explique qué interpretación se le da a dicho término en este texto. Dé ejemplos, diferentes a los del libro, sobre cada tipo de entidad débil.
- 3.17 Defina el término *entidad dependiente de un identificador* (ID-dependiente) y proporcione algún otro ejemplo diferente a los que ya se mencionaron.
- 3.18 Muestre cómo usar una entidad débil para representar el atributo multivalor para Habilidad en una entidad EMPLEADO. Indique las cardinalidades máxima y mínima en ambos lados de la relación. Use los símbolos tradicionales.
- 3.19 Muestre cómo usar una entidad débil para representar el atributo compuesto multivalor Teléfono, que contiene los atributos de un solo valor CódigodeÁrea, NúmeroTelefónico. Suponga que Teléfono aparece en una entidad llamada VENDEDOR.

Indique las cardinalidades máxima y mínima en ambos lados de la relación. Utilice los símbolos de estilo UML.

- 3.20 Describa las entidades de subtipos y dé algún ejemplo diferente a los que están en este texto.
- 3.21 Explique el término *herencia* y muestre cómo se aplica a su respuesta de la pregunta 3.20.
- 3.22 Explique la diferencia entre una relación TIENE-UN y una relación ES-UN, y dé un ejemplo sobre cada una.
- 3.23 ¿Cómo son abordadas las reglas del negocio en un modelo E-R?
- 3.24 Describa por qué es importante evaluar un modelo de datos una vez que ha sido creado. Resuma una técnica para la evaluación de un modelo de datos y explique qué técnica se podría usar para evaluar el modelo de datos de la figura 3-21.

► PREGUNTAS DEL GRUPO II

- 3.25 Cambie el diagrama E-R en la figura 3-19 para incluir una entidad LECCIÓN. Permita a LECCIÓN PRIVADA y LECCIÓN EN GRUPO ser subtipos de LECCIÓN. Modifique las relaciones según sea necesario. Use los símbolos tradicionales.
- 3.26 Cambie el diagrama E-R en la figura 3-19 para excluir MAESTRO. Modifique la relación como sea necesario. Use los símbolos de estilo UML.
- 3.27 ¿Cuál de los modelos de la figura 3-19 y de sus respuestas a las preguntas 3.25 y 3.26 prefiere? Explique por qué.
- 3.28 Cambie el diagrama de la figura 3-21 para incluir los subtipos de equipo. Suponga que el equipo de Paseos en barco San Juan pertenece a ARRENDAMIENTO, y otro equipo a BARCO. Modele las diferencias entre BARCO relacionado con EQUIPO fijo de los barcos y el BARCO relacionado con EQUIPO que no sea fijo. ¿Qué beneficios agregan más complejidad a este modelo?

► PROYECTOS

A. Desarrolle un diagrama E-R para una base de datos que apoye las necesidades de seguimiento de la siguiente organización: la Agencia Metropolitana de Viviendas (AMV) es una organización no lucrativa que se dedica al desarrollo y mejoramiento de la vivienda de interés social. La AMV opera en un área metropolitana con aproximadamente 2.2 millones de habitantes.

La AMV conserva datos referentes a la ubicación, disponibilidad y condiciones de alojamiento con bajo costo en 11 diferentes zonas censadas en el área metropolitana. Dentro de los límites de estas zonas hay aproximadamente 250 edificios diferentes que proporcionan alojamiento barato. En promedio, cada edificio consta de 25 departamentos u otras unidades.

La AMV conserva datos acerca de cada zona censada, incluyendo los límites geográficos, ingresos medios de la población, servidores públicos, negocios destacados, principales inversionistas de conformidad con los atributos de esa zona, y otros datos demográficos y económicos. También mantiene una cantidad limitada de datos acerca de la criminalidad. Para cada edificio, la AMV almacena nombre, dirección, tamaño, nombre del o de los propietarios, dirección, nombre y dirección del o de los hipotecarios, renovaciones y reparaciones, y disponibilidad de instalaciones para personas discapacitadas. Además, la AMV conserva una lista por unidad dentro de cada edificio, incluyendo el tipo de unidad, tamaño, número de recámaras, número de baños, instalaciones en la cocina y el comedor, localización del edificio y cualquier observación especial. A la

AMV le gustaría conservar los datos referentes a las tasas promedio de ocupación por unidad, pero, a la fecha, no ha podido compilar o almacenar esos datos. Sin embargo, la AMV tiene información acerca de si una unidad está ocupada o no.

La AMV funciona como un centro de información de la vivienda y ofrece tres servicios básicos. Primero, trabaja con políticos, congresistas y grupos de abogados para apoyar la legislación que se encarga del desarrollo de la vivienda de interés social a través de incentivos fiscales, desarrollo de zonas preferenciales y otros incentivos legislativos. Para lograrlo, la AMV proporciona información acerca de la vivienda de bajo costo para los gobiernos estatales, del municipio y de la ciudad. Segundo, mediante discursos, seminarios, pláticas en convenciones y otras actividades de relaciones públicas, la AMV lucha por aumentar la conciencia de la comunidad acerca de la necesidad de que exista el alojamiento de interés social. Por último, la AMV proporciona información sobre la disponibilidad de la vivienda de interés social a otras agencias que trabajan con comunidades de bajos ingresos y con población sin hogar.



B. Ingrese al sitio Web de un fabricante de computadoras, por ejemplo Dell (www.dell.com). Use el sitio Web para determinar qué computadora portátil podría comprar para un usuario que tenga un presupuesto de \$10000. Cuando esté en el sitio Web, piense acerca de la estructura de una posible base de datos de sistemas y subsistemas de computación para apoyar ese sitio.

Desarrolle un diagrama E-R de una base de datos de sistemas y subsistemas computacionales para ese sitio Web. Muestre todas las entidades y relaciones cuando menos con dos o tres atributos por entidad. Indique las cardinalidades máximas y mínimas a ambos lados de cada relación. Las posibles entidades son SISTEMA-BASE, OPCIÓN-MEMORIA, TARJETA-DE-VÍDEO, e IMPRESORA. Por supuesto, hay una gran variedad de posibles entidades. Modele cualquier atributo multivalor como se muestra en el texto. Use los subtipos donde sea apropiado. Para evitar que este proyecto se extienda demasiado, limite el diseño a las necesidades de alguien que esté decidiendo una compra.



C. Ingrese al sitio Web de un vendedor de libros, por ejemplo Amazon (www.amazon.com). Use el sitio Web para determinar los tres mejores libros de XML (siglas en inglés de *Extended Markup Language*, o Lenguaje extendido de marcaje) para alguien que apenas está aprendiendo sobre el tema. Cuando visite el sitio Web piense en la estructura de una posible base de datos de libros, autores, temas y temas afines.

Desarrolle un diagrama E-R de una base de datos de libros para este sitio Web. Muestre todas las entidades y relaciones y cuando menos dos o tres atributos por entidad. Indique las cardinalidades máximas y mínimas para ambos lados de cada relación. Las posibles entidades son TÍTULO, AUTOR, EDITORIAL, COPIA y TEMA. Por supuesto, hay muchas más entidades posibles. Modele cualquier atributo multivalor como se muestra en el texto. Use subtipos donde sea apropiado. Para evitar que este proyecto se expanda demasiado, suponga que sólo le dará seguimiento a los libros. Además, limite su diseño a las necesidades de alguien que está buscando libros que quiere comprar. No considere pedido del cliente, entrega del pedido, orden de compra y otros procesos de negocios.

► PREGUNTAS DEL PROYECTO FIREDUP

Considere la situación de FiredUp que analizamos al final de los capítulos 1 y 2. Suponga que FiredUp ahora ha desarrollado una línea de tres estufas diferentes: FiredNow, FiredAlways y FiredAtCamp. Además, suponga que los propietarios están vendiendo refacciones de cada estufa y que también las reparan. Algunas reparaciones son gratuitas porque están dentro del periodo de garantía de la estufa; en otras sólo se cobran las refacciones, y en otras más se cobran las refacciones y la mano de obra. FiredUp quiere

tener el seguimiento de todos estos datos. Cuando se solicitaron mayores detalles a los propietarios, hicieron la siguiente lista:

CLIENTE: Nombre, Dirección, Calle, Número de Departamento, Ciudad, Estado/Provincia, CP, País, Correo Electrónico, Número Telefónico

ESTUFA: Número de Serie, Tipo, Fecha de Fabricación, Clave del Inspector

FACTURA: Número de Factura, Fecha, Cliente, con una lista de artículos y precios a los que fueron vendidos, Precio Total

REPARACIÓN: Número de Reparación, Cliente, Estufa, Descripción, con una lista de artículos que fueron utilizados en la reparación y el costo de éstos, cuando proceda, y Cantidad Total de la reparación

PARTE: Número, Descripción, Costo, Precio de Venta

A. Elabore un diagrama entidad-relación de una base de datos para FiredUp. Establezca las cardinalidades máxima y mínima de las relaciones entre entidades, según lo considere apropiado. Explique su razonamiento para cada valor de cardinalidad. Use las entidades débiles como lo juzgue conveniente. No use subtipos. Mencione cualquiera de las entidades dependientes de un identificador, si las hay.

B. Modifique su diagrama entidad-relación en su respuesta a la pregunta A mediante la representación de una FACTURA y REPARACIÓN con los subtipos apropiados. ¿Bajo qué circunstancias es mejor este diseño que el de la respuesta a la pregunta A?

C. Suponga que FiredUp quiere dar seguimiento a la casa, fax y números de teléfono celular, así como a múltiples direcciones de correo electrónico para cada uno de sus clientes. Modifique su diagrama E-R para permitir valores múltiples como Número Telefónico y Correo Electrónico.

D. Suponga que FiredUp desarrolla diferentes versiones del mismo producto de estufa. Desarrolle la versión 1 de FiredNow, la versión 2 de FiredNow, y así sucesivamente. Modifique su diagrama entidad-relación de la pregunta A anterior como sea necesario para considerar esta situación.

E. Cuando se les pregunta a los usuarios a qué datos quieren dar seguimiento, no necesariamente recuerdan todas sus necesidades. Usando sus conocimientos sobre pequeñas operaciones de negocios, haga una lista de las entidades que ellos puedan haber olvidado. Muestre el potencial de las relaciones entre estas entidades en un diagrama E-R. ¿Cómo determinaría qué datos adicionales de FiredUp se necesitan?

El modelo de objeto semántico

El presente capítulo analiza el modelo de objeto semántico, el cual, al igual que el modelo E-R que estudiamos en el capítulo 3, se utiliza para crear modelos de datos. Como se muestra en la figura 4-1, el equipo analista entrevista a los usuarios y analiza sus reportes, formas y consultas, y a partir de eso construye un modelo de los datos que proporcionan los usuarios. Este modelo de datos se transforma posteriormente en el diseño de una base de datos.

La forma particular del modelo de datos depende de las estructuras que se utilizaron para construirlo. Si se utilizó un modelo E-R, éste tendrá entidades, relaciones, etc. Si se empleó un modelo semántico, entonces tendrá objetos semánticos y construcciones relacionadas, las cuales analizaremos en este capítulo.

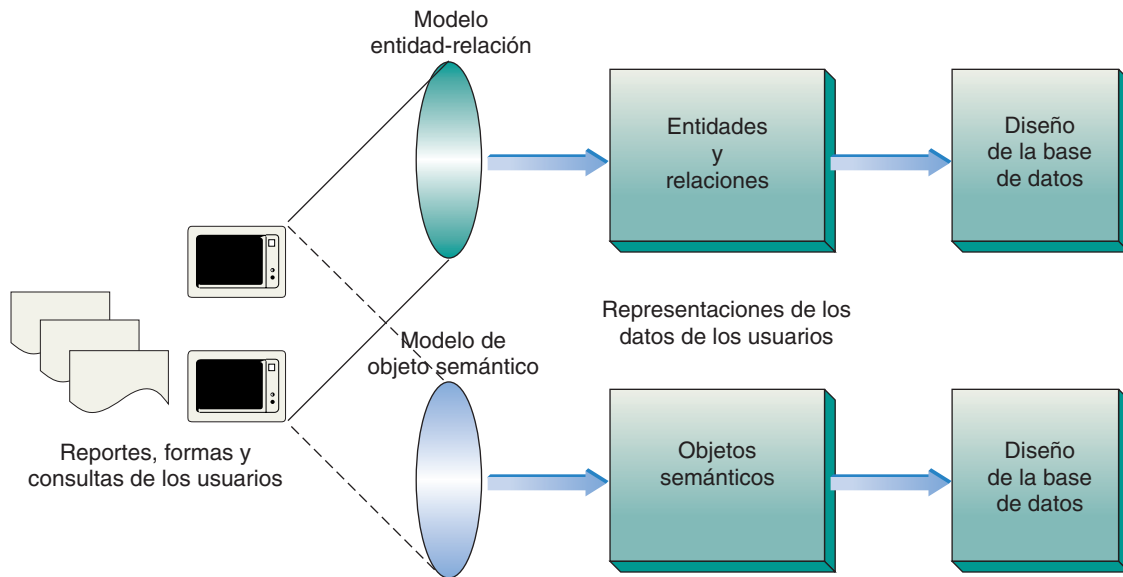
El modelo E-R y el modelo de objeto semántico son como lentes con los cuales los analistas de bases de datos observan mientras estudian y documentan los datos de los usuarios. Ambos lentes funcionan y finalmente dan como resultado el diseño de una base de datos. Los analistas utilizan distintos lentes para formar el diseño; sin embargo, y debido a que los lentes crean distintas imágenes, los diseños que producen no pueden ser exactamente los mismos. Cuando desarrolla una base de datos, usted debe determinar qué enfoque usar, de la misma manera en que un fotógrafo necesita decidir qué lentes utilizará. Cada enfoque tiene fortalezas y debilidades, las cuales analizaremos al final de este capítulo.

El modelo de objeto semántico se presentó por primera vez en 1988, en la tercera edición de este libro. Dicho modelo está basado en conceptos que desarrollaron y publicaron Codd y Hammer y McLeod.¹ El modelo de objeto semántico es un modelo de datos. Es diferente al **procesamiento de bases de datos orientadas a objetos**, el cual estudiaremos en el capítulo 18. Ahí usted aprenderá de qué manera los propósitos, las características y construcciones de la modelación de objeto semántico difieren del procesamiento de la base de datos orientada a objetos.

¹ E. F. Codd, "Extending the Relational Model to Capture More Meaning", *ACM Transactions on Database Systems*, diciembre de 1976, pp. 397-424; y Michael Hammer y Dennis McLeod, "Database Description with SDM: A Semantic Database Model", *ACM Transactions on Database Systems*, septiembre de 1981, pp. 351-386.

► FIGURA 4-1

Uso de diferentes modelos para el diseño de bases de datos



► OBJETOS SEMÁNTICOS

El propósito de una aplicación de una base de datos es proporcionar formas, reportes y consultas, de tal manera que los usuarios puedan dar seguimiento a entidades u objetos que son importantes para su trabajo. Las metas para las primeras etapas del desarrollo de la base de datos consisten en determinar las cosas que serán representadas en ésta, con el fin de especificar las características de dichas cosas y establecer las relaciones entre todas ellas.

En el capítulo 3 nos referimos a estas cosas como entidades. En el presente capítulo les llamaremos **objetos semánticos**, o algunas veces sólo objetos. La palabra *semántico* quiere decir significado, y un objeto semántico es aquél que modela, en parte, el significado de los datos de los usuarios. Los objetos semánticos modelan las percepciones de los usuarios con mayor precisión que el modelo E-R. Utilizamos el adjetivo *semántico* con la palabra *objeto* para distinguir los objetos analizados en este capítulo de los objetos definidos en los lenguajes de programación orientada a objetos (OOP).

DEFINICIÓN DE OBJETOS SEMÁNTICOS

Las entidades y los objetos son similares en ciertos aspectos y diferentes en otros. Comenzaremos con las similitudes. Un objeto semántico es una representación de algunas cosas identificables en el ambiente de trabajo de los usuarios. De manera más formal, un objeto semántico es un *conjunto de atributos que describen suficientemente una identidad bien definida*.

Al igual que las entidades, los objetos semánticos se agrupan en clases. Una clase de objetos tiene un *nombre* que la distingue de otras y que corresponde a los nombres de las cosas que representa. Por consiguiente, una base de datos que apoya a los usuarios en su trabajo con registros de estudiantes tiene una clase de objetos llamada ESTUDIANTE. Observe que los nombres de la clase de objetos, al igual que los nombres de la clase de entidades, se escriben con letras mayúsculas. Un objeto semántico en particular es una instancia de la clase. Por lo tanto, “William Jones” es una instancia de la clase ESTUDIANTE y “Contabilidad” es una instancia de la clase DEPARTAMENTO.

Como sucede con las entidades, un objeto tiene un *conjunto de atributos*. Cada atributo representa una característica de la entidad que se representa. Para fines de este ejemplo, el objeto ESTUDIANTE puede tener atributos como Nombre, DirecciónParticular,

Dirección del Campus, Fecha de Nacimiento, Fecha de Graduación, y Especialidad. Este conjunto de atributos también es una *descripción suficiente*, lo cual significa que los atributos representan todas las características que los usuarios necesitan para realizar su trabajo. Como establecimos al final del capítulo 3, las cosas en el mundo tienen un conjunto infinito de características; no es posible representarlas todas. En lugar de eso, representamos aquellas que son necesarias para que los usuarios satisfagan sus necesidades de información, de modo que puedan realizar con éxito su trabajo. La descripción suficiente también significa que los objetos son completos en sí mismos. Todos los datos que se requieren sobre un CLIENTE, por ejemplo, se localizan en el objeto CLIENTE, así que no necesitamos buscar en otro lado para encontrar esos datos.

Los objetos representan *entidades bien definidas*; esto es, son ciertas cosas que los usuarios reconocen como independientes y separadas a las que desean dar seguimiento, y a partir de eso elaborar reportes. Estas entidades son los nombres acerca de los cuales se va a producir la información. Para entender mejor el término *identidad bien definida*, recuerde la diferencia que hay entre los objetos y las instancias de los objetos. CLIENTE es el nombre de un objeto, y “CLIENTE 12345” es el nombre de una instancia de un objeto. Cuando decimos que un objeto representa una identidad bien definida, queremos decir que los usuarios consideran cada *instancia* de un objeto como única e identificable por derecho propio.

Finalmente, observe que las identidades que representan a los objetos pueden o no tener una existencia física. Por ejemplo, EMPLEADOS existe físicamente, pero PEDIDOS, no. Los PEDIDOS son, por sí mismos, modelos de un acuerdo contractual para proporcionar algunos bienes o servicios bajo ciertos términos y condiciones. No son objetos físicos, sino representaciones o acuerdos. Por consiguiente, alguna cosa no necesita ser física para que sea considerada como un objeto; sólo necesita ser identificable por derecho propio en la mente de los usuarios.

ATRIBUTOS

Los objetos semánticos tienen atributos que definen sus características. Existen tres tipos de atributos: los **atributos simples** que tienen sólo un elemento, como por ejemplo Fecha de Contratación, Número de Factura, Total de Ventas; los **atributos grupales**, los cuales son combinaciones de otros atributos, como por ejemplo Dirección, que contiene los atributos {Calle, Ciudad, Estado, Código Postal}, o también Nombre Completo, el cual contiene los atributos {Nombre, Inicial del Segundo Nombre, Apellido}, y los **atributos de los objetos semánticos**, que establecen una relación entre un objeto semántico y otro.

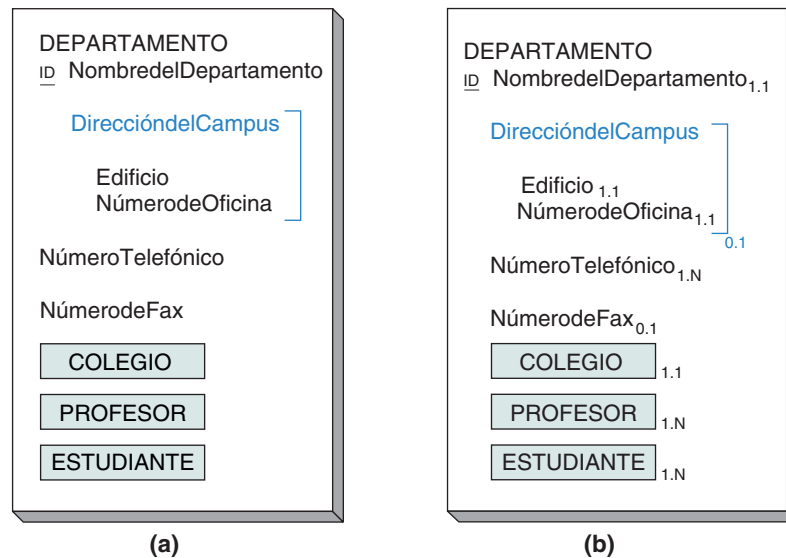
Para entender mejor estos enunciados observe la figura 4-2(a), la cual es un ejemplo de un **diagrama del objeto semántico**, o **diagrama del objeto**. Los equipos de analistas utilizan dichos diagramas para resumir las estructuras de los objetos y representarlos visualmente. Los objetos se muestran en rectángulos verticales. El nombre del objeto aparece en la parte superior y los atributos están escritos en orden después del nombre del objeto.

El objeto DEPARTAMENTO contiene un ejemplo de cada uno de los tres tipos de atributos. Nombre del Departamento, Número Telefónico y Número de Fax son atributos simples, y cada uno representa sólo un elemento de datos. Dirección del Campus es un atributo grupal que contiene los atributos simples Edificio y Número de Oficina. Finalmente, COLEGIO, PROFESOR y ESTUDIANTE son atributos del objeto semántico, lo cual significa que esos objetos están conectados a DEPARTAMENTO y lógicamente contenidos en éste.

Los atributos del objeto o, como se les llama a veces, **enlaces del objeto**, significan que cuando un usuario piensa en un DEPARTAMENTO no sólo estará considerando Nombre del Departamento, Dirección del Campus, Número Telefónico y Número de Fax, sino también COLEGIO, PROFESORES y ESTUDIANTES que se relacionan con dicho departamento. Ya que COLEGIO, PROFESOR y ESTUDIANTE son objetos, el modelo de datos completo también contiene diagramas de objetos para éstos. El objeto COLEGIO contiene atributos del colegio; el objeto PROFESOR tiene atributos del profesorado, y el objeto ESTUDIANTE posee atributos de los estudiantes.

► FIGURA 4-2

Diagrama del objeto DEPARTAMENTO:
(a) objeto DEPARTAMENTO y (b) objeto DEPARTAMENTO con cardinalidades



CARDINALIDAD DE LOS ATRIBUTOS. Cada atributo en un objeto semántico tiene una cardinalidad mínima y una cardinalidad máxima. La mínima indica la cantidad de instancias del atributo que deben existir para que el objeto sea válido; por lo general este número es 0 o 1. Si es 0 no se requiere que el atributo tenga un valor. Si es 1, debe tener un valor. Aunque es poco común, a veces la cardinalidad mínima puede ser mayor que 1. Por ejemplo, el atributo JUGADOR en un objeto llamado EQUIPO DE BÁSQUETBOL puede tener una cardinalidad mínima de 5, puesto que este es el número más pequeño de jugadores que se necesitan para formar un equipo.

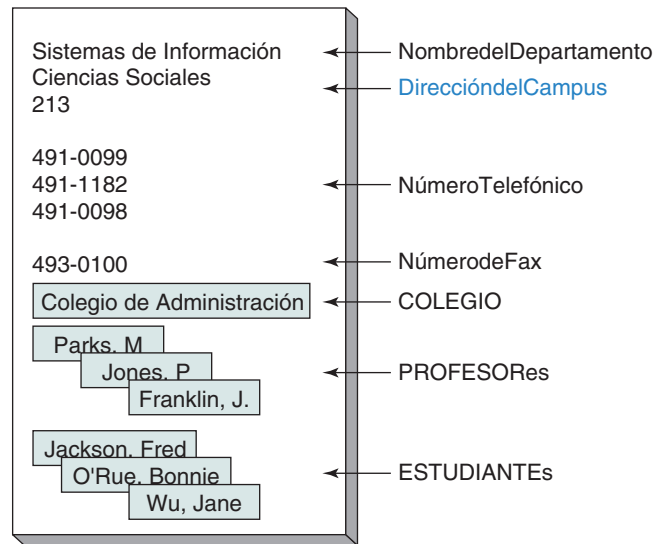
La cardinalidad máxima indica el número máximo de instancias del atributo que el objeto puede tener; por lo regular es 1 o N. Si es 1, el atributo no puede tener más de una instancia; si es N, el atributo puede tener muchos valores y el número absoluto no está especificado. Algunas veces la cardinalidad máxima es un número específico, como por ejemplo 5, lo cual significa que el objeto sólo puede contener exactamente cinco instancias del atributo. Por ejemplo, el atributo JUGADOR en EQUIPO DE BÁSQUETBOL puede tener una cardinalidad máxima de 15, lo cual indicaría que no se pueden asignar más de 15 jugadores a la lista de un equipo.

Las cardinalidades se muestran como subíndices de los atributos en el formato **N.M.**, donde N es la cardinalidad mínima y M la máxima. En la figura 4-2(b), la cardinalidad mínima de NombredelDepartamento es 1, y la máxima también es 1, lo cual significa que se requiere exactamente un valor de NombredelDepartamento. La cardinalidad de NúmeroTelefónico es 1.N, y esto significa que se requiere un DEPARTAMENTO para tener cuando menos un NúmeroTelefónico, aunque puede tener muchos. La cardinalidad de 0.1 en NúmerodeFax quiere decir que un DEPARTAMENTO puede tener cero o un NúmerodeFax.

Las cardinalidades de los grupos y sus atributos pueden ser muy sutiles. Considere el atributo DireccióndelCampus. Sus cardinalidades son 0.1, lo cual significa que un DEPARTAMENTO no necesita tener una dirección, o cuando mucho tener una. Ahora, examine los atributos dentro de DireccióndelCampus. Tanto Edificio como NúmerodeOficina tienen las cardinalidades 1.1. Probablemente usted se está preguntando cómo un grupo puede ser opcional si en dicho grupo se requieren los atributos. La respuesta es que las cardinalidades operan sólo entre el atributo y el poseedor de éste. La cardinalidad mínima de DireccióndelCampus indica que no necesita existir un valor para la dirección en DEPARTAMENTO. Pero las cardinalidades mínimas de Edificio y NúmerodeOficina indican que tanto Edificio como NúmerodeOficina deben existir en DireccióndelCampus. Por lo tanto, no necesita aparecer un grupo de DireccióndelCampus, pero si aparece uno debe tener un valor tanto para Edificio como para NúmerodeOficina.

► FIGURA 4-3

Una instancia del objeto DEPARTAMENTO en la figura 4-2



INSTANCIAS DE OBJETOS. Los diagramas de los objetos para DEPARTAMENTO que se muestran en la figura 4-2 son un formato, o una estructura general, que puede utilizarse para cualquier departamento. Una instancia del objeto DEPARTAMENTO se muestra en la figura 4-3, con cada valor del atributo para un departamento en particular. El NombredelDepartamento es Sistemas de Información y se localiza en la Oficina 213 del Edificio de Ciencias Sociales. Observe que existen tres valores para NúmeroTeléfono, debido a que el Departamento de Sistemas de Información tiene tres líneas telefónicas en esa oficina. Otros departamentos pueden tener menos o más, pero cada uno tiene cuando menos una.

Además, existe una instancia de COLEGIO, el Colegio de Administración, y existen valores múltiples para los atributos de los objetos PROFESOR y ESTUDIANTE. Cada uno de estos atributos de los objetos es un objeto completo y tiene todos los atributos definidos de un objeto de tal tipo. Para que este diagrama siga siendo simple sólo se muestran los nombres de identificación en cada una de las instancias del atributo del objeto.

Un diagrama del objeto es una representación de la percepción del usuario con respecto a un objeto de su ambiente de trabajo. Por lo tanto, en la mente del usuario el objeto DEPARTAMENTO incluye todos estos datos. Un DEPARTAMENTO lógicamente contiene datos sobre el COLEGIO en el que reside, así como de los PROFESORes y ESTUDIANTEs que se relacionan con dicho departamento.

ATRIBUTOS PAREADOS. El modelo de objeto semántico no tiene ninguna relación de objetos independientes en un solo sentido. Si un objeto contiene a otro, el segundo abarcará al primero. Por ejemplo, si DEPARTAMENTO contiene el atributo de objeto COLEGIO, entonces COLEGIO contendrá al atributo de objeto asociado DEPARTAMENTO. Estos atributos de objetos son denominados **atributos pareados**, puesto que siempre existen en pareja.

¿Por qué deben ser pareados los atributos de los objetos? La respuesta radica en la manera en que los seres humanos consideran las relaciones. Si el Objeto A tiene una relación con el Objeto B, entonces el Objeto B tendrá una relación con el Objeto A. Por lo menos, B se relaciona con A en la relación de “cosas que se relacionan con B”. Si este argumento no le resulta claro, trate de imaginarse una relación independiente entre dos objetos. Eso es imposible.

IDENTIFICADORES DE OBJETOS

Un **identificador de objetos** es uno o más de los atributos de los objetos que los usuarios emplean para identificar las instancias de objetos. Dichos identificadores son nombres potenciales para un objeto semántico. Por ejemplo, en CLIENTE, los posibles identificadores son IdentificacióndelCliente y NombredelCliente. Cada uno es un atri-

buto que los usuarios consideran como nombre válido de las instancias CLIENTE. Compare estos identificadores con atributos como Fecha del Primer Pedido, Precio de la Mercancía y Número de Empleados. Dichos atributos no son identificadores porque los usuarios no piensan en ellos como nombres de las instancias CLIENTE.

Un **identificador de grupo** es aquel que tiene más de un atributo. Algunos ejemplos son {Nombre, Apellido}, {Nombre, Número Telefónico} y {Estado, Número de Licencia}.

Los identificadores de grupo pueden ser únicos o no, dependiendo de cómo vean sus datos los usuarios. Por ejemplo, Número de Factura es un identificador único de PEDIDO, pero Nombre del Estudiante no es un identificador único de ESTUDIANTE. Por ejemplo, puede haber dos estudiantes llamadas “Mary Smith”. Si es así, los usuarios emplearán el Nombre del Estudiante para identificar a un grupo de uno o más estudiantes y entonces, si es necesario, utilizarán valores de otros atributos para identificar a un miembro de dicho conjunto en particular.

En los diagramas de objeto semántico, los identificadores de objetos están representados por las letras *ID* junto al atributo. Si el identificador es único, estas letras aparecerán subrayadas. Por ejemplo, en la figura 4-2(b), el atributo Nombre del Departamento es un identificador único de DEPARTAMENTO.

Normalmente, si se va a utilizar un atributo como identificador se requerirá su valor. Asimismo, por lo general no existe más que un valor de un atributo identificador para un objeto determinado. Por lo tanto, en la mayoría de las instancias la cardinalidad de un atributo de identificación (ID) es 1.1, y por consiguiente usaremos este valor como predeterminado.

Sin embargo, hay casos (relativamente pocos), en los cuales la cardinalidad de un identificador es diferente a 1.1. Por ejemplo, considere el atributo Alias (apodo) en el objeto semántico PERSONA. Una persona no necesita tener un alias, o probablemente él o ella tengan diversos alias. De ahí que la cardinalidad de Alias sería 0.N.

Al mostrar los subíndices de todos los atributos se desordenaría el diagrama de objeto semántico. Para simplificar, supondremos que las cardinalidades de los atributos identificadores con valores simples son 1.1 y las cardinalidades de otros atributos con valores simples son 0.1. Si la cardinalidad del atributo con valor simple es diferente a estas suposiciones, lo mostraremos en el diagrama; de lo contrario, omitiremos los subíndices de los atributos con valores simples.

DOMINIOS DE ATRIBUTOS

El **dominio** de un atributo es una descripción de los posibles valores de éste. Las características de un dominio dependen del tipo de atributo. El dominio de un atributo simple consiste tanto en una descripción física como en una descripción semántica. La descripción física indica el tipo de datos (por ejemplo, numéricos contra cadenas), su longitud y otras restricciones o condiciones (por ejemplo, que el primer carácter deba ser alfabético, o que el valor no exceda de 9999.99). La descripción semántica indica la función o el propósito del atributo; esto lo distingue de otros atributos que puedan tener la misma descripción física.

Por ejemplo, el dominio de Nombre del Departamento puede definirse como “el conjunto de cadenas con más de siete caracteres que representan los nombres de los departamentos de la Universidad Highline”. La frase *cadenas con más de siete caracteres* es la descripción física del dominio, y la frase *que representa los nombres de los departamentos de la Universidad Highline* es la descripción semántica. La descripción semántica diferencia a las cadenas de siete caracteres que representan los nombres de los departamentos, de las cadenas similares que representan, por así decirlo, nombres de cursos, edificios o algún otro atributo.

En algunos casos, la descripción física del dominio de un atributo simple es una **lista enumerada**, es decir, el conjunto de valores específicos de un atributo. Por ejemplo, el dominio del atributo Color podría ser la lista enumerada {“Azul”, “Amarillo”, “Rojo”}.

El dominio de un atributo grupal también tiene descripciones física y semántica. La primera es una lista de todos los atributos en el grupo y el orden que ocupan. La descrip-

ción semántica es la función o el propósito del grupo. De esta manera, la descripción física de dominio `DireccióndelCampus` (en la figura 4-2) es la lista {Edificio, NúmerodeOficina}; la descripción semántica es *la localización de una oficina en la Universidad Highline*.

El dominio del atributo de un objeto es el conjunto de las instancias del objeto de este tipo. Por ejemplo, en la figura 4-2 el dominio del atributo del objeto PROFESOR es el conjunto de todos los ejemplos del objeto PROFESOR en la base de datos. El dominio del objeto COLEGIO es el conjunto de todos los COLEGIOS en la base de datos. En cierta forma, el dominio del atributo de un objeto es una lista dinámicamente enumerada, la cual contiene todas las instancias de determinado tipo de objeto.

VISTAS DE OBJETOS SEMÁNTICOS

Los usuarios tienen acceso a los valores de los atributos del objeto a través de las aplicaciones de la base de datos que proporciona formas de entrada de datos, reportes y consultas. En la mayoría de las instancias dichas formas, reportes y consultas no requieren que se tenga acceso a todos los atributos de un objeto. Por ejemplo, la figura 4-4 muestra dos vistas de aplicación de DEPARTAMENTO. Algunos atributos de DEPARTAMENTO (por ejemplo, `NombredelDepartamento`) son visibles en ambas vistas de aplicación, otros atributos sólo son visibles en una vista. Por ejemplo, ESTUDIANTE sólo es visible en la vista de `ListadeEstudiantes`; pero PROFESOR, sólo en la Vista de Personal.

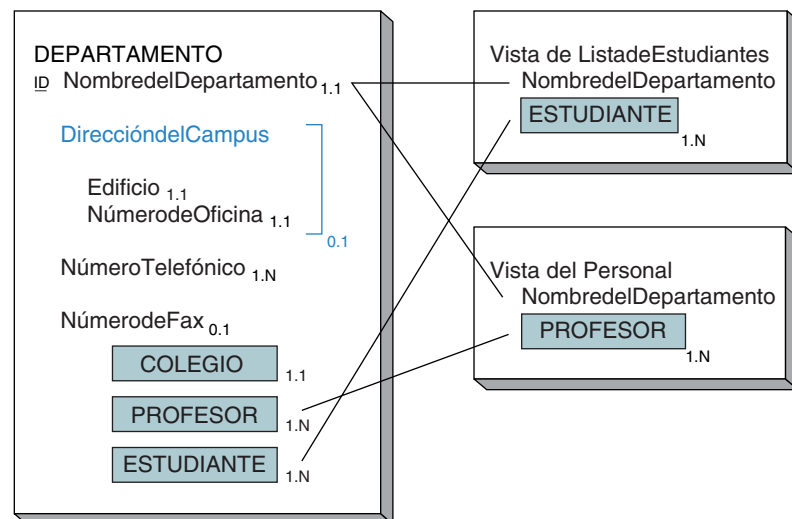
La parte de un objeto que está visible en una aplicación en particular se denomina **vista del objeto semántico**, o simplemente **vista**. Consta del nombre del objeto más una lista de todos los atributos visibles.

Las vistas se utilizan de dos maneras. Cuando usted crea una base de datos, puede emplearlas para desarrollar el modelo de datos. Observe nuevamente la figura 4-1. Cuando crean el modelo de datos, los analistas de la base de datos y de las aplicaciones trabajan de manera retrospectiva; esto es, comienzan con las formas, reportes y consultas que los usuarios dicen necesitar, y por lo tanto trabajan de manera inversa en el diseño de la base de datos. Para hacer esto, el equipo selecciona una forma, reporte o consulta y determina la vista que debe haber para que éstos se creen. Posteriormente, el equipo selecciona la siguiente forma, reporte o consulta y hace lo mismo. Entonces se integran las dos vistas. Este proceso se repite hasta que la estructura de toda la base de datos haya sido creada.

La segunda manera en la que se utilizan las vistas se presenta después de que la estructura de la base de datos ha sido creada. En este punto, se construyen vistas para dar soporte a las nuevas formas, reportes y consultas basadas en la estructura de la base de datos. Algunos ejemplos de este segundo uso los veremos en la Parte IV en la que analizaremos el SQL Server y Oracle.

► FIGURA 4-4

Vistas de ListadeEstudiantes y de Personal del objeto semántico DEPARTAMENTO



► CREACIÓN DE MODELOS DE DATOS CON OBJETOS SEMÁNTICOS

Esta sección ilustra un proceso para desarrollar objetos semánticos, en el cual los analistas examinan la interfaz de aplicación —formas, reportes y consultas— y trabajan de manera retrospectiva (ingeniería de reversa) con el fin de derivar la estructura del objeto. Por ejemplo, para modelar la estructura del objeto DEPARTAMENTO primero reunimos todos los reportes, formas y consultas basadas en DEPARTAMENTO. A partir de éstos, definimos el objeto DEPARTAMENTO el cual permite que se construyan dichos formatos y consultas.

Sin embargo, para una aplicación totalmente nueva no se examinarán reportes, formas o consultas computacionales. En este caso, los analistas comienzan por determinar a qué objetos necesitan dar seguimiento los usuarios. Posteriormente, mediante entrevistas con ellos, el equipo determina qué atributos del objeto son importantes. A partir de ahí, se pueden construir prototipos de formas o reportes que después se utilizarán para perfeccionar el modelo de datos.

UN EJEMPLO: BASE DE DATOS ADMINISTRATIVA DE LA UNIVERSIDAD HIGHLINE

Suponga que la administración de la Universidad Highline quiere dar seguimiento a los datos referentes a departamento, facultad y especialidad de cada estudiante. Además, suponga que la aplicación necesita producir cuatro reportes (figuras 4-5, 4-7, 4-9 y 4-11). Nuestro objetivo es examinarlos y mediante ingeniería de reversa determinar los objetos y atributos que deben ser almacenados en la base de datos.

EL OBJETO COLEGIO. El ejemplo de reporte en la figura 4-5 se refiere específicamente a un colegio: el de Administración. Éste es sólo un ejemplo del reporte; la Universidad Highline tiene reportes similares de otros colegios, como el de Artes y Ciencias y el Colegio de Ciencias Sociales. Cuando se crea un modelo de datos es importante reunir suficientes ejemplos para formar uno que sea representativo de todos los reportes. Aquí suponemos que el de la figura 4-5 es representativo.

Al examinar el reporte, encontramos datos específicos del colegio, tales como nombre, rector, número de teléfono y dirección del campus, así como información sobre cada uno de los departamentos. Esto *sugiere* que la base de datos puede contener los objetos COLEGIO y DEPARTAMENTO, con una relación entre ambos.

Esta información preliminar está documentada en los diagramas de los objetos en la figura 4-6. Observe que hemos omitido las cardinalidades de los atributos simples, los cuales tienen una cardinalidad de 0.1.

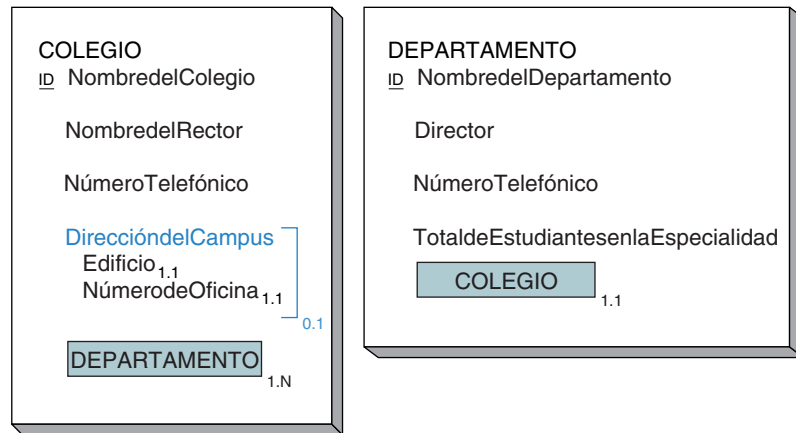
► FIGURA 4-5

Ejemplo del reporte
COLEGIO

Colegio de Administración Rector, Mary B. Jefferson			
Número Telefónico: 232-1187		Dirección del Campus: Edificio de Administración, Oficina 100	
Departamento	Director	Número Telefónico	Total de estudiantes en la especialidad
Contabilidad	Jackson, Seymour P.	232-1841	318
Finanzas	HeuTeng, Susan	232-1414	211
Sistemas de información	Brammer, Nathaniel D.	236-0011	247
Administración	Tuttle, Christine A.	236-9988	184
Producción	Barnes, Jack T.	236-1184	212

FIGURA 4-6

Primera versión de los objetos COLEGIO y DEPARTAMENTO



La cardinalidad de DEPARTAMENTO dentro de COLEGIO es 1.N, lo cual indica que un COLEGIO debe tener cuando menos un DEPARTAMENTO y que puede tener varios. Esta cardinalidad mínima no puede deducirse a partir del reporte en la figura 4-5; por el contrario, se preguntó a los usuarios si podía existir un colegio sin departamentos, y su respuesta fue que no.

Asimismo, observe que la estructura de DEPARTAMENTO se infiere de los datos que se muestran en la figura 4-5. Ya que los atributos de los objetos siempre están pareados, COLEGIO se muestra en DEPARTAMENTO; sin embargo, estrictamente hablando, este hecho no puede determinarse a partir de la figura 4-5. Igual que con el atributo DEPARTAMENTO en COLEGIO, se pidió a los usuarios que determinaran las cardinalidades de ese atributo, que son 1.1, lo cual significa que un DEPARTAMENTO se debe relacionar únicamente con un COLEGIO.

Paralelamente, hemos interpretado el reporte en la figura 4-5 para indicar que los grupos de datos repetitivos se refieren a DEPARTAMENTO como objeto independiente. De hecho, a menudo estos grupos repetitivos son una señal de que existe otro objeto. *Sin embargo, éste no es siempre el caso.* El grupo repetitivo puede ser también un atributo grupal que se presenta para tener diversos valores.

Quizás usted se pregunte cuál es la diferencia entre los datos de objetos repetitivos y los de grupos repetitivos. No existe ninguna regla inflexible y automática, debido a que la respuesta depende de la manera en que los usuarios ven su mundo. Consecuentemente, el mejor enfoque es consultar a los usuarios sobre la semántica de los datos. Pregunte si estos datos de grupos repetitivos son sólo una parte del colegio, o si se refieren a algo más que existe por sí mismo. Si se trata del primer caso, constituyen un atributo grupal; pero si se trata del segundo caso, entonces estamos hablando de un objeto semántico. También busque otros reportes (o formas, o consultas). ¿Los usuarios tienen uno para los departamentos? Si es así, entonces se confirmaría la suposición de que DEPARTAMENTO es un objeto semántico. De hecho, el personal de Highline utiliza dos reportes referentes a DEPARTAMENTOS. Este hecho comprueba más adelante la noción de que debe definirse un objeto DEPARTAMENTO.

Asimismo, los grupos de atributos que representan un objeto independiente por lo general tienen atributos de identificación obvios. Los automóviles tienen un NúmerodeIdentificación o NúmerodeLicencia; los productos tienen un NúmerodeProducto o SKU. Los pedidos tienen un NúmerodePedido. Sin embargo, el grupo de atributos (Fecha de Medición, Presión de los Neumáticos) no tiene un identificador obvio. Cuando le pregunte a un usuario sobre dicho identificador de grupo, le responderá algo como: “¿presión de los neumáticos de qué?” Puede ser la presión de los neumáticos de un auto, de un autobús, de un remolque o de algún otro vehículo. Por lo tanto, dicho grupo sería un atributo grupal dentro de otro objeto —el cual constituye la respuesta a la pregunta “¿de qué?”

EL OBJETO DEPARTAMENTO. El reporte DEPARTAMENTO que se muestra en la figura 4-7 contiene datos departamentales, junto con una lista de los profesores que es-

► FIGURA 4-7

Ejemplo de reporte de
DEPARTAMENTO

Departamento de Sistemas de Información Colegio de Administración		
Director:	Brammer, Nathaniel D.	
Teléfono:	236-0011	
Dirección del Campus:	Edificio de Ciencias Sociales, Oficina 213	
Profesor	Oficina	Teléfono
Jones, Paul D.	Ciencias Sociales, 219	232-7713
Parks, Mary B.	Ciencias Sociales, 308	232-5791
Wu, Elizabeth	Ciencias Sociales, 207	232-9112

tán asignados a dichos departamentos. Observe que este reporte contiene datos referentes a la dirección del campus del departamento. Puesto que estos datos no aparecen en el objeto en la figura 4-6, necesitamos agregarlos al objeto DEPARTAMENTO, como se hizo en la figura 4-8. Este ajuste es común en el procesamiento de modelos de datos. Esto es, los objetos semánticos se ajustan continuamente como reportes, formas y consultas nuevos que serán identificados y analizados.

EL OBJETO PROFESOR. El reporte en la figura 4-7 no sólo indica que el objeto DEPARTAMENTO necesita ser modelado, también sugiere que se puede necesitar otro objeto para representar los datos del profesor. Por lo tanto, se le agregó al modelo un objeto PROFESOR, como se muestra en la figura 4-8. La identificación de PROFESOR, que es NombredelProfesor, no es única; esto se denota con el hecho de no subrayar ID en la figura 4-8.

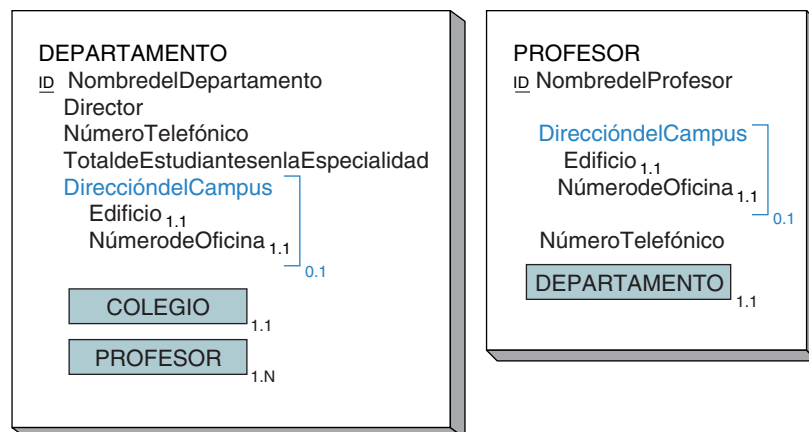
De acuerdo con los diagramas de objetos que se muestran en la figura 4-8, un DEPARTAMENTO debe tener cuando menos un PROFESOR y puede tener muchos profesores; pero un PROFESOR debe tener sólo un DEPARTAMENTO. Por consiguiente, de acuerdo con este modelo, se prohíben los nombramientos adjuntos. Esta restricción es parte de las reglas del negocio que se deben obtener de las entrevistas con los usuarios.

La figura 4-9 muestra un segundo reporte sobre un departamento, el cual se refiere a éste y a los estudiantes que cursan determinada especialidad. Es común tener dos reportes sobre un objeto, porque documentan puntos de vista diferentes sobre una misma cosa. Además, la existencia de este segundo reporte refuerza la noción de que departamento es un objeto en la mente de los usuarios.

EL OBJETO ESTUDIANTE. El reporte de la figura 4-9 proporciona datos sobre estudiantes que se especializan en una área del departamento, lo que implica que estos es-

► FIGURA 4-8

Objetos ajustados
DEPARTAMENTO y
NUEVO PROFESOR



► FIGURA 4-9

Segundo ejemplo de
reporte de
DEPARTAMENTO

Lista de estudiantes de especialidad Departamento de Sistemas de Información		
Director: Brammer, Nathaniel D.	Teléfono: 236-0011	
NombredelEstudiante de Especialidad	Número de Estudiante	Teléfono
Jackson, Robin R.	12345	237-8713
Lincoln, Fred J.	48127	237-8713
Madison, Janice A.	37512	237-8713

tudiantes también son un objeto. El objeto DEPARTAMENTO debe contener un objeto ESTUDIANTE y un objeto PROFESOR, como se observa en la figura 4-10.

El objeto ESTUDIANTE en la figura 4-10 contiene los atributos NombredelEstudiante, Número de Estudiante y Número Telefónico, los cuales son los atributos listados en el reporte de la figura 4-9. Observe que NombredelEstudiante y Número de Estudiante son identificadores. Los atributos NombredelEstudiante no son únicos, pero los de Número de Estudiante sí lo son.

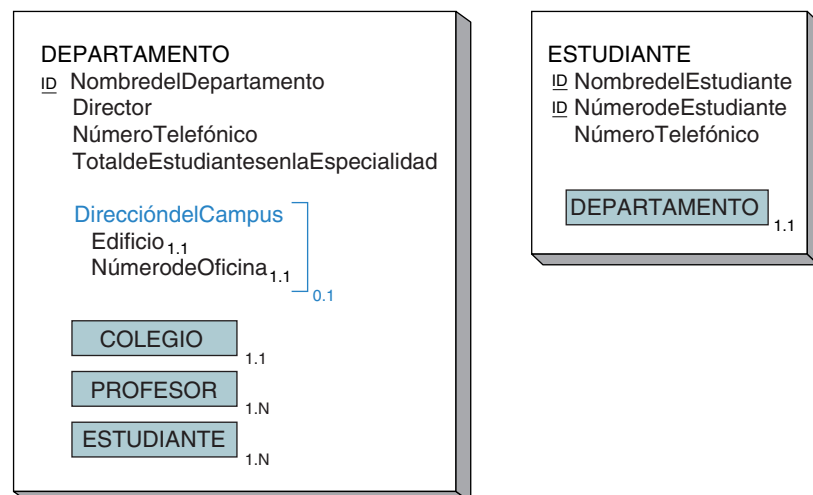
La figura 4-11 es un ejemplo de otro reporte sobre un estudiante —la carta de aceptación que la Universidad Highline envía a los aspirantes a ingresar—. Aunque es una carta, también es un reporte; probablemente la produjeron mediante una lista de correo con un procesador de textos.

Estos elementos de datos en la carta, que se deben almacenar en la base de datos, se muestran con letras negritas. Además de los datos referentes al estudiante, la carta también contiene datos sobre el DEPARTAMENTO de su especialidad y de su asesor. Puesto que un asesor es un PROFESOR, esta carta comprueba la necesidad de tener por separado un objeto PROFESOR. Los diagramas de los objetos revisados PROFESOR y ESTUDIANTE se muestran en la figura 4-12. De acuerdo con el objeto ESTUDIANTE, tanto DEPARTAMENTO como PROFESOR tienen un solo valor (una cardinalidad máxima de 1). Por tal razón, un estudiante tiene, cuando mucho, un departamento de especialidad y un asesor, y se requiere que tenga ambos.

El objeto ESTUDIANTE en la figura 4-12 corresponde a los datos que se muestran en la carta de la figura 4-11. Sin embargo, puede ser que el estudiante tenga en realidad más de una especialidad; en tal caso, DEPARTAMENTO y PROFESOR tendrían valores múltiples. Este hecho no puede determinarse sólo a partir de este formato de carta, por lo que se necesitan documentos y entrevistas adicionales para descubrir si se permiten varias especializaciones. Aquí supondremos que sólo se permite una.

► FIGURA 4-10

Objetos ajustados
DEPARTAMENTO y
nuevo ESTUDIANTE



► FIGURA 4-11

Carta de aceptación

Sr. Fred Parks
Calle Elm No. 123
Los Ángeles, California 98002

Estimado **Sr. Parks:**

Usted ha sido aceptado como estudiante de especialidad en el Departamento de **Contabilidad** de la Universidad Highline, a partir del **semestre de otoño del 2002**. La oficina del Departamento de Contabilidad se localiza en el Edificio de **Administración**, oficina **210**.

Su asesora es la profesora **Elizabeth Johnson**, cuyo número telefónico es el **232-8740**; su oficina es la número **227** y se localiza en el edificio de **Administración**. Por favor haga una cita con ella tan pronto como llegue al campus.

¡Felicidades y bienvenido a la Universidad Highline!

Sinceramente,

Jan P. Smathers
 Presidente

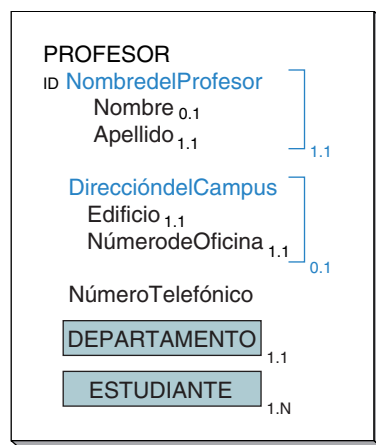
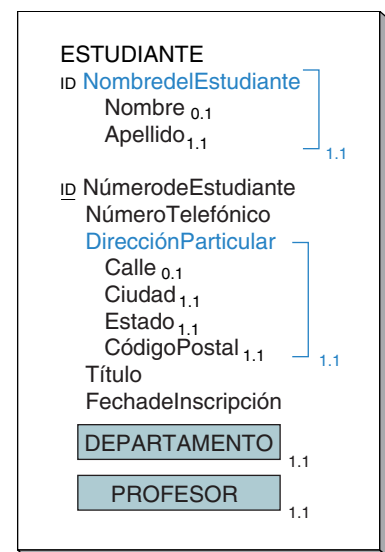
JPS/rkp

El formato de nombre del estudiante se proporciona primero en el de *nombre y apellido* en la parte superior de la carta, y después en el formato *apellido* en el saludo inicial. Si la presentación de los nombres en este formato es un requisito, entonces el atributo *NombredelEstudiante* (en la figura 4-10) no será suficiente y en su lugar se debe definir el grupo *Nombre, Apellido*. Esto se hizo en la figura 4-12. Asimismo observe que el nombre del asesor está en el formato *{Nombre, Apellido}*, lo cual significa que el nombre del PROFESOR debe cambiarse también.

Además, esta carta indica que los nombres en las direcciones y los saludos deben estar precedidos por el título “Sr.” o “Srita.” Si hay que hacer esto, debe colocarse un atributo adicional en ESTUDIANTE. Una alternativa es registrar el género del estu-

► FIGURA 4-12

*Objetos ajustados
 PROFESOR
 y ESTUDIANTE*



diente y seleccionar el título con base en este atributo. Otra alternativa es almacenar el título por sí mismo. La ventaja de esta segunda alternativa es que se pueden almacenar títulos diferentes a “Sr.” y “Srita.”, tales como “Dr.”, “Ing.”, etcétera.

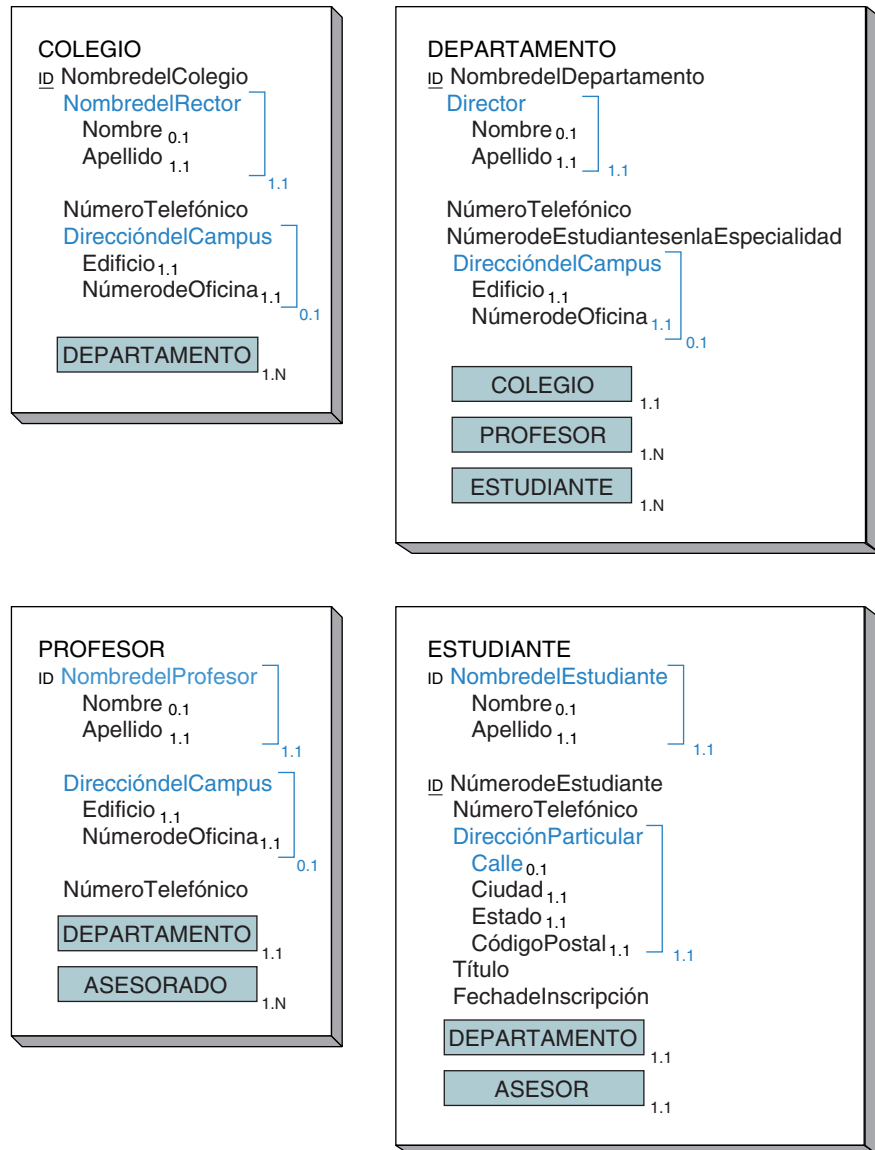
Como generalmente se documenta, el modelo no requiere un título diferente a Sr. o Srita. No obstante, es probable que se puedan necesitar títulos adicionales; por lo tanto, la segunda alternativa parece más convincente y por esta razón en la figura 4-2 se ha agregado el atributo Título a ESTUDIANTE.

De nuevo, estos cambios ilustran la naturaleza repetitiva del modelo de datos. Las decisiones del diseño con frecuencia necesitan ser reflexionadas y revisadas muchas veces. Tal iteración no significa que el proceso de diseño sea deficiente, lo cual de hecho es muy común.

ESPECIFICACIÓN DE OBJETOS

La figura 4-13 muestra los diagramas de los objetos completos para la base de datos de la Universidad Highline. Se han realizado pocos cambios: NombredelRector y Director se han modelado en el formato {Nombre, Apellido} de manera que todos los nombres

► FIGURA 4-13
Conjunto completo
de diagramas de
objetos semánticos



► FIGURA 4-14

Especificaciones de objetos para la base de datos de la Universidad
Highline:
 (a) especificaciones de objetos semánticos, y
 (b) especificaciones de dominio

Nombre del objeto	Nombre de la propiedad	Card. mín.	Card. máx.	Estatus del ID	Nombre del dominio
COLEGIO	NombredelColegio	1	1	<u>ID</u>	NombredelColegio
	NombredelRector	1	1		NombredelaPersona
	Nombre	0	1		Nombre
	Apellido	1	1		Apellido
	NúmeroTeléfono	0	1		NúmeroTeléfono
	DireccióndelCampus	0	1		DireccióndelCampus
	Edificio	1	1		Edificio
	NúmerodeOficina	1	1		NúmerodeOficina
DEPARTAMENTO	DEPARTAMENTO	1	N	<u>ID</u>	DEPARTAMENTO
	NombredelDepartamento	1	1		NombredelDepartamento
	Director	1	1		NombredelaPersona
	Nombre	0	1		Nombre
	Apellido	1	1		Apellido
	NúmeroTeléfono	0	1		NúmeroTeléfono
	TotaldeEstudiantesenla-	0	1		TotaldeAlumnosenla-
	Especialidad		1		Especialidad
	DireccióndelCampus	0	1		DireccióndelCampus
	Edificio	1	1		Edificio
	NúmerodeOficina	1	1		NúmerodeOficina
	COLEGIO	1	N		COLEGIO
PROFESOR	PROFESOR	1	N	ID	PROFESOR
	ESTUDIANTE	1	N		ESTUDIANTE
	NombredelProfesor	1	1		NombredelaPersona
	Nombre	0	1		Nombre
	Apellido	1	1		Apellido
	DireccióndelCampus	0	1		DireccióndelCampus
	Edificio	1	1		Edificio
ESTUDIANTE	NúmerodeOficina	1	1	<u>ID</u>	NúmerodeOficina
	NúmeroTeléfono	0	1		NúmeroTeléfono
	DEPARTAMENTO	1	1		DEPARTAMENTO
	ASESOR	1	N		ESTUDIANTE
	NombredelEstudiante	1	1		NombredelaPersona
	Nombre	0	1		Nombre
	Apellido	1	1		Apellido
ESTUDIANTE	NúmerodeEstudiante	1	1	<u>ID</u>	NúmerodeEstudiante
	NúmeroTeléfono	0	1		NúmeroTeléfono
	DirecciónParticular	1	1		Dirección
	Título	0	1		Título
	FechadelInscripción	0	1		FechadelTrimestre
	DEPARTAMENTO	1	1		DEPARTAMENTO
	ASESOR	1	1		PROFESOR

(a)

estén en un formato similar. Para mejorar la precisión del modelo, el atributo PROFESOR en ESTUDIANTE ha sido renombrado como ASESOR. La instancia PROFESOR que se conecta con una instancia ESTUDIANTE por medio de este atributo no es solamente cualquiera de los profesores del ESTUDIANTE, es un PROFESOR en particular que funge como asesor de ese ESTUDIANTE, por lo que el término ASESOR es más preciso que el de PROFESOR.

El dominio de este atributo es fijo. El dominio de ASESOR es PROFESOR, así como el dominio del atributo PROFESOR fue PROFESOR. Este atributo incluso señala o se conecta con las instancias del objeto semántico PROFESOR. El cambio de nombre es sólo eso: un mejoramiento en la especificación del papel que desempeña el dominio de PROFESOR en los objetos semánticos de ESTUDIANTE. Un cambio similar se hizo en PROFESOR. El atributo ESTUDIANTE fue renombrado como ASESORADO, pero este atributo aún se conecta con los objetos del dominio ESTUDIANTE.

► FIGURA 4-14

(Continuación)

Nombre del dominio	Tipo ^a	Descripción semántica	Descripción física
Dirección	G	Otra dirección	Calle Ciudad Estado CódigoPostal
Edificio	S	Nombre de un edificio en el campus	Texto 20
Dirección del Campus	G	Una dirección en el campus	Edificio Número de Oficina
Ciudad	S	Un nombre de ciudad	Texto 25
COLEGIO	OS	Uno de los diez colegios de la Universidad Highline	Véase la tabla de especificación del objeto semántico
Nombre del Colegio	S	Nombre oficial de un colegio en la Universidad Highline	Texto 25
DEPARTAMENTO	OS	Un departamento académico en el campus	Véase la tabla de especificación del objeto semántico
Nombre del Departamento	S	El nombre oficial de un departamento académico	Texto 25
Nombre	S	Parte del nombre de pila de Nombre de la Persona	Texto 20
Apellido	S	Parte del apellido del Nombre de la Persona	Texto 30
Total de Alumnos en la Especialidad	F	Total de los estudiantes asignados a un departamento específico	Entero, valores {de 0 a 999}; formato 999
Número de Oficina	S	El número de una oficina en el campus	Texto 4
Nombre de la Persona	G	Nombre y apellido de un administrador, profesor o estudiante	Nombre Apellido
Número Telefónico	S	Número Telefónico con código de área	Texto 4
PROFESOR	OS	El nombre de un miembro de tiempo completo de la facultad de la Universidad Highline	Véase la tabla de especificación del objeto semántico
Fecha del trimestre	S	Trimestre académico y año	Texto 3; los valores [q01, donde q = uno de {'F', 'W', 'S', 'M'} y 01 es el número decimal del 00 al 99]
Estado	S	La abreviación de dos dígitos de un estado	Texto 2
Calle	S	Dirección de una calle	Texto 30
ESTUDIANTE	OS	Una persona que ha sido aceptada para estudiar en la Universidad Highline	Véase la tabla de especificación del objeto semántico
Número de Estudiante	S	La ID asignada a un estudiante aceptado en la Universidad Highline	Entero; valores {de 10000 a 99999}; formato 99999
Título	S	El título de los individuos que se utilizarán en las direcciones	Texto 3; valores {Sr., Srita.}
Código Postal	S	Un código postal de nueve dígitos	Texto 10; formato 99999-9999

^aF = Fórmula
G = Grupo
S = Simple
OS = Objeto semántico

(b)

La figura 4-14 presenta una especificación tabular del modelo de datos. Los objetos semánticos y los atributos se definen en la especificación del objeto semántico, y los dominios en la especificación del dominio. La primera tabla es una presentación alternativa de la información en los diagramas de los objetos semánticos, y su interpretación es directa.

La segunda tabla, la de dominio, proporciona información sobre los dominios que no están disponibles en los diagramas de los objetos semánticos. Como establecimos anteriormente, un dominio tiene tanto una descripción semántica como una física. La descripción semántica de cada dominio se muestra en la columna Descripción y la descripción física se muestra en la columna Especificación. La columna de Descripción se explica por sí misma.

La especificación para los dominios incluye una descripción física y, en ciertos casos, un conjunto de valores y un formato. Por ejemplo, `NúmerodeEstudiante` se especifica como un entero con valores entre 10000 y 99999 y con un formato de cinco dígitos decimales. (En esta tabla, un 9 en una especificación del formato significa un dígito decimal.) Otros dominios se documentan de manera similar. `Título` es un ejemplo de un dominio enumerado cuyos valores para `Título` son Sr., o Srita. La descripción física de un dominio de grupo consiste en una lista de los dominios incluidos en el grupo. La descripción física de un objeto semántico es sólo una referencia de la descripción del objeto semántico.

El dominio de `EspecialidadesTotales` es un ejemplo de un cuarto tipo de dominio, el **dominio de fórmula**. Las fórmulas representan atributos computados a partir de otros valores. El dominio `TotaldeAlumnosEnlaEspecialidad` es el conteo de los objetos `ESTUDIANTE`, los cuales están conectados a un objeto `DEPARTAMENTO` determinado. No deberíamos tratar de documentar los medios por los cuales este cómputo se llevará a cabo en la definición del dominio. Al respecto, todo aquello que es importante documenta la necesidad y la especificación de la fórmula.

► TIPOS DE OBJETOS

Esta sección describe e ilustra siete tipos de objetos. Para cada uno, examinamos un reporte o forma y mostramos cómo modelar el de un objeto. Más adelante, en el capítulo 7, transformamos cada uno de esos tipos de objetos.

En esta sección se utilizan tres términos nuevos: **atributo de un valor** es el que tiene una cardinalidad máxima de 1; **atributo de valores múltiples** es aquel cuya cardinalidad máxima es mayor que 1; y **atributo que no es de objeto** es un atributo simple o grupal.

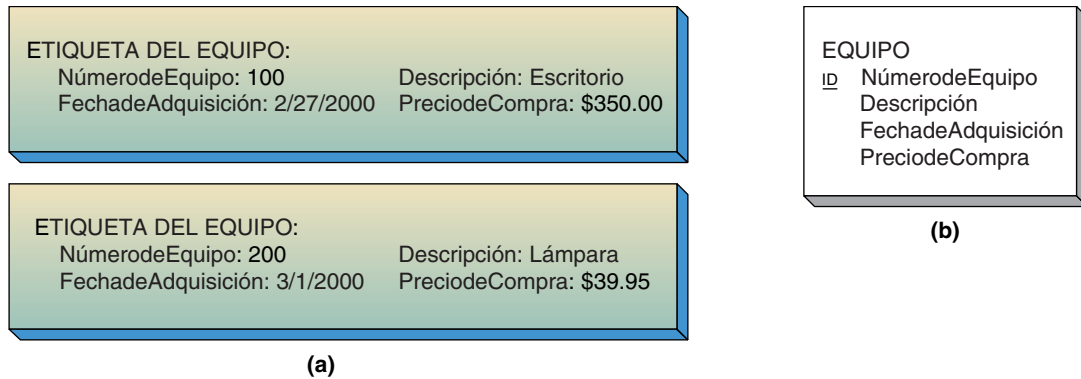
OBJETOS SIMPLES

Un **objeto simple** es un objeto semántico que contiene sólo un valor, atributos simples o grupales. En la figura 4-15 se muestra un ejemplo. El inciso (a) de esta figura muestra dos casos de un reporte llamado *Etiqueta de equipo*. Estas etiquetas se aplican a los artículos del equipo de oficina con el fin de facilitar el seguimiento del inventario. Dichas etiquetas pueden considerarse como un reporte.

La figura 4-15(b) muestra un objeto simple, `EQUIPO`, el cual modela la *Etiqueta de equipo*. Los atributos del objeto incluyen los artículos que se muestran en la etiqueta: `NúmerodeEquipo`, `Descripción`, `FechaDeAdquisición` y `PreciodeCompra`. Observe que ninguno de estos atributos tiene valores múltiples, y que ninguno es un atributo de objeto. Por lo tanto, `EQUIPO` es un objeto simple.

► FIGURA 4-15

Ejemplo de un objeto simple: (a) reportes basados en un objeto simple, y (b) el objeto simple EQUIPO



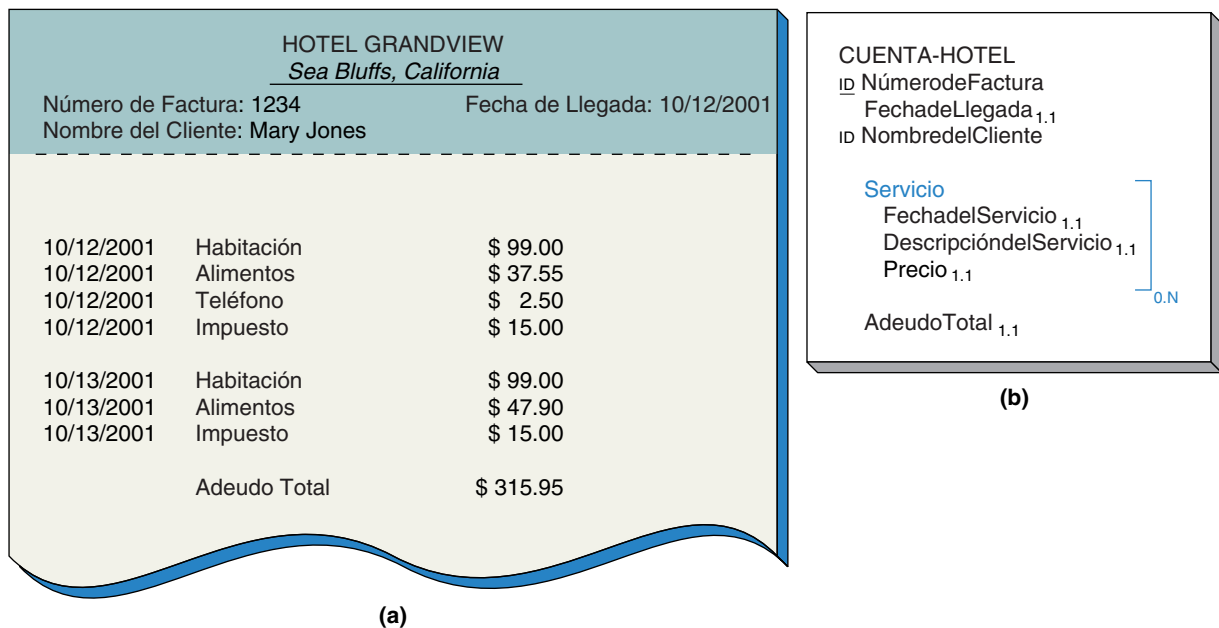
OBJETOS COMPUESTOS

Un **objeto compuesto** es un objeto semántico que contiene uno o más atributos de valores múltiples, simples o grupales, pero no atributos de objetos. La cuenta de hotel que se muestra en la figura 4-16(a) da origen a la necesidad de un objeto compuesto. Dicha cuenta incluye datos relacionados con ésta: Número de Factura, Fecha de Llegada, Nombre del Cliente y Adeudo Total. También contiene un grupo de atributos que se repite para los servicios proporcionados al cliente. Cada grupo incluye Fecha de Servicio, Descripción del Servicio y Precio.

La figura 4-16(b) muestra un diagrama de objeto para el objeto CUENTA-HOTEL. Servicio es un atributo de grupo que tiene una cardinalidad máxima de N, lo cual significa que el grupo Fecha de Servicio, Descripción del Servicio, Precio puede presentarse muchas veces en un ejemplo del objeto semántico CUENTA-HOTEL.

► FIGURA 4-16

Ejemplo de un objeto compuesto: (a) reporte basado en un objeto compuesto, y (b) objeto compuesto CUENTA-HOTEL



Servicio no se modeló como un objeto semántico independiente; por el contrario, se le consideró un atributo dentro de una CUENTA-HOTEL. Este diseño es conveniente debido a que no considera uno solo de los cargos de un huésped como algo independiente, y por ese motivo los servicios en la cuenta del huésped no tienen identificadores propios. Ningún empleado intenta introducir un Servicio, excepto en el contexto de una cuenta. El empleado introduce los datos para el número de cuenta 1234 y después, en el contexto de dicha cuenta, introduce los cargos, o cobra una cuenta e introduce los cargos adicionales que forman parte de ésta.

La cardinalidad mínima de Servicio es 0, lo cual significa que un objeto CUENTA-HOTEL puede existir sin ningún dato de Servicio. Esto permite que inicie una cuenta cuando el cliente se registra y antes de que haya ningún cargo. Si la cardinalidad mínima fuera 1, entonces no podría iniciarse ninguna CUENTA-HOTEL hasta que hubiera cuando menos un cargo. Esta decisión del diseño debe realizarse desde el punto de vista de las reglas del negocio. Puede ser que la política del hotel sea que no comience la cuenta hasta que haya un cargo. Si es así, entonces la cardinalidad mínima de Servicio debería ser 1.

Un objeto compuesto puede tener más de un atributo de valores múltiples. La figura 4-17(a) muestra una cuenta de hotel que tiene atributos de valores múltiples para NombredelCliente, así como un grupo de valores múltiples para los cargos de los servicios. Cada uno de estos grupos es independiente uno del otro. Por ejemplo, la segunda instancia de NombredelCliente no se asocia lógicamente con el segundo Servicio.

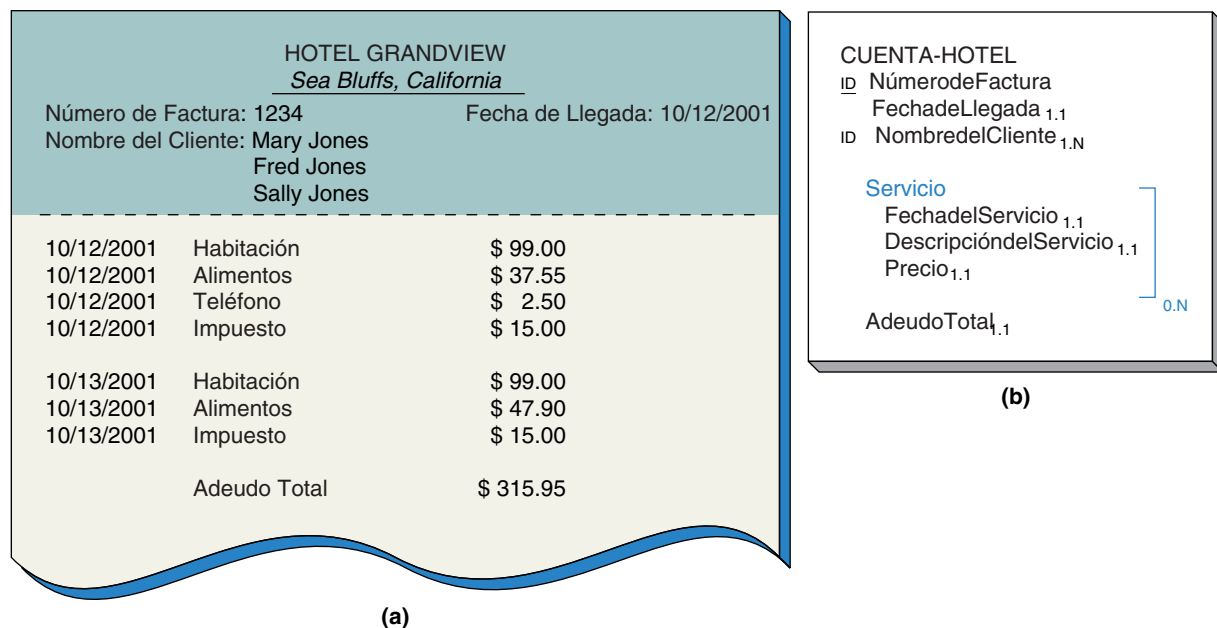
La figura 4-17(b) es un diagrama de objeto para la cuenta del hotel en la figura 4-17(a). NombredelCliente se muestra como un atributo de valores múltiples. No se incluye en el paréntesis de los cargos del servicio debido a que las repeticiones de NombredelCliente no tienen nada que ver con las repeticiones de los servicios. Como podemos observar, los dos son independientes.

Tanto los atributos simples como los grupales pueden ser de valores múltiples. Por ejemplo, en la figura 4-17(a), NombredelCliente es un atributo simple de valores múltiples. Por sí mismo, es suficiente para el objeto que es considerado compuesto.

Los atributos de valores múltiples pueden anidarse uno dentro del otro. Por ejemplo, suponga que es importante registrar los gastos individuales dentro de un Servicio.

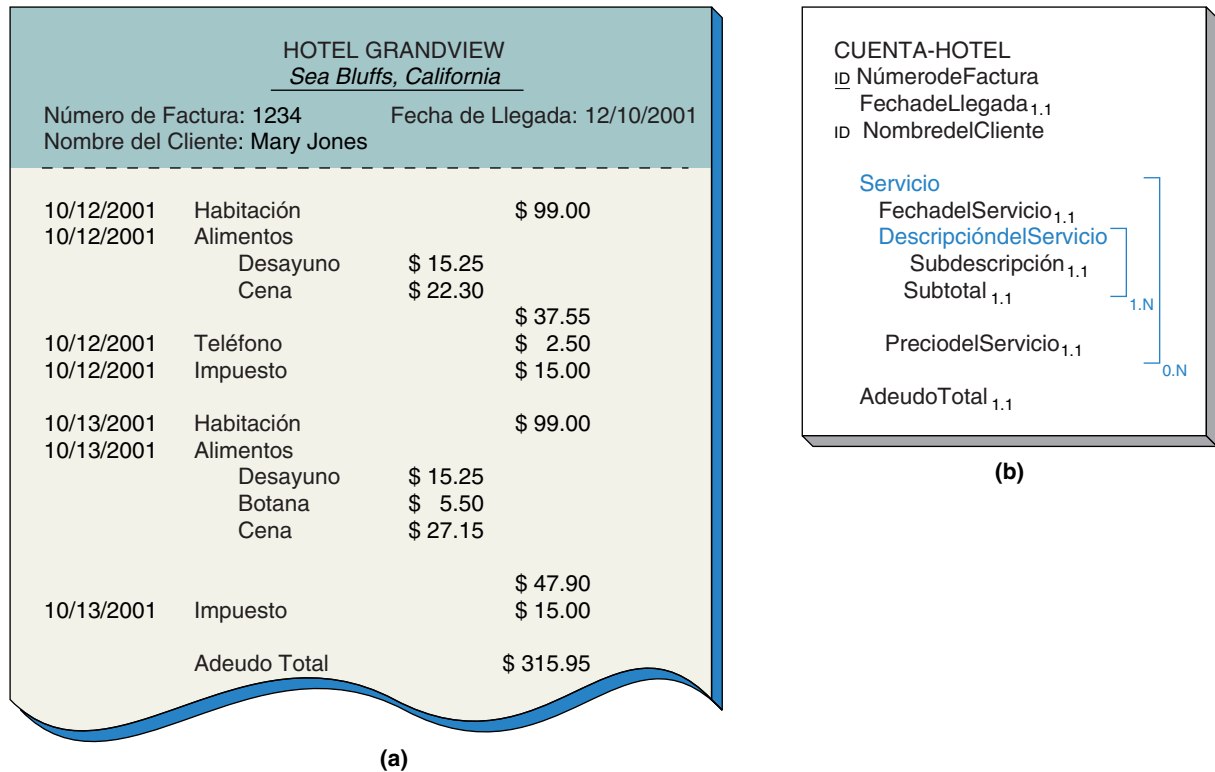
► FIGURA 4-17

Un objeto compuesto con dos grupos: (a) CUENTA-HOTEL con valores múltiples en Nombre del Cliente, y (b) CUENTA-HOTEL con dos grupos de valores múltiples



► FIGURA 4-18

Un objeto compuesto con grupos anidados: (a) CUENTA-HOTEL con subdescripciones de servicios, y
(b) CUENTA-HOTEL con grupos anidados de valores múltiples



En la forma de la figura 4-18(a) se subdividen los cargos. Por ejemplo, alimentos se desglosa por comidas. En la figura 4-18(b) se presenta un diagrama de objeto para estos atributos anidados y se muestran los cargos por servicios que tienen subelementos.

A manera de repaso, diremos que un objeto compuesto es el que contiene uno o más atributos simples de valor múltiple o atributos grupales. No tiene atributos de objeto.

OBJETOS COMBINADOS

Un **objeto combinado** contiene cuando menos un atributo de un objeto. La figura 4-19(a) muestra dos formas de ingreso de datos diferentes. Una de éstas, que utiliza el Departamento de Finanzas de la compañía, se emplea para llevar el registro de los vehículos. La segunda forma se usa para guardar los datos de los empleados. De acuerdo con estas formas, se asigna un vehículo a un empleado y un empleado tiene asignado cuando mucho un vehículo.

No podemos decir a partir de estas formas si un automóvil debe ser asignado a un empleado o si cada empleado debe tener uno. Para obtener dicha información, tendríamos que preguntar a los usuarios en los departamentos de finanzas o de recursos humanos. Suponga que descubrimos que un EMPLEADO no necesita tener un VEHÍCULO, pero que se debe asignar un VEHÍCULO a un empleado.

La figura 4-19(b) muestra los diagramas de objeto para EMPLEADO y VEHÍCULO. Un EMPLEADO tiene un VEHÍCULO como uno de sus atributos, y VEHÍCULO, a su vez, tiene EMPLEADO como uno de sus atributos. Puesto que tanto EMPLEADO como VEHÍCULO contienen atributos de objeto, ambos son objetos combinados. Además, debido a que ningún atributo es de valores múltiples, la relación de EMPLEADO con VEHÍCULO es uno a uno, o 1:1.

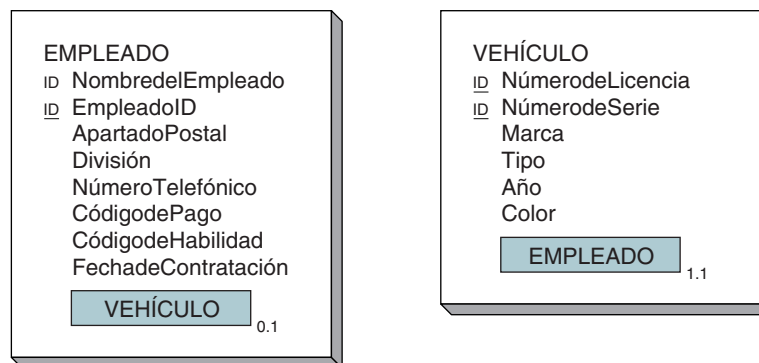
► FIGURA 4-19

Objetos combinados con propiedades pareadas 1:1:
 (a) ejemplo de las formas de ingreso de los datos de Vehículo y Empleado, y (b) objetos combinados EMPLEADO y VEHÍCULO

DATOS DEL VEHÍCULO			
Número de licencia	Número de serie		
Marca	Tipo	Año	Color
Asignado a empleado			

DATOS LABORALES DEL EMPLEADO			
Nombre del empleado		Identificación del empleado	
Apartado postal		División	Teléfono
Código de pago	Código de habilidad	Fecha de contratación	Automóvil asignado

(a)



(b)

En la figura 4-19(a) las formas de Empleado y Vehículo se interrelacionan. Es decir, los DATOS DEL VEHÍCULO tienen el campo de asignación del Empleado, y los DATOS LABORALES DEL EMPLEADO tienen el campo del Automóvil asignado. Pero éste no siempre es el caso; a veces la relación puede aparecer sólo en una dirección. Considere el reporte y la forma de la figura 4-20(b), los cuales se refieren a dos objetos: DORMITORIO y ESTUDIANTE. En el Reporte de ocupación del dormitorio, podemos apreciar que los usuarios piensan en un dormitorio conforme a los atributos correspondientes a dormitorio (Dormitorio, AsistenteResidente, NúmeroTelefónico), y también en los atributos correspondientes a los estudiantes (NombredelEstudiante, NúmerodeEstudiante, Clase) que viven en el dormitorio.

Por otra parte, la forma datos del estudiante muestra sólo los datos de éste; no incluye ningún dato acerca del dormitorio. La dirección del campus puede contener la dirección de un dormitorio, pero si es así, aparentemente no es tan importante como para documentarlo en el formato. En el proyecto de desarrollo de una base de datos esta posibilidad puede comprobarse mediante una entrevista con los usuarios. Aquí supondremos que la forma datos del estudiante no incluye los datos del dormitorio.

Como establecimos anteriormente, los atributos del objeto siempre se presentan pareados. Incluso si las formas, reportes y consultas indican que sólo se puede apreciar una parte de la relación, siempre existen ambas partes de ésta. Por analogía, un puente que conecta dos islas toca a ambas y puede ser utilizado en las dos direcciones, incluso si el puente es, por costumbre o por ley, de un solo sentido.

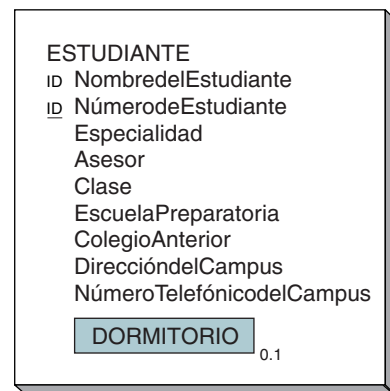
Cuando no puede encontrarse ningún reporte o ninguna forma para documentar una parte de una relación, el equipo analista puede preguntarle a los usuarios acerca de la cardinalidad de dicha relación. En este caso, el equipo necesitaría saber cuántos DORMITORIOS puede tener un ESTUDIANTE y si un ESTUDIANTE tiene que estar relacionado con un DORMITORIO. Aquí suponemos que las respuestas a estas preguntas son que un ESTUDIANTE está relacionado sólo con un DORMITORIO y que puede no

► FIGURA 4-20

Objetos combinados con las propiedades pareadas 1:N:
(a) ejemplo de Reporte de ocupación del dormitorio y de la Forma de datos del estudiante (Student Data Form), y
(b) objetos combinados DORMITORIO y ESTUDIANTE

REPORTE DE OCUPACIÓN DE DORMITORIO		
<u>Dormitorio</u>	<u>Asistente Residente</u>	<u>Teléfono</u>
Ingersoll	Sarah y Allen French	3-5567
<u>Nombre del estudiante</u>	<u>Número de estudiante</u>	<u>Clase</u>
Adams, Elizabeth	710	OS
Baker, Rex	104	FR
Baker, Brydie	744	JN
Charles, Stewart	319	OS
Scott, Sally	447	OS
Taylor, Lynne	810	FR

(a)



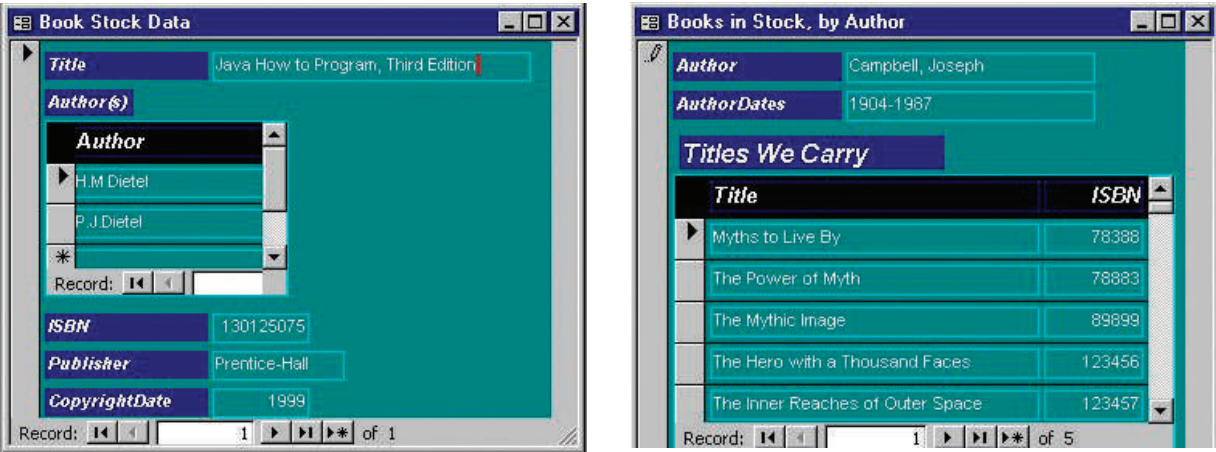
(b)

estar relacionado con ningún DORMITORIO. Así, en la figura 4-20(b) DORMITORIO contiene valores múltiples de ESTUDIANTE y ESTUDIANTE contiene un valor de DORMITORIO; por consiguiente, la relación de DORMITORIO con ESTUDIANTE es de uno a muchos, o 1:N.

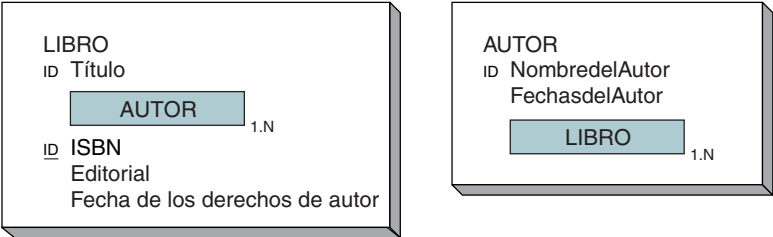
Una tercera ilustración de los objetos combinados aparece en la figura 4-21(a). A partir de estas dos formas podemos deducir que varios autores pueden escribir un libro (esto deriva de la forma Book Stock Data) y que un autor puede escribir varios libros (proviene de Book in Stock, by Author). Por consiguiente, en la figura 4-21(b) el objeto LIBRO contiene muchos valores de AUTOR, y AUTOR contiene varios valores de LIBRO. De ahí que la relación entre LIBRO y AUTOR sea de muchos a muchos, o

► FIGURA 4-21

Objetos combinados con propiedades pareadas N:M: (a) forma Ingreso de datos en una librería (Bookstore Data Entry Form), y (b) objetos LIBRO y AUTOR



(a)



(b)

N:M. Además, un LIBRO debe tener un AUTOR, y un AUTOR (para serlo) debe haber escrito cuando lo menos un LIBRO. De este modo, ambos objetos tienen una cardinalidad mínima de uno.

La figura 4-22 resume los cuatro tipos de objetos compuestos. En general, OBJETO1 puede contener un máximo de uno o varios OBJETOs2. De manera similar, OBJETO2 puede contener uno o varios OBJETOs1. Usaremos esta tabla para analizar el diseño de una base de datos en el capítulo 7.

OBJETOS HÍBRIDOS

Los **objetos híbridos** son combinaciones de objetos combinados y compuestos. En particular, un objeto híbrido es un objeto semántico que cuando menos tiene un atributo grupal de valores múltiples e incluye un atributo del objeto semántico.

La figura 4-23(a) es una segunda versión del reporte sobre la ocupación del dormitorio que se muestra en la figura 4-20(a). La diferencia radica en que la tercera columna de los datos del estudiante contiene Renta, en lugar de Clase. Ésta es una diferencia importante puesto que renta no es un atributo del ESTUDIANTE, sino que pertenece a la combinación ESTUDIANTE y DORMITORIO y es un atributo de DORMITORIO.

► FIGURA 4-22

Cuatro tipos de objetos compuestos

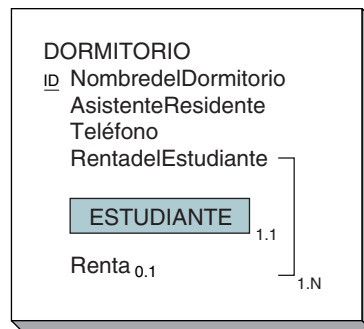
El objeto1 puede contener		El objeto2 puede contener	
	Uno	Muchos	
Uno	1:1	1:N	
Muchos	M:1	M:N	

FIGURA 4-23

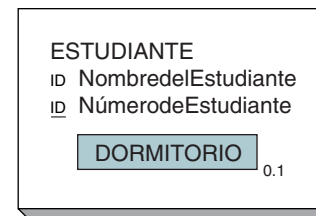
Objeto híbrido
DORMITORIO:
(a) Reporte
de ocupación de
dormitorio con la
propiedad Renta;
(b) objetos
DORMITORIO y
ESTUDIANTE
correctos, y
(c) objetos
DORMITORIO y
ESTUDIANTE
incorrectos

REPORTE DE OCUPACIÓN DE DORMITORIO		
Dormitorio	AsistenteResidente	Teléfono
Ingersoll	Sarah y Allen French	3-5567
NombredelEstudiante	NúmerodeEstudiante	Renta
Adams, Elizabeth	710	\$175.00
Baker, Rex	104	\$225.00
Baker, Brydie	744	\$175.00
Charles, Stewart	319	\$135.00
Scott, Sally	447	\$225.00
Taylor, Lynne	810	\$175.00

(a)



(b)



(c)

La figura 4-23(b) es un diagrama de objeto que modela esta forma. DORMITORIO contiene un grupo de valores múltiples que tienen el atributo del objeto ESTUDIANTE y el atributo del no objeto Renta. Esto significa que Renta se pareja con ESTUDIANTE en el contexto de DORMITORIO.

Ahora examine el objeto alternativo DORMITORIO en la figura 4-23(c). Éste es un modelo *incorrecto* del reporte en la figura 4-23(a), ya que muestra que Renta y ESTUDIANTE son independientemente valores múltiples, lo cual es incorrecto debido a que RENTA y ESTUDIANTE son valores múltiples como un par.

La figura 4-24(a) muestra una forma basada en otro objeto híbrido. La forma Sales Order Form contiene datos acerca de un pedido (Sales Order Number, Date, Subtotal, Tax, Total) (Número de Pedido, Fecha, Subtotal, Impuesto, Total), datos respecto a un CUSTOMER (CLIENTE) y un SALESPERSON (VENDEDOR), así como un grupo de valores múltiples que contienen por sí mismos datos acerca de los artículos del pedido. Además, los datos de ITEM, o ARTÍCULO (Item Number, Description, y Unit Price) (Número de Artículo, Descripción y Precio Unitario) aparecen dentro del grupo de valores múltiples.

La figura 4-24(b) muestra el objeto semántico PEDIDO. Éste contiene los atributos de los no objetos Número de Pedido, Fecha, Subtotal, Impuesto y Total (Sales Order Number, Date, Subtotal, Tax y Total.) Asimismo, contiene atributos de los objetos

► FIGURA 4-24

Objetos híbridos PEDIDO (SALES-ORDER) y objetos relacionados: (a) forma Pedido (Sales Order Form), y (b) objetos para modelar la forma Pedido

Carbon River Furniture Sales Order Form

Sales Order Number: 10643 Date: 25-Sep-01

Customer Name: Carbon River Bookshop

Address: 1145 Elm Street

City: Carbon River State: IL Zip: 02234

Phone: 232-0010

Salesperson Name: Dodsworth, Anne Salesperson Code: EZ-1

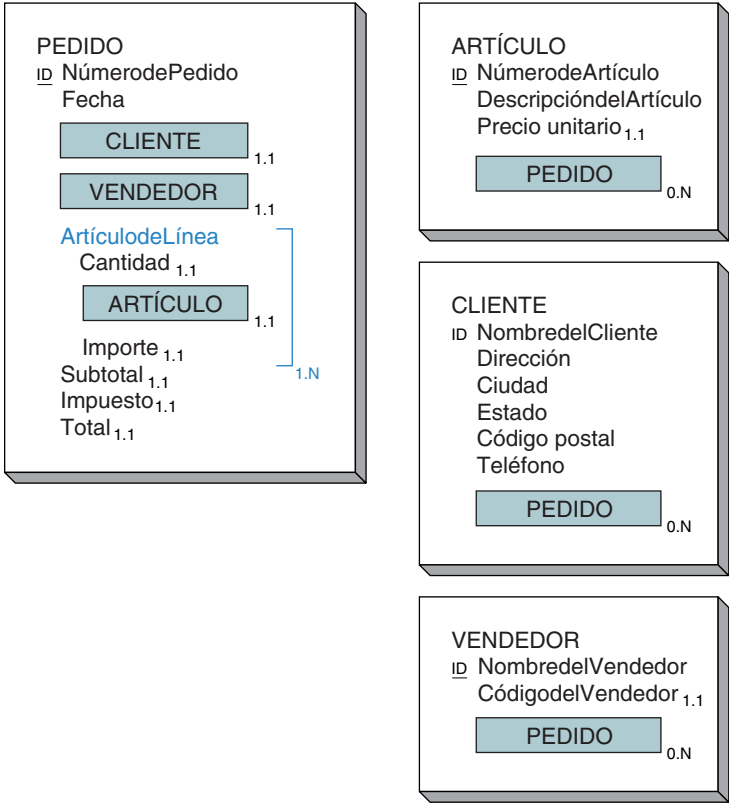
Quantity:	Item Number	Description	Unit Price:	Extended Price:
▶ 1	78	Executive Desk	\$959.00	\$959.00
1	79	Conference Table	\$1,750.00	\$1,750.00
4	80	Side Chair	\$99.00	\$396.00
*				

Subtotal: \$3,105.00

Tax: \$29.46

Total: \$3,134.46

(a)



(b)

CLIENTE (CUSTOMER) y VENDEDOR (SALESPERSON) y un grupo multivalor que representa cada artículo de línea en el pedido. El grupo contiene atributos de los no objetos Cantidad e Importe (Quantity y ExtendedPrice) y el atributo del objeto ARTÍCULO (ITEM).

Los diagramas de los objetos en la figura 4-24(b) son ambiguos en un aspecto que puede o no ser importante, dependiendo de la aplicación. De acuerdo con el diagrama del objeto ARTÍCULO, un ARTÍCULO puede conectarse con más de un PEDIDO. Pero debido a que el grupo de valores múltiples LíneadelArtículo está encapsulado (oculto) en este diagrama de PEDIDO no es claro si un ARTÍCULO puede presentarse *una o muchas veces* en el mismo PEDIDO.

En general, existen cuatro interpretaciones de la cardinalidad máxima de los atributos pareados en el objeto híbrido PEDIDO:

1. Un ARTÍCULO sólo puede aparecer en un PEDIDO y en sólo uno de ArticulodeLínea dentro de PEDIDO.
2. Un ARTÍCULO sólo puede aparecer en un PEDIDO, pero en varios ArticulodeLínea distintos dentro de dicho PEDIDO.
3. Un ARTÍCULO puede aparecer en varios PEDIDOS distintos pero sólo en ArticulodeLínea dentro de cada uno de esos PEDIDOS.
4. Un ARTÍCULO puede aparecer en distintos PEDIDOS y en diferentes ArticulodeLínea dentro de dichos PEDIDOS.

Cuando sea importante distinguir entre estas instancias debe utilizarse la siguiente notación: si el Caso 1 o el 2 están vigentes, la cardinalidad máxima del atributo del objeto híbrido debe terminarse como 1. Por lo tanto, en el caso de este ejemplo, la cardinalidad máxima de PEDIDO en ARTÍCULO se establece en 1. Si un ARTÍCULO sólo aparece en ArticulodeLínea de un PEDIDO (Caso 1), debe indicarse que tiene una Identificación única en dicho grupo. De lo contrario (Caso 2), no es necesario que se indique. Estas dos instancias se muestran en la figura 4-25(a) y (b).

Si los Casos 3 o 4 están vigentes, la cardinalidad máxima del atributo del objeto híbrido debe determinarse como N. Para este ejemplo la cardinalidad máxima de PEDIDO en ARTÍCULO se determina como N. Además, si un ARTÍCULO sólo aparece en un ArticulodeLínea de un PEDIDO (Caso 3), debe indicarse que tiene una Identificación única en dicho grupo. De lo contrario (Caso 4), no es necesario indicarlo. Estas dos instancias se muestran en la figura 4-25(c) y (d).

OBJETOS DE ASOCIACIÓN

Un **objeto de asociación** relaciona dos (o más) objetos y almacena datos que son peculiares para dicha relación. La figura 4-26(a) muestra un reporte y dos pantallas de ingreso de datos que originan la necesidad de un objeto de asociación. El reporte contiene datos acerca de los vuelos en una aerolínea, datos de un avión en particular, así como del piloto asignado a dicho vuelo. Las dos formas de ingreso de datos se refieren a un piloto y un avión.

En la figura 4-26(b), VUELO es un objeto de asociación que vincula a dos objetos: AVIÓN y PILOTO, y almacena datos sobre su asociación. VUELO contiene AVIÓN y PILOTO, pero AVIÓN y PILOTO contienen valores múltiples de VUELO. Este patrón particular de la asociación entre dos (o más) objetos con los datos ocurre con frecuencia, especialmente en las aplicaciones que involucran la asignación de dos o más cosas. Otros ejemplos son un TRABAJO que asigna un ARQUITECTO a un CLIENTE, una TAREA que asigna un EMPLEADO a un PROYECTO y un PEDIDO-DE-COMPRA que asigna un VENDEDOR a un SERVICIO.

Para el ejemplo de la figura 4-26, el objeto de asociación VUELO tiene un identificador propio: el grupo {NúmerodeVuelo, Fecha}. Con frecuencia los objetos de asociación no tienen identificadores propios, en tal caso el identificador es la combinación de los identificadores de los objetos que se asocian.

FIGURA 4-25

Ejemplos de los cuatro casos de cardinalidad máxima en un objeto híbrido: (a) ARTÍCULO en un PEDIDO, (b) ARTÍCULO en (posiblemente) varios Artículos de Línea de un PEDIDO, (c) ARTÍCULO en un Articulo de Línea (posiblemente) varios PEDIDOS, y (d) ARTÍCULO en (posiblemente) varios Articulos de Línea de (tal vez) varios PEDIDOS

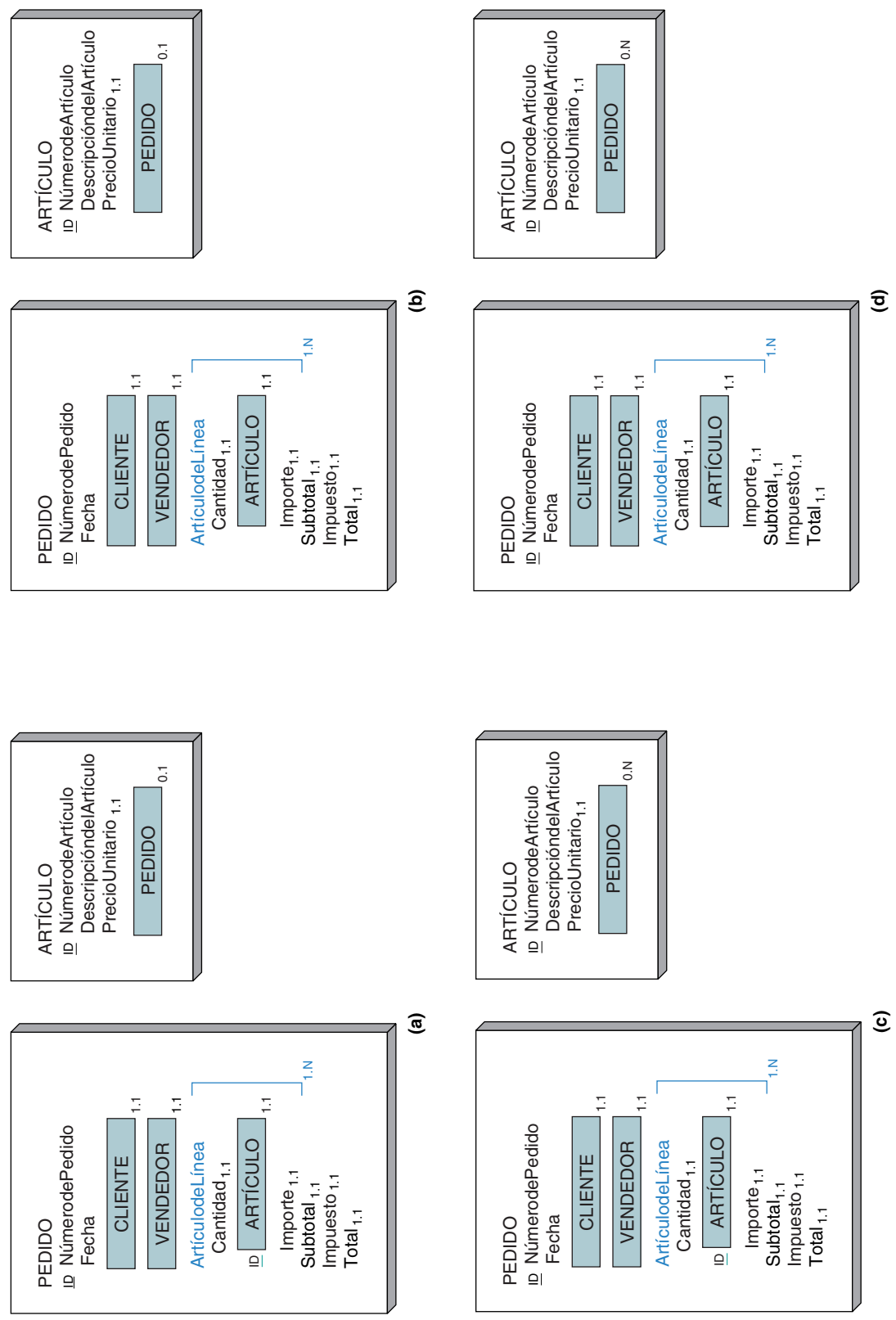


FIGURA 4-26

Ejemplos de un objeto de asociación: (a) ejemplos del reporte de vuelo y formatos, y (b) objetos VUELO, PILOTO, AVIÓN

COMPAÑÍA FLY CHEAP INTERNATIONAL			
Reporte de los datos de la planificación del vuelo			
NÚMERO DE VUELO	FC-17	FECHA	7/30/2001
CIUDAD DE ORIGEN	Seattle	DESTINO	Hong Kong
COMBUSTIBLE EN EL DESPEGUE			
PESO EN EL DESPEGUE			

AVIÓN			
Número de la cola	N1234FI		
Tipo	747-SP		
Capacidad	148		

PILOTO			
Nombre	Michael Nilson		
Identificación de vuelo	32887		
Horas de vuelo	18,348		

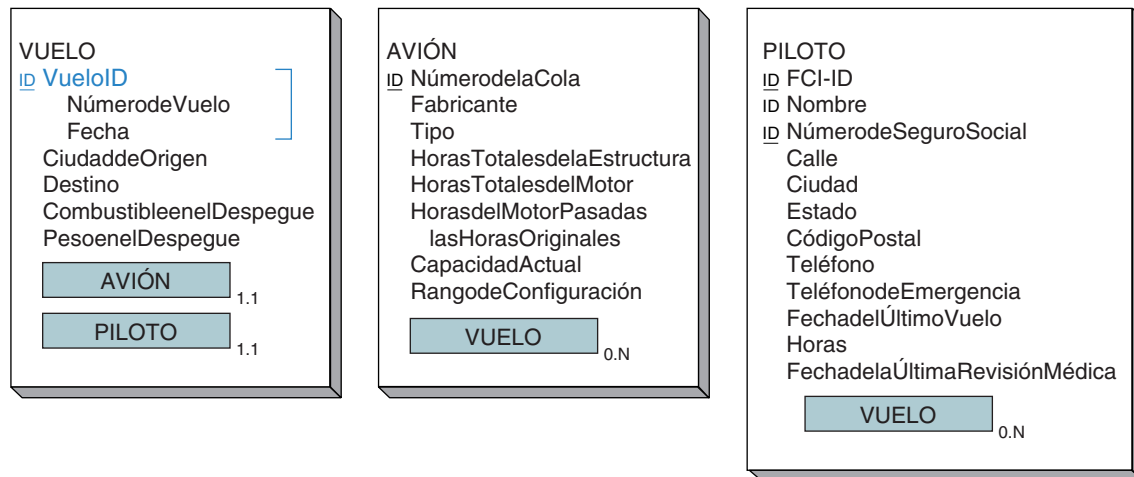
Fly Cheap International Pilot Summary Data Form			
FCI_ID	32887	Date Of Last Check Out	7/7/01
Name	Michael Nilson	Hours	18348
Social Security Number	000-45-0040	Date Of Last Physical	5/19/01
Address	1011 Western		
City	Denver		
State	CO		
Zip	80210		
Phone	555-55-5589		
Emergency Phone	555-33-9090		

Fly Cheap International Airplane Data Form	
Tail Number	N12324FI
Manufacturer	Boeing
Type	747-SP
Total Airframe Hours	11756
Total Engine Hours	11756
Engine Hours Since Overhaul	756
Current Capacity	148
Range As Configured	8900

(continúa)

► FIGURA 4-26

(Continuación)



(b)

Para comprender mejor esto, considere la figura 4-27(a), la cual muestra un reporte de la asignación de arquitectos a proyectos. Aunque ésta no tiene ningún identificador obvio, en realidad el identificador es la combinación de {Nombre del Proyecto, Nombre del Arquitecto}. Sin embargo, estos atributos pertenecen a PROYECTO y ARQUITECTO y no a ASIGNACIÓN. Por lo tanto, el identificador de ASIGNACIÓN es la combinación de dichos identificadores de las cosas que están asignadas.

La figura 4-27(b) muestra los diagramas de los objetos para este caso. PROYECTO y ARQUITECTO son atributos del objeto de ASIGNACIÓN, y el grupo {PROYECTO, ARQUITECTO} es el identificador de la ASIGNACIÓN. Esto significa que la combinación de un ejemplo de PROYECTO y un ejemplo de ARQUITECTO identifica una ASIGNACIÓN en particular.

Observe que el identificador de AsignaciónID en la figura 4-27(b) no es único, por lo tanto indica que un arquitecto puede ser asignado a un proyecto más de una vez. Si esto no es correcto, debe aclararse que el identificador no es único. También, si se puede asignar un empleado a un proyecto más de una vez, y si por alguna razón es importante que se tenga un identificador único para una ASIGNACIÓN, se debe agregar al grupo el atributo Fecha, o algún otro que indique el tiempo (Semana, Trimestre, etcétera).

OBJETOS PADRE-SUBTIPO

Para entender a los objetos padre y subtipo, considere al objeto EMPLEADO en la figura 4-28(a). Algunos de los atributos en EMPLEADO pertenecen a todos los empleados y otros sólo a los empleados que son administradores. El objeto en la figura 4-28(a) no es muy preciso debido a que los atributos orientados al administrador no son apropiados para los empleados que no son administradores.

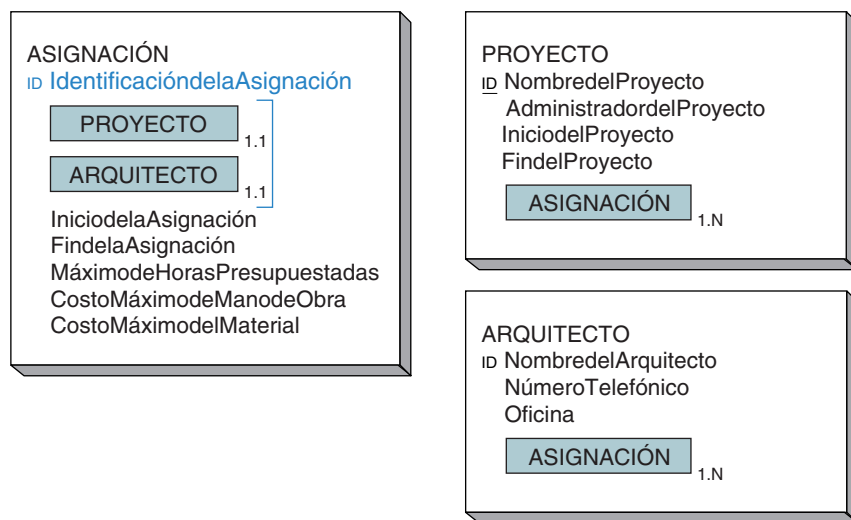
En la figura 4-28(b) se muestra un modelo mejor, en el cual el objeto EMPLEADO contiene un objeto subtipo, ADMINISTRADOR. Todos los atributos orientados al administrador se han trasladado al objeto ADMINISTRADOR. Los empleados que no son administradores tienen una instancia del objeto EMPLEADO y ninguna del objeto ADMINISTRADOR. Los empleados que son administradores tienen una instancia de EMPLEADO y una instancia de ADMINISTRADOR. En este ejemplo, al objeto EMPLEADO se le denomina **objeto padre** u **objeto supertipo**; y al objeto ADMINISTRADOR se le denomina **objeto subtipo**.

► FIGURA 4-27

Objeto de asociación
ASIGNACIÓN:
(a) ejemplo del
reporte de asignación,
y (b) objeto
ASIGNACIÓN con
objeto de ID
semántico

Reporte de asignación del proyecto			
Nombre del proyecto	Casa Abernathy	Nombre del arquitecto	Jackson, B.
Administrador del proyecto	Smith, J.	Teléfono	232-8878
Inicio del proyecto	11/11/1999	Número de Oficina	J-1133
Terminación del proyecto			
Inicio de la asignación		15/12/2001	
Fin de la asignación		15/3/2001	
Máximo de horas presupuestadas		345	
Costo máximo de la mano de obra		\$27,500	
Costo máximo del material		\$17,500	

(a)



(b)

El primer atributo de un subtipo es el atributo padre y se denota mediante el subíndice P. Se requieren siempre los atributos padre. Los identificadores del subtipo son los mismos identificadores que los del padre. En la figura 4-28, *Número de Empleado* y *Nombre de Empleado* son identificadores de EMPLEADO y ADMINISTRADOR.

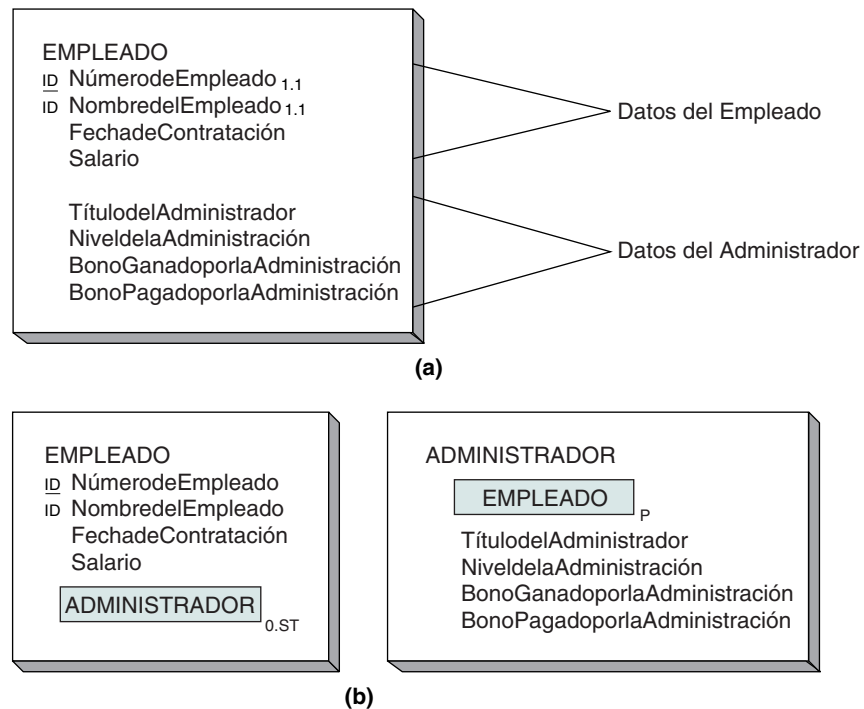
Los atributos de subtipo se muestran con el subíndice 0.ST o 1.ST. El primer dígito (0 o 1) es la cardinalidad máxima del subtipo. Si es 0, el subtipo es opcional, y si es 1 se requiere el subtipo. (Un subtipo requerido no tiene sentido en este ejemplo, pero sí lo tendrá para otros ejemplos de mayor complejidad.) El subíndice *ST* indica que el atributo es un subtipo, o un atributo ES-UN.

Los objetos padre y subtipo tienen una característica importante denominada herencia. Un subtipo adquiere, o *hereda*, todos los atributos de su padre, y por lo tanto un ADMINISTRADOR hereda todos los atributos de un EMPLEADO. Además, el padre adquiere todos los atributos de sus subtipos, y un EMPLEADO que es ADMINISTRADOR adquiere todos los atributos de un ADMINISTRADOR.

Un objeto semántico puede contener más de un atributo de subtipo. La figura 4-29 muestra un segundo objeto de EMPLEADO el cual tiene dos atributos de subtipo, ADMINISTRADOR y PROGRAMADOR. Puesto que todos estos atributos son opcionales, un EMPLEADO puede no tener ninguno, o tener uno o ambos de estos subtipos. Esto significa que algunos empleados no son ni administradores ni programadores; algunos son administradores pero no programadores; otros son programadores pero no administradores, y algunos más son programadores y administradores.

► FIGURA 4-28

Necesidad del subtipo ADMINISTRADOR:
 (a) EMPLEADO sin el subtipo, y
 (b) EMPLEADO con el subtipo ADMINISTRADOR

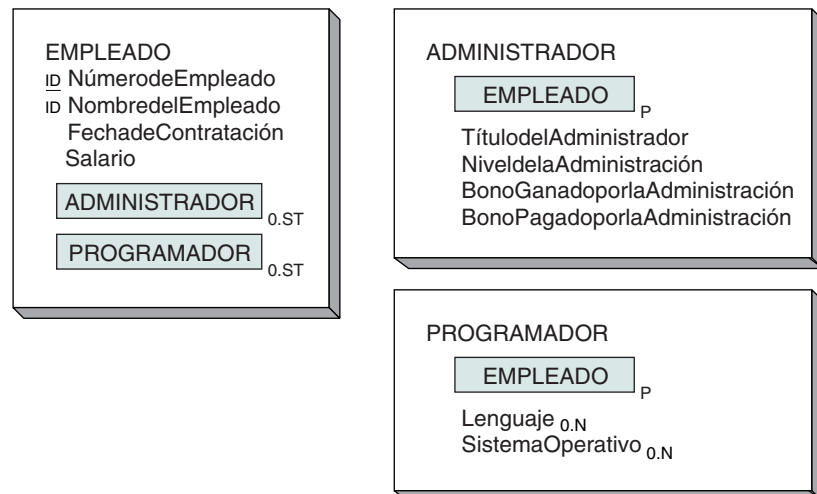


Algunas veces los subtipos se excluyen unos a otros. Esto es, un VEHÍCULO puede ser un AUTO o un AUTOBÚS, pero no ambos. Un CLIENTE puede ser una PERSONA, una SOCIEDAD, o una EMPRESA, pero sólo uno de esos tres tipos. Cuando los subtipos se excluyen unos a otros se colocan en un grupo de subtipo, y al grupo se le asigna un subíndice del formato X.Y.Z. X es la cardinalidad mínima, 0 o 1, dependiendo de si se requiere o no el grupo subtipo. Y y Z son las sumas del número de atributos en el grupo a los que se les permite tener un valor. Y es el número mínimo requerido y Z es el número máximo permitido.

La figura 4-30(a) muestra los tres tipos de CLIENTE como un grupo de subtipos. El subíndice del grupo, 0.1.1, significa que no se requiere del subtipo, pero si éste existe debe haber un mínimo de uno y un máximo de uno (o exactamente uno) de los subtipos en el grupo. Observe que cada uno de los subtipos tiene el subíndice 0.ST, lo cual significa que todos son opcionales, tal y como debe ser. Si se requirieran todos, la suma máxima tendría que ser tres y no uno. Esta notación es lo suficientemente poderosa

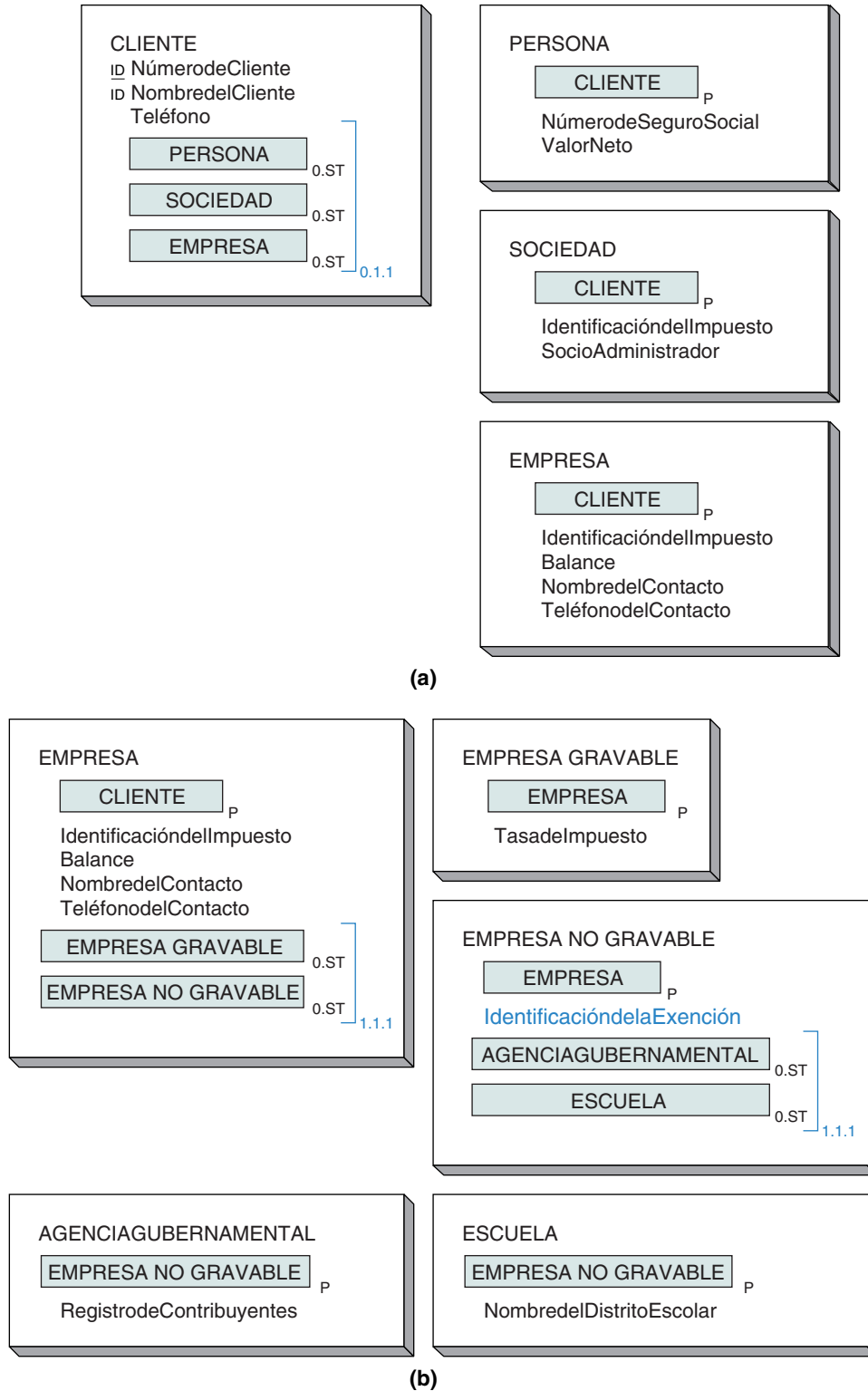
► FIGURA 4-29

EMPLEADO con dos propiedades subtipo



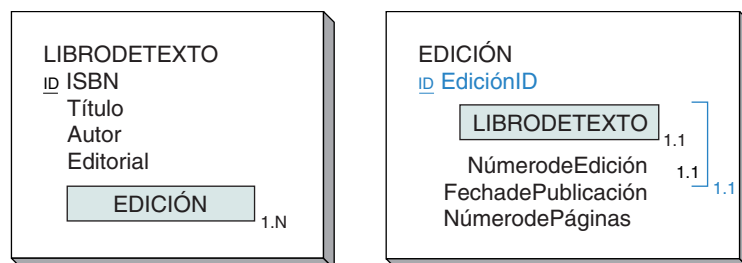
► FIGURA 4-30

Subtipos: (a) exclusivo y (b) anidado



► FIGURA 4-31

*Ejemplo de un objeto
arquetipo-versión*



como para emplearla en situaciones en las que sea necesaria una lista de tres, cinco o siete de 10 subtipos.

Incluso se pueden modelar restricciones mucho más complejas cuando los subtipos se anidan. El grupo de subtipos de la figura 4-30(b) modela una situación en la que el subtipo EMPRESA debe ser una EMPRESA SUJETA A IMPUESTOS o una EMPRESA NO SUJETA A IMPUESTOS. Si es una EMPRESA NO SUJETA A IMPUESTOS, debe ser una AGENCIA GUBERNAMENTAL o una ESCUELA. En este ejemplo sólo se muestran pocos atributos de no objetos. En realidad, si se requiriera de dicha estructura compleja probablemente existirían más atributos.

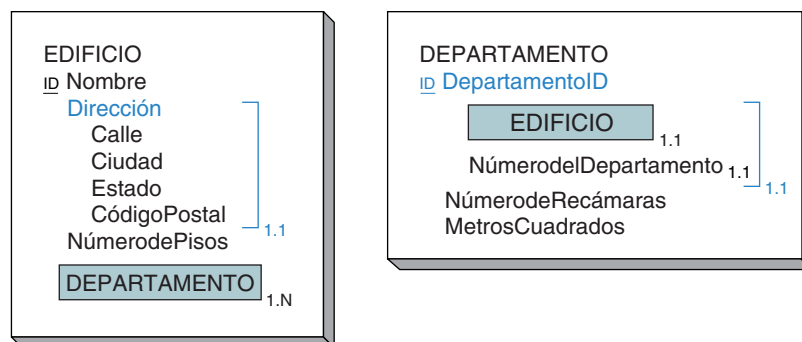
OBJETOS ARQUETIPO-VERSIÓN

El tipo final de un objeto es el **objeto arquetipo-versión**. Un objeto arquetipo es un objeto semántico que produce otros objetos semánticos, los cuales representan versiones, publicaciones o ediciones del arquetipo. Por ejemplo, en la figura 4-31 el objeto arquetipo LIBRODETEXTO produce los objetos de versiones de EDICIONES. De acuerdo con este modelo, los atributos Título, Autor y Editorial pertenecen al objeto LIBRODETEXTO, y los atributos Número de Edición, Fecha de Publicación y Número de Páginas pertenecen a la EDICIÓN del LIBRODETEXTO.

El grupo ID en EDICIÓN consta de dos partes: LIBRODETEXTO y Número de edición; éste es el patrón general para una ID de un objeto de versión. La primera parte de la ID contiene al objeto arquetipo y la segunda parte es un atributo simple que identifica la versión dentro del arquetipo. La figura 4-32 muestra otro ejemplo de los objetos arquetipo-versión.

► FIGURA 4-32

*Otro ejemplo de un
objeto arquetipo-
versión*



► COMPARACIÓN DEL OBJETO SEMÁNTICO CON EL MODELO E-R

El modelo E-R y el de objeto semántico tienen similitudes y diferencias. Son similares en tanto que son herramientas para entender y documentar la estructura de datos de los usuarios. Ambos procuran modelar la estructura de las cosas en el mundo de los usuarios, así como las relaciones entre ellos.

La diferencia principal entre los dos modelos es la orientación. El modelo E-R considera básico el concepto de *entidad*. Las entidades y sus relaciones son consideradas los átomos, por así decirlo, de un modelo de datos. Estos átomos pueden combinarse para formar lo que el modelo E-R denomina *puntos de vista del usuario*, los cuales son combinaciones de entidades cuyas estructuras son similares a la del objeto semántico.

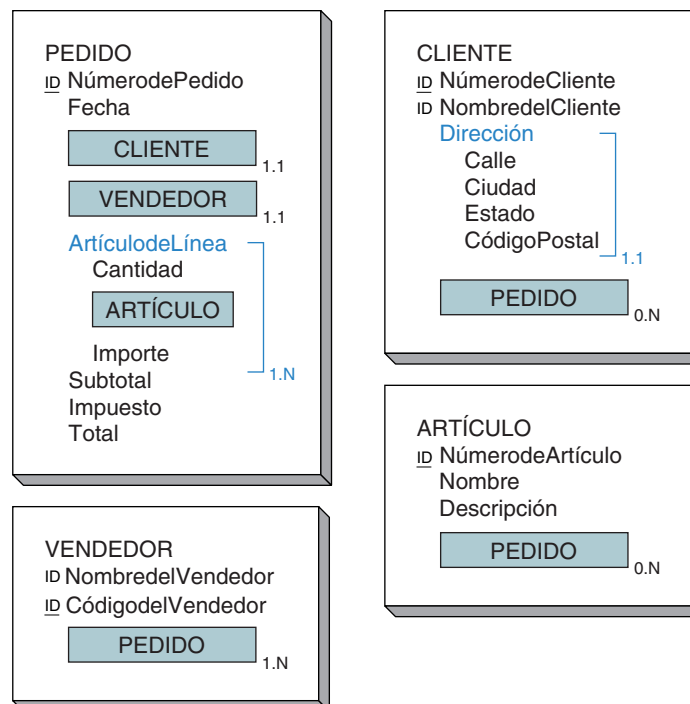
El modelo de objeto semántico considera como básico al concepto del *objeto semántico*. El conjunto de objetos semánticos en un modelo de datos es un mapa de la estructura esencial de las cosas que el usuario considera importantes. Estos objetos son los átomos del mundo de los usuarios y, por lo tanto, constituyen las unidades más pequeñas distinguibles que los usuarios desean procesar. Se pueden desglosar en partes aún más pequeñas dentro del DBMS (o la aplicación), pero esas partes pequeñas no son de interés o utilidad para los usuarios.

De acuerdo con la perspectiva del objeto semántico, las entidades, como se definen en el modelo E-R, no existen, sólo son piezas o trozos de las entidades reales. Las únicas entidades que tienen significado para los usuarios son, de hecho, los objetos semánticos. Otra manera de establecer esto es afirmar que los objetos semánticos son *semánticamente autocontenidos*, o que son *semánticamente completos*. Considere un ejemplo. La figura 4-33 muestra cuatro objetos semánticos: PEDIDO, CLIENTE, VENDEDOR y ARTÍCULO. Cuando un usuario dice: “Muéstrame el pedido número 2000”, quiere decir que se le muestre el PEDIDO como está modelado en la figura 4-33. Esto incluye, entre otros atributos, los datos de CLIENTE. Debido a que CLIENTE es parte de PEDIDO, el objeto PEDIDO incluye a CLIENTE.

La figura 4-34 es un modelo E-R de los mismos datos y contiene las entidades de PEDIDO, CLIENTE, VENDEDOR, ARTÍCULO-DE-LÍNEA e INVENTARIO. La entidad PE-

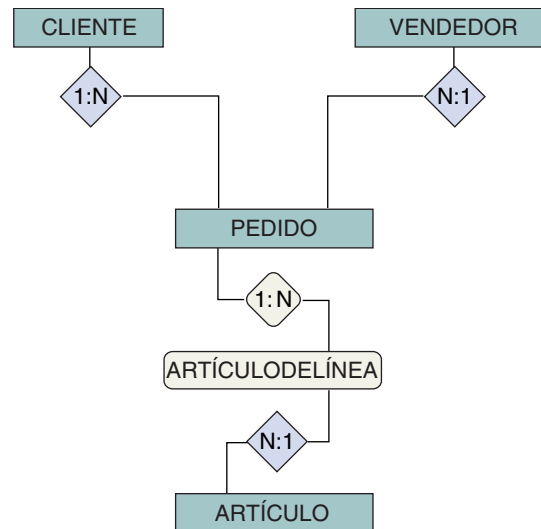
► FIGURA 4-33

PEDIDO y objetos semánticos relacionados



► FIGURA 4-34

Modelo entidad-relación de PEDIDO y CLIENTE



DIDO incluye los atributos Número de Pedido, Fecha, Subtotal, Impuestos y Total. Si un usuario dijera: “Enséñame el pedido número 2000” y sólo obtuviera los atributos Fecha, Subtotal, Impuesto y Total, se sentiría desilusionado. Quizá la respuesta de la mayoría de los usuarios sería: “¿Dónde están los demás datos?” Esto es, la entidad PEDIDO no representa el significado que los usuarios tienen de la identidad distinta PEDIDO. La entidad sólo es una parte de PEDIDO.

Al mismo tiempo, cuando un usuario (quizás el mismo) dice: “Muéstrame al cliente 12345”, seguramente desea que se le muestren todos los datos modelados para CLIENTE en la figura 4-33, incluyendo Nombre del Cliente, todos los atributos del grupo Dirección y todos los PEDIDOS relacionados con dicho CLIENTE. En la figura 4-34 la entidad CLIENTE sólo tiene los atributos Nombre del Cliente, Calle, Ciudad, Estado, Código Postal. Si el usuario dijera: “Muéstrame al cliente ABC” y se le dieran sólo estos datos, de nuevo reclamaría: “No, esto sólo es parte de lo que quiero.”

De acuerdo con la visión del objeto semántico, las entidades E-R no son necesarias. Los objetos semánticos pueden transformarse fácilmente en diseños de bases de datos sin haber considerado antes las entidades del modelo E-R. Son, por así decirlo, casas incompletas construidas en el proceso de cambio del paradigma de las estructuras de datos computacionales al paradigma del usuario.

Otra diferencia es que los objetos semánticos contienen más metadatos que las entidades. En la figura 4-33 el modelo de objeto semántico registra el hecho de que Número de Cliente es un identificador único en la mente de los usuarios. Puede o no utilizarse como un identificador para la tabla principal, pero esto no es importante para el modelo de datos. Además, el Nombre del cliente no es el único identificador para los usuarios. Por otra parte, los objetos semánticos representan el hecho de que existe un grupo semántico de atributos denominado *Dirección*. Este grupo contiene otros atributos que forman la dirección. El hecho de que este grupo exista será importante cuando se diseñen las formas y los reportes. Finalmente, los objetos semánticos indican que un ARTÍCULO puede estar relacionado con más de un PEDIDO, pero sólo puede estar relacionado con un Artículo de Línea dentro de dicho PEDIDO. Este hecho no puede mostrarse en el diagrama de entidad-relación.

En el análisis final, decida cuál de las figuras 4-33 y 4-34 le dan una mejor vista de lo que debe contener una base de datos. Mucha gente descubre que los límites dibujados alrededor de los objetos semánticos y los paréntesis en torno a los atributos grupales les ayudan a tener una mejor vista de la imagen global del modelo de datos.

► RESUMEN

Los modelos E-R y de objeto semántico se utilizan para interpretar los requerimientos y construir los modelos de datos de los usuarios. Estos modelos son como lentes a través de los cuales los analistas observan cuando estudian y documentan los datos de los usuarios. Finalmente, ambos llevan al diseño de una base de datos.

Un objeto semántico es un conjunto designado de atributos que describen suficientemente a una entidad distinta. Los objetos semánticos se agrupan en clases y las clases y las instancias de los objetos semánticos tienen nombres. Por ejemplo, el nombre de una clase es EMPLEADO y el nombre de una instancia es Empleado 2000.

Los atributos representan características de las identidades que están siendo representadas. El conjunto de atributos es suficiente en tanto que represente todas las características que el usuario necesita para realizar su trabajo. Los objetos representan identidades distintas, instancias que los usuarios consideran como independientes y por separado. Las identidades distintas representadas pueden o no ser físicas; pero sí son representativas, como por ejemplo un contrato.

Los atributos de objetos pueden ser datos simples, grupos u otros objetos semánticos. Los diagramas de objetos sintetizan la estructura de los objetos. Los nombres de los objetos se escriben con letras mayúsculas en la parte superior del diagrama. Los atributos de los no objetos se indican con mayúsculas iniciales, y los atributos grupales se indican con paréntesis.

Todos los atributos tienen una cardinalidad mínima que indica cuántas instancias del atributo deben existir para que el objeto sea válido. Asimismo, tienen una cardinalidad máxima que indica el número máximo de instancias del atributo permitidas. La cardinalidad se escribe en el formato N.M, donde N es la cardinalidad mínima y M es la máxima. Para reducir el desorden en un diagrama de objetos, si la cardinalidad del atributo de un valor simple es 0.1, no se muestra. Pero la cardinalidad siempre se muestra para los atributos de objeto y grupales.

Los atributos de objeto siempre seorean. Si un primer objeto tiene una relación con un segundo objeto, este último debe tener una relación con el primero. Los identificadores de objetos son atributos que sirven, en la mente de los usuarios, para identificar objetos. Los identificadores pueden ser únicos o no únicos. Cualquier tipo de atributo puede ser un identificador. Los identificadores se muestran con las letras ID junto al atributo. Si éste es único, entonces se subrayan las letras ID. Por lo general las cardinalidades de un identificador son 1.1.

El dominio de un atributo es el conjunto de todos los valores posibles que puede tener. Los dominios tienen una definición física y una semántica. Existen tres tipos de dominios: simple, grupal y de objeto semántico.

Las aplicaciones procesan los objetos a través de las vistas de usuarios. Una vista de un objeto consta del nombre de éste y de todos los atributos visibles desde dicha vista. Con frecuencia la definición de una vista y un objeto es un proceso iterativo.

El proceso de desarrollar un conjunto de diagramas de objetos es iterativo. Se examinan las formas y los reportes; se documenta un conjunto inicial de objetos, y se revisan los reportes y las formas nuevas para revelar los objetos y cambios en los objetos existentes. Este proceso continúa hasta que se hayan examinado todas las formas y reportes.

Existen siete tipos de objetos. Los objetos simples no tienen atributos de valores múltiples ni atributos de objetos. Los objetos compuestos tienen atributos de valores múltiples, pero no tienen atributos de objetos. Los objetos compuestos tienen atributos de objetos, y los objetos híbridos combinan a los objetos combinados y compuestos. Los objetos de asociación relacionan a dos o más objetos distintos. Los objetos de subtipo se utilizan para representar las especializaciones de los objetos. Finalmente, los objetos versión-arquetipo se emplean para modelar objetos que contienen datos de la base junto con versiones o variaciones múltiples.

El modelo E-R considera básicas a las entidades, mientras que el modelo de objeto semántico asume como básicos a los objetos semánticos. El modelo de objeto semántico también contiene más información referente a la interpretación de los datos que el modelo E-R.

► PREGUNTAS DEL GRUPO I

- 4.1 Explique por qué el modelo E-R y el modelo de objeto semántico son como lentes.
- 4.2 Defina al *objeto semántico*.
- 4.3 Explique la diferencia entre el nombre de la clase de un objeto y el nombre de una instancia del objeto. Proporcione un ejemplo sobre cada uno.
- 4.4 ¿Qué se requiere para que un conjunto de atributos sea una descripción suficiente?
- 4.5 Explique las palabras *identidad bien definida* cuando corresponden a la definición de un objeto semántico.
- 4.6 Explique por qué una línea de artículo de un pedido no es un objeto semántico.
- 4.7 Enumere los tres tipos de atributos.
- 4.8 Dé un ejemplo sobre cada uno de los siguientes atributos:
 - a. un atributo simple de un valor único
 - b. un atributo grupal de un valor único
 - c. un atributo simple de valores múltiples
 - d. un atributo grupal de valores múltiples
 - e. un atributo de objeto simple
 - f. un atributo de objeto de valores múltiples
- 4.9 ¿Qué es la cardinalidad mínima? ¿Cómo se utiliza? ¿Qué tipo de atributos tiene?
- 4.10 ¿Qué es la cardinalidad máxima? ¿Cómo se utiliza? ¿Qué tipo de atributos tiene?
- 4.11 ¿Qué son los atributos pareados? ¿Por qué son necesarios?
- 4.12 ¿Qué es un identificador de objeto? Mencione un ejemplo referente a un identificador de objeto de atributo simple, y un ejemplo de un identificador de objeto de un atributo grupal.
- 4.13 Defina el *dominio de un atributo*. ¿Cuáles son los tipos de dominio de un atributo? ¿Por qué es necesaria una descripción semántica?
- 4.14 ¿Qué es una vista del objeto semántico? Dé un ejemplo sobre un objeto y dos vistas diferentes a los ejemplos que se mencionan en este texto.
- 4.15 Mencione un ejemplo de un objeto simple, diferente al que analizamos en este capítulo.
- 4.16 Proporcione tres ejemplos de objetos compuestos diferentes a los de este capítulo. Uno de los ejemplos deberá tener sólo un atributo simple de valores múltiples; el otro, dos grupos de valores múltiples independientes; y el tercero, grupos de valores múltiples anidados.
- 4.17 Mencione un ejemplo de cuatro conjuntos de objetos combinados diferentes a los que analizamos en este capítulo. Un conjunto deberá tener una relación 1:1; el otro, una relación 1:N; otro más, una relación M:1, y el conjunto restante una relación M:N.
- 4.18 Mencione un ejemplo de un objeto híbrido diferente al de este capítulo.
- 4.19 Proporcione un ejemplo de un objeto de asociación, y dos objetos compuestos diferentes a los analizados en este capítulo.
- 4.20 Proporcione un ejemplo de un objeto de supertipo con tres objetos de subtipo diferentes a los del presente capítulo.
- 4.21 Dé un ejemplo referente a los objetos arquetipo-versión diferente al que se proporcionó en este capítulo.

- 4.22 Explique las similitudes entre el modelo E-R y el modelo de objeto semántico.
- 4.23 Explique las diferencias principales entre el modelo E-R y el modelo de objeto semántico.
- 4.24 Explique la razón por la cual las entidades, según lo define el modelo E-R, no existen realmente.
- 4.25 Muestre cómo el modelo E-R y el modelo de objeto semántico representarían los datos que fundamentan la forma de PEDIDO en la figura 4-24(a), y explique las diferencias principales.

► PREGUNTAS DEL GRUPO II

- 4.26 Modifique el diagrama del objeto semántico en la figura 4-13 para incluir a los objetos CLASE, PRESENTACIÓN DE LA CLASE, e INSCRIPCIÓN. Suponga que INSCRIPCIÓN es un objeto de asociación que relaciona a un ESTUDIANTE con una CLASE.
- 4.27 Modifique el diagrama del objeto semántico en la figura 4-13 para incluir al objeto COMITÉ. Suponga que varios PROFESORES son asignados a un comité y que un COMITÉ incluye a varios PROFESORES. Cree el objeto JUNTA como un objeto arquetipo-versión, el cual represente las juntas de un COMITÉ.
- 4.28 Modifique su respuesta a la pregunta 4.27, de tal manera que JUNTA sea un grupo de valores múltiples dentro de COMITÉ. ¿Este modelo es mejor que el de la pregunta 4.27? Justifique su respuesta.

► PROYECTOS

A. Desarrolle un modelo de objeto semántico para el caso AMV en el proyecto “A” al final del capítulo 3.



B. Ingrese a la página Web de un fabricante de computadoras, por ejemplo Dell (www.dell.com). Utilice el sitio de Internet para determinar qué computadora portátil compraría para un usuario potencial que tiene un presupuesto de \$10,000. Mientras usa el sitio de Internet, piense en la estructura de una posible base de datos de sistemas y subsistemas computacionales para apoyar este sitio.

Desarrolle un modelo de objeto semántico de la base de datos de sistemas y subsistemas computacionales para este sitio de Internet. Los objetos posibles son SISTEMA DE BASE, OPCIÓN DE MEMORIA, TARJETA DE VÍDEO e IMPRESORA. Muestre las relaciones de los objetos y cuando menos dos o tres atributos por objeto. Indique el tipo de cada objeto semántico. Para evitar que este proyecto se dispare en tamaño, restrinja su diseño a las necesidades de alguien que realiza una decisión de compra.



C. Ingrese al sitio de una librería como Amazon (www.amazon.com) y seleccione tres de los mejores libros sobre XML (Lenguaje extendido de marcaje) para alguien que apenas está comenzando a estudiar ese tema. Conforme use el sitio de Internet, piense en la estructura de una posible base de datos de libros, autores, títulos y temas relacionados.

Desarrolle un modelo de objeto semántico de una base de datos de libros para ese sitio de Internet. Los objetos posibles son TÍTULO, AUTOR, EDITORIAL y TEMA. Muestre las relaciones de los objetos y cuando menos dos o tres atributos por objeto. Indique el tipo de cada objeto semántico. Para evitar que este proyecto se dispare en tamaño, suponga que sólo dará seguimiento a libros; además, restrinja su diseño a las

necesidades de alguien que busca comprar libros. No considere el pedido del cliente, surtido de la mercancía, ni otros procesos de negocio.

► PREGUNTAS DEL PROYECTO FIREDUP

Considere la situación de la compañía FiredUp que analizamos al final de los capítulos 1 y 2. Suponga que esa compañía ha desarrollado una línea de tres estufas nuevas: FiredNow, FiredAlways y FiredAtCamp. Además suponga que los dueños están vendiendo refacciones para cada línea y también hacen reparaciones. Algunas reparaciones se llevan a cabo sin ningún cargo debido a que todavía tienen garantía; en otras, sólo se cobran las refacciones; y en algunas más se cobran la mano de obra y las refacciones. La compañía FiredUp desea realizar un seguimiento de todos estos datos. Cuando se les pidieron detalles adicionales los dueños hicieron la siguiente lista:

CLIENTE: Nombre, Dirección, NúmeroDepartamento, Ciudad, Estado/Provincia, CódigoPostal, País, CorreoElectrónico, NúmeroTelefónico

ESTUFA: NúmerodeSerie, Tipo, FechadeFabricación, InicialesdelInspector

FACTURA: NúmerodeFactura, Fecha, Cliente, con una lista de artículos vendidos y sus precios, PrecioTotal

REPARACIÓN: NúmerodeReparación, Cliente, Estufa, Descripción, con una lista de las refacciones que se utilizaron y su costo, si es el caso, y CantidadTotal de la reparación

PARTE: Número, Descripción, Costo, PreciodeVenta

A. Cree un conjunto de objetos semánticos para una base de datos de la compañía FiredUp. Establezca las cardinalidades mínima y máxima de todos los atributos como lo considere apropiado. Explique las razones en las que se basó para establecer cada valor de cardinalidad. Utilice tantos tipos de objetos semánticos como considere necesario, pero no utilice los subtipos. Esta tarea es más fácil si descarga el *Tabledesigner*, que es la herramienta que se describe en el apéndice B para hacer el diagrama de un objeto semántico. Utilícela para definir sus objetos semánticos e imprimir los reportes de los objetos semánticos que le propondrá a su instructor. Vea el apéndice B y pregunte a su instructor antes de llevar a cabo esto.

B. Modifique sus diagramas de objetos en su respuesta a la pregunta A mediante la representación de FACTURA y REPARACIÓN como subtipos. ¿Bajo qué circunstancias es mejor este diseño que el de su respuesta a la pregunta A?

C. Suponga que la compañía FiredUp desea hacer un seguimiento de números telefónicos particulares, números de fax y teléfonos celulares, así como de las múltiples direcciones de correo electrónico de sus clientes. Modifique sus objetos para tomar en cuenta los valores múltiples de NúmeroTelefónico y CorreoElectrónico.

D. Suponga que la compañía FiredUp desarrolla diferentes versiones del mismo producto de estufa. Por lo tanto, desarrollaron “FiredNow”, versión 1, y “FiredNow”, versión 2, etc. Modifique sus objetos en la anterior pregunta A como sea necesario para responder a esta situación.

E. Cuando se les pregunta a los usuarios sobre los datos que desean registrar, no siempre recuerdan todo lo que necesitan. Empleando su conocimiento sobre las operaciones de un negocio pequeño, haga una lista de los objetos semánticos que los usuarios podrían haber olvidado. Asegúrese de mostrar las relaciones entre los objetos. ¿Cómo determinará si se necesitan cualquiera de estos datos adicionales en la compañía FiredUp?

F. Si usted contestó las preguntas del proyecto de la compañía FiredUp al final del capítulo 3, compare el modelo entidad-relación desarrollado para dicho capítulo con el modelo de objeto semántico que estudiamos aquí. ¿Cuál de los dos prefiere? ¿Cuál cree que entenderían mejor los dueños de la compañía FiredUp?





DISEÑO DE BASES DE DATOS

Los capítulos correspondientes a la parte III analizan el diseño de bases de datos. El capítulo 5 presenta el modelo relacional y la normalización. El modelo relacional es importante porque con él se expresan la mayoría de los diseños de las bases de datos, y porque es el fundamento de la mayoría de los productos DBMS actuales.

La normalización es importante porque constituye una técnica para comprobar la calidad del diseño relacional. Con base en los conceptos del capítulo 5, en el capítulo 6 abordamos el proceso de transformación de modelos de datos de entidad-relación en diseños relacionales independientes de un DBMS. A continuación, en el capítulo 7 se describe el proceso para transformar modelos de datos de objeto semántico en dichos diseños.

El modelo relacional y la normalización

El modelo relacional es importante por dos razones. Primera, debido a que los elementos de construcción del modelo relacional son extensos y generales, y se pueden usar para expresar diseños de bases de datos independientes de un DBMS. Segunda, el modelo relacional es la base de casi todos los productos DBMS. Por lo tanto, es esencial comprender los conceptos de este modelo.

El presente capítulo aborda las bases del modelo relacional y explica los conceptos fundamentales de la *normalización*. Empezaremos por señalar el hecho de que no todas las relaciones son iguales; algunas son mejores que otras. La normalización es un proceso para convertir una relación que tiene ciertos problemas, en dos o más relaciones que no los tienen. Lo que es más importante: la normalización se puede usar como un lineamiento para comprobar la pertinencia y validez de las relaciones. Se han realizado muchos trabajos teóricos en torno a la pregunta: ¿qué es una relación bien estructurada? Este trabajo se llama *normalización* porque uno de los pioneros en la tecnología de bases de datos, E. F. Codd, definió varias *formas normales* de relaciones. En el presente capítulo examinaremos la normalización, incluyendo los resultados de teoremas que son útiles e importantes para los profesionales de bases de datos. Las pruebas de estos teoremas, así como un tratamiento más formal y riguroso de estos temas, los puede encontrar en los trabajos de C. J. Date y J. D. Ullman.¹

¹ C. J. Date, *An Introduction to Database Systems*, Sexta edición (Reading, MA: Addison-Wesley, 1994); y J. D. Ullman y Jennifer Widom, *A First Course in Database Systems* (Upper Saddle River, NJ: Prentice Hall, 1997).

► EL MODELO RELACIONAL

Una **relación** es una tabla bidimensional. Cada renglón tiene datos que pertenecen a alguna cosa o a una parte de ésta. Cada columna de esta tabla contiene datos referentes a un atributo. Algunas veces los renglones se llaman **tuples** y las columnas, **atributos**.

Los términos *relación*, *tuple* y *atributo* derivan de las matemáticas relacionales, que constituyen la fuente teórica de este modelo. Los profesionales de Sistemas de información gerencial (MIS, por sus siglas en inglés) consideran más adecuada la analogía de los términos *archivo*, *registro* y *campo*, y la mayoría de los usuarios encuentran más razonables los términos *tabla*, *renglón* y *columna*. La figura 5-1 resume esta terminología.

Para que una tabla sea una relación debe cumplir ciertas restricciones.² Primero, las celdas de la tabla deben ser de un valor único; no se permite repetir grupos ni tener series en calidad de valores.³ Todas las entradas en cualquier columna deben ser del mismo tipo. Por ejemplo, si la tercera columna en el primer renglón de una tabla contiene *NúmerodeEmpleados*, entonces la tercera columna también debe contener *NúmerodeEmpleados* en todos los demás renglones de la tabla. Cada columna tiene un nombre único y el orden de las columnas en la tabla no es importante. Por último, dos renglones en la tabla no pueden ser idénticos y el orden de los renglones no tiene importancia.

La figura 5-2 es un ejemplo de tabla. Observe que hay siete renglones de cuatro columnas. Si fuésemos a reacomodar las columnas (es decir, colocando *NúmerodeEmpleado* en el extremo izquierdo) o a reordenar los renglones (quizás en orden ascendente por edad), obtendríamos una tabla equivalente.

La figura 5-2 muestra una ocurrencia o instancia de una tabla. El formato generalizado, EMPLEADO (Nombre, Edad, Sexo, *NúmerodeEmpleado*) se denomina estructura de la relación, y esto es lo que la mayoría de las personas entiende cuando usa el término *relación*.

Para comprender la normalización necesitamos definir dos términos importantes: **dependencia funcional** y **llave**.

DEPENDENCIAS FUNCIONALES

Una *dependencia funcional* es una relación entre uno o más atributos. Suponga que si asignamos el valor de un atributo podemos obtener (o buscar) el valor de otro. Por ejemplo, si conocemos el valor de *NúmerodeCuentadelCliente*, podemos encontrar el valor de *BalancedelCliente*. Si esto es verdad, podemos decir que *BalancedelCliente* es *funcionalmente dependiente* de *NúmerodeCuentadelCliente*.

En términos generales, el atributo Y es funcionalmente dependiente del atributo X si el valor de X determina al valor de Y. Dicho de otra manera, si conocemos el valor de X podemos obtener el valor de Y.

Las ecuaciones representan dependencias funcionales. Por ejemplo, si conocemos el precio de un artículo y la cantidad de artículos que han sido vendidos, podemos calcular el precio total de esos artículos:

$$\text{PrecioTotal} = \text{PreciodelArtículo} \times \text{Cantidad}$$

► FIGURA 5-1

Terminología
relacional
equivalente

Modelo relacional	Programador	Usuario
Relación	Archivo	Tabla
Tuple (renglón)	Registro	Renglón
Atributo	Campo	Columna

²E. F. Codd, "A Relational Model of Data for Large Shared Databanks", *Communications of the ACM*, junio de 1970, pp. 377-387.

³Esto no significa que los valores deban tener una longitud fija. Por ejemplo, un campo de longitud variable es un valor perfectamente legítimo. Sin embargo, se permite *un* solo valor.

► FIGURA 5-2

Relación
EMPLEADO

	Atributo 1 Nombre	Atributo 2 Edad	Atributo 3 Género	Atributo 4 Número de Empleado
Tuple 1	Anderson	21	F	010110
Tuple 2	Decker	22	M	010100
.	Glover	22	M	101000
.	Jackson	21	F	201100
.	Moore	19	M	111100
.	Nakata	20	F	111101
Tuple 7	Smith	19	M	111111

En este caso, podríamos decir que PrecioTotal es funcionalmente dependiente de Precio del Artículo y de Cantidad.

Las dependencias funcionales entre atributos en una relación por lo general no implican ecuaciones. Por ejemplo, suponga que los estudiantes tienen un número único de identificación (ID del estudiante, o EID) y que cada uno tiene una y sólo una especialidad. Considerando el valor de un EID, podemos encontrar la especialidad del estudiante, por lo que la especialidad es funcionalmente dependiente de EID. También considere las microcomputadoras en un laboratorio de cómputo. Cada una tiene únicamente un tamaño de memoria principal, por lo que Tamaño de Memoria es funcionalmente dependiente de Número de Serie de la Computadora.

A diferencia de una ecuación, estas dependencias funcionales no se pueden resolver usando aritmética; en vez de eso, se listan en una base de datos. De hecho, uno puede argumentar que el almacenamiento y la recuperación de las dependencias funcionales es la única razón para tener una base de datos.

Las dependencias funcionales se escriben usando la siguiente notación:

EID : Especialidad

Número de Serie de la Computadora : Tamaño de Memoria

La primera expresión se lee como “EID determina funcionalmente a Especialidad”, o “Especialidad es dependiente de EID”. Los atributos en el lado izquierdo de la flecha se llaman **determinantes**.

Como se ha establecido, si EID determina a Especialidad, un valor particular de EID será pareado con *un* solo valor de Especialidad. A la inversa, un valor de Especialidad podrá ser pareado con *uno o más* valores diferentes de EID. Suponga que un estudiante tiene un valor EID de 123 y se especializa en contabilidad. Cuando EID y Especialidad se encuentren juntas en una relación, el valor EID de 123 será siempre pareado con el valor de Especialidad en Contabilidad. Sin embargo, lo opuesto no es cierto, ya que Especialidad en Contabilidad se puede parear con varios valores de EID (varios estudiantes pueden tener la especialidad en contabilidad). Consecuentemente, podemos afirmar que la relación de EID con Especialidad es de muchos a uno (N:1). En general, se puede decir que A determina a B, la relación de valores de A y B es N:1.

Las dependencias funcionales pueden involucrar grupos de atributos. Considere la relación CALIFICACIONES (EID, Nombre de Clase, Calificación). La combinación de un EID y un Nombre de Clase determina una calificación, una dependencia funcional que se escribe

(EID, Nombre de Clase) → Calificación

Observe que tanto EID como Nombre de Clase son necesarios para determinar Calificación. No podemos subdividir la dependencia funcional porque ni EID ni Nombre de Clase determinan la calificación por sí mismos.

Note la diferencia en los siguientes patrones: Si $X \rightarrow (Y, Z)$, entonces $X \rightarrow Y$ y $X \rightarrow Z$. Por ejemplo, si $EID \rightarrow (\text{Nombre del Estudiante}, \text{Especialidad})$, entonces $EID \rightarrow \text{Nombre del Estudiante}$ y $EID \rightarrow \text{Especialidad}$. Pero si $(X, Y) \rightarrow Z$, entonces en general no es

► FIGURA 5-3

Relación
ACTIVIDAD

ACTIVIDAD (EID, Actividad, Cuota)

Llave: EID

Datos de muestra

EID	Actividad	Cuota
100	Esquí	200
150	Natación	50
175	Squash	50
200	Natación	50

verdad que $X \rightarrow Y$ o $Y \rightarrow Z$. Por lo tanto, si $(EID, NombredeClase) \rightarrow Calificación$, entonces ni EID ni NombredeClase determinan por sí mismos a Calificación.

LLAVES

Una *llave* es un grupo de uno o más atributos que identifica únicamente a un renglón. Considere la relación ACTIVIDAD en la figura 5-3, cuyos atributos son EID, Actividad y Cuota. El significado de un renglón es que un estudiante se inscribe en una actividad con la cuota especificada. Se supone que al estudiante sólo se le permite participar en una sola actividad a la vez. En este caso, un valor de EID determina un único renglón y esto es una llave.

Las llaves también pueden estar compuestas por un grupo de atributos tomados en conjunto. Por ejemplo, si a los estudiantes se les permitiera inscribirse en distintas actividades al mismo tiempo, sería posible que un valor de EID apareciera en dos o más renglones de la tabla, y de esta forma EID podría no identificar únicamente a un renglón. Quizá se requerirían algunas combinaciones de atributos (EID, Actividad).

Por otra parte, existe un sutil pero importante aspecto en el párrafo anterior. Si los atributos son llaves o no, y si son dependencias funcionales o no, esto no se determina a través de un conjunto de reglas abstractas, sino mediante los modelos mentales de los usuarios, y con base en las reglas del negocio de la empresa u organización que utiliza la base de datos. En este ejemplo, ya sea EID la llave (EID, Actividad), o alguna otra combinación, la llave se determina por completo mediante la semántica fundamental de las personas de la empresa que usan la base de datos. Debemos preguntar a los usuarios para resolver estas interrogantes. Recuerde que todas las suposiciones que hagamos acerca de dependencias funcionales, llaves, restricciones y similares están determinadas por los modelos mentales de los usuarios.

Después de entrevistar a los usuarios, suponga que descubrimos que los estudiantes tienen, en efecto, permiso para participar en varias actividades al mismo tiempo. Esta situación se representa mediante la relación ACTIVIDADES, que se muestra en la figura 5-4. Como establecimos, EID *no* es una llave de esta relación. Por ejemplo, el estudiante 100 se ha inscrito en esquí y en golf, y el valor EID de 100 aparece en dos ren-

► FIGURA 5-4

Relación
ACTIVIDADES

EID	Actividad	Cuota
100	Esquí	200
100	Golf	65
150	Natación	50
175	Squash	50
175	Natación	50
200	Natación	50
200	Golf	65

giones diferentes. De hecho, para esta relación, ningún atributo solo es una llave, por lo que ésta debe ser una combinación de dos o más atributos.

Considere la combinación de dos atributos de esta tabla. Existen tres posibilidades: (EID, Actividad), (EID, Cuota) y (Actividad, Cuota). ¿Alguna de estas combinaciones es una llave? Si es así, únicamente debe identificar un renglón. Nuevamente, para decidir preguntas como ésta debemos consultar con los usuarios. No podemos depender de datos muestra como los de la figura 5-4, o confiar en nuestras propias suposiciones para tomar una decisión.

Después de hablar con los usuarios, suponga que descubrimos que las distintas actividades pueden tener la misma cuota. En este caso, la combinación (EID, Cuota) no puede determinar un renglón único. Por ejemplo, el estudiante 100 puede inscribirse en dos actividades diferentes cuyo costo en conjunto es de \$200. Esto significaría que la combinación (100, \$200) ocurre dos veces en la tabla, así que esta combinación no puede ser una llave.

¿Puede ser una llave la combinación (Actividad, Cuota)? ¿La combinación (Esquí, \$200) determina un renglón único? No, debido a que muchos estudiantes pueden participar en esquí. ¿Qué pasa con (EID, Actividad)? Considerando lo que nos han dicho los usuarios, ¿una combinación de valores para EID y Actividad puede determinar un renglón único? Sí, en tanto no necesitemos conservar registros de las diferentes ocasiones en las que un estudiante se inscribe en una actividad determinada. En otras palabras, ¿esta tabla será usada para registrar sólo las actividades actuales de los estudiantes, o se supone que también se conservarán los registros de las actividades anteriores?

Nuevamente, debemos consultar con los usuarios para responder esta pregunta. Supongamos que nos dicen que sólo hay que conservar los registros de actividades actuales. Entonces la combinación (EID, Actividad) puede determinar un renglón único, y consecuentemente (EID, Actividad) es la llave para esta relación. Si los usuarios especificaron que se conservaran los registros de actividades actuales y pasadas, la relación en la figura 5-3 tendrá renglones duplicados. Debido a que esto lo prohíbe la definición de relación, necesitaremos agregar otros atributos tales como Fecha.

Esto nos genera un punto importante. Cada relación tiene cuando menos una llave. Esto puede ser verdad porque no hay relación que pueda tener renglones duplicados y, por lo tanto, en el extremo, las llaves constan de todos los atributos de la relación.

DEPENDENCIA FUNCIONAL, LLAVES Y UNICIDAD

Muchos estudiantes confunden los conceptos dependencia funcional, llaves y unicidad. Para evitar esta confusión, considere lo siguiente: un determinante de una dependencia funcional puede o no ser única en una relación. Si sabemos que A determina a B y que A y B están en la misma relación, aún no sabemos si A es única en esta relación. Sólo tenemos conocimiento de que A determina a B.

Por ejemplo, en la relación ACTIVIDADES, Actividad determina funcionalmente a Cuota, e incluso puede haber varias instancias de una Actividad particular en la relación. La dependencia funcional sólo establece que cada vez que Actividad ocurre con Cuota, ésta siempre ocurre con el mismo valor de Cuota. Esto es, esquí siempre costará \$200, sin importar cuántas veces aparezca el valor Esquí en la tabla.

A diferencia de los determinantes, las llaves son siempre únicas. Una llave determina funcionalmente a todo el renglón. Si se duplica el valor de la llave, se duplicará el tuple completo. Pero lo anterior no está permitido porque, por definición, los renglones en una relación deben ser únicos. Así, cuando decimos que un atributo (o una combinación) es una llave, sabemos que será única. Si (EID, Actividad) es una llave, por ejemplo, la combinación (100, Esquí) sólo ocurrirá una vez en una relación.

Para comprobar su comprensión sobre estos conceptos, intente explicar por qué, en la relación ACTIVIDAD de la figura 5-3, EID es tanto un determinante como una llave, pero Actividad es un determinante y no una llave. (Recuerde que la relación en la figura 5-3 refleja la política de que un estudiante puede participar, cuando mucho, en una actividad.)

► NORMALIZACIÓN

Por desgracia, no todas las relaciones son igualmente deseables. Una tabla que cumple la mínima definición de una relación puede no tener una estructura eficaz o apropiada. Para algunas relaciones el cambio de datos puede tener consecuencias indeseables, llamadas **anomalías de modificación**. Éstas se pueden eliminar redefiniendo la relación entre dos o más relaciones. En la mayoría de los casos son preferibles las relaciones redefinidas, o **normalizadas**.

ANOMALÍAS DE MODIFICACIÓN

Nuevamente considere Actividad en la figura 5-3. Si elimináramos el tuple para Estudiante 100, perderíamos no sólo el hecho de que el estudiante 100 es un esquiador, sino también que esquiar cuesta \$200. Esto se llama **anomalía de eliminación**; esto es, al suprimir los hechos acerca de una entidad (que Estudiante 100 es un esquiador), inadvertidamente eliminamos los hechos de otra entidad (que esquiar cuesta \$200). Con una supresión, o borrado, se pierden hechos acerca de dos entidades.

La misma relación se puede usar para ilustrar una **anomalía de inserción**. Suponga que deseamos almacenar el hecho de que bucear cuesta \$175, pero no podemos ingresar este dato en la relación ACTIVIDAD hasta que un estudiante bucee. Esta restricción parece ridícula. ¿Por qué tendríamos que esperar hasta que alguien realizara la actividad para registrar el precio? A esta restricción se le llama anomalía de inserción. No podemos insertar un hecho de una entidad hasta que tengamos un hecho adicional acerca de otra entidad.

La relación en la figura 5-3 se puede usar para algunas aplicaciones, pero esto obviamente representa problemas. Podemos eliminar tanto las anomalías como la inserción de anomalías dividiendo la relación ACTIVIDAD en dos, cada una relacionada con un tema diferente. Por ejemplo, podemos poner los atributos EID y Actividad en una relación (a la nueva relación la llamaremos ESTUD-ACT-para denotar actividad de estudiante) y colocar los atributos de Actividad y Cuota en otra relación llamada COSTO-ACT (para costo de actividad). La figura 5-5 contiene los mismos datos de muestra almacenados en esas dos nuevas relaciones.

Ahora, si borramos Estudiante 100 de ACT-ESTUD, no perdemos el hecho de que esquiar cuesta \$200. Además, podemos agregar buceo y sus cuotas a la relación COSTO-ACT, aun antes de cualquier inscripción. Por lo tanto, han quedado suprimidas las anomalías de eliminación e inserción.

Sin embargo, separar una relación en dos tiene una desventaja. Suponga que un estudiante trata de inscribirse en una actividad que no existe. Por ejemplo, Estudiante 250 quiere registrarse en ráquetbol. Podemos insertar este nuevo tuple en ACT-ESTUD (el renglón puede contener 250, Ráquetbol), pero ¿debemos hacerlo? ¿Un estudiante podrá registrarse en una actividad que no esté en la relación COSTO-ACT? Dicho de otra manera, ¿el sistema podrá prevenir de algún modo que se agreguen renglones de estudiantes si el valor de ACTIVIDAD no está en la tabla de COSTO-ACT? La respuesta a esta pregunta se encuentra en los requerimientos del usuario. Si la acción estuviera prohibida, esta restricción (un tipo de regla del negocio) se debería documentar como parte del esquema de diseño. Posteriormente, en la puesta en marcha, la restricción se-

► FIGURA 5-5

*División de
ACTIVIDAD en
dos relaciones*

ESTUD-ACT (EID, Actividad)
Llave: EID

EID	Actividad
100	Esquí
150	Natación
175	Squash
200	Natación

COSTO-ACT (Actividad, Cuota)
Llave: Actividad

Actividad	Cuota
Esquí	200
Natación	50
Squash	50

rá definida en el DBMS si el producto en uso proporciona la verificación de restricciones; de lo contrario, dicha restricción debe imponerse mediante programas de aplicación.

Suponga que el usuario especifica que las actividades pueden existir antes de que cualquier estudiante se inscriba en ellas, pero que no se puede inscribir en una actividad que no tenga una cuota asignada (esto es, en ninguna actividad que no se encuentre en la tabla COSTO-ACT). Podemos documentar esta restricción en alguna de las varias formas durante el diseño de la base de datos: Actividad en ESTUD-ACT es un subconjunto de Actividad en COSTO-ACT, o ESTUD-ACT [Actividad] es un subconjunto de COSTO-ACT [Actividad], o ESTUD-ACT [Actividad] $\subseteq$ COSTO-ACT [Actividad].

De acuerdo con esta notación, los corchetes [] denotan una columna de datos que se extrae de una relación. Estas expresiones simplemente significan que el valor en el atributo Actividad de ESTUD-ACT debe existir en el atributo Actividad de COSTO-ACT. También significa que antes de permitir que Actividad sea introducida en ESTUD-ACT, debemos comprobar y asegurarnos que ya esté presente en COSTO-ACT. Restricciones como esta se llaman **restricciones de integridad referencial**, o **restricciones de interrelación**.

ESENCIA DE LA NORMALIZACIÓN

Las anomalías en la relación ACTIVIDAD en la figura 5-3 se pueden establecer de la siguiente forma intuitiva: los problemas ocurren porque ACTIVIDAD contiene hechos acerca de dos temas diferentes:

- Los estudiantes que participan en cada actividad
- Cuánto cuesta cada actividad

Cuando abrimos un nuevo renglón, debemos agregar datos de dos temas al mismo tiempo, y cuando eliminamos un renglón, por fuerza tenemos que suprimir datos de dos temas al mismo tiempo.

¿Recuerda a su maestro de español de segundo de secundaria? Él o ella le decían que un párrafo podía tener un solo tema o idea. Si el párrafo tenía más de un tema, le enseñaban cómo dividirlo en dos o más párrafos, de tal forma que cada uno tuviera sólo un tema. Instrucciones similares se aplican a las relaciones. Cada relación normalizada tiene sólo un tema. Cualquier relación que tenga dos o más temas debe dividirse en dos o más relaciones, que tengan cada una un tema. Este proceso es la esencia de la normalización. Cuando encontramos una relación con anomalías de modificación las eliminamos separando la relación en dos o más, de tal forma que cada una contenga sólo un tema.

Sin embargo, cada vez que dividimos una relación podemos crear restricciones de integridad referencial. Por lo tanto, recuerde verificar estas restricciones cada vez que divida una relación en dos o más.

En el resto de este capítulo aprenderá diversas reglas acerca de la normalización. Todas se refieren a instancias especiales de los procesos que acabamos de describir.

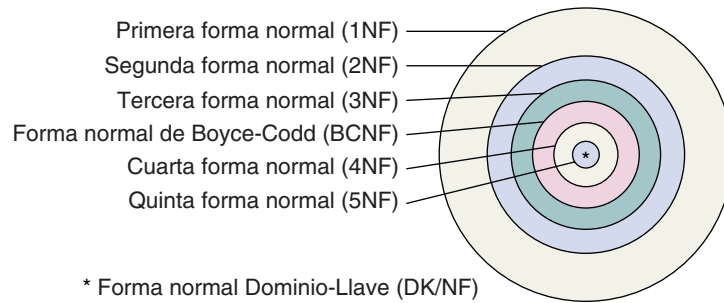
CLASES DE RELACIONES

Las relaciones se pueden clasificar por tipos de anomalías de modificación a las cuales son vulnerables. En la década de 1970, los teóricos relacionales investigaron acerca de estos tipos. Cuando alguien encontraba una anomalía, la clasificaba y pensaba en una manera de prevenirla. Cada vez que esto sucedía se mejoraban los criterios de diseño de las relaciones. Estas clases de relaciones, así como las técnicas para prevenir anomalías, se denominan **formas normales**. Dependiendo de su estructura, una relación puede estar en la primera forma normal, en la segunda forma normal, o en alguna otra.

En un trabajo posterior a su artículo de 1970, Codd y otros definieron las primera, segunda y tercera formas normales (1NF, 2NF, 3NF). Más tarde se especificó la forma normal de Boyce-Codd (BCNF), y después se definieron la cuarta y la quinta formas. Como se muestra en la figura 5-6, estas formas normales están anidadas. Esto es, una

► FIGURA 5-6

Relación de formas normales



relación en la segunda forma normal está también en primera forma normal, y una relación en 5NF (quinta forma normal) está asimismo en 4NF, BCNF, 3NF, 2NF y 1NF.

Estas formas normales fueron útiles, aunque tenían una seria limitación: ninguna teoría garantizaba que cualquiera de ellas podría eliminar todas las anomalías; cada forma podría quitar sólo algunos defectos. Sin embargo, esto cambió cuando en 1981 R. Fagin definió una nueva forma normal llamada **forma normal dominio-llave (DK/NF, por sus siglas en inglés)**. En un importante artículo, Fagin mostró que una relación en DK/NF está libre de todas las anomalías de modificación, sin importar de qué tipo sean.⁴ También enseñó que cualquier relación libre de anomalías de modificación debe estar en la forma DK/NF.

Hasta que se definió DK/NF era necesario que los teóricos de bases de datos relacionales buscaran más y más anomalías y más y más formas normales. Sin embargo, la prueba de Fagin simplificó la situación. Si se puede poner una relación en DK/NF, entonces se puede asegurar que no habrá anomalías. El truco es saber cómo poner las relaciones en DK/NF.

► DE LA PRIMERA A LA QUINTA FORMAS NORMALES

Cualquier tabla de datos que cumpla con la manera en que se define una relación se dice que está en la **primera forma normal**. Recuerde que para que una tabla sea una relación, necesita cumplir lo siguiente: las celdas de la tabla deben tener un solo valor y no se permiten ni grupos ni arreglos repetidos como valores. Todas las entradas en cualquier columna (atributos) deben ser de la misma clase. Cada columna debe tener un nombre único, pero el orden de las columnas en la tabla no es importante. Finalmente, dos renglones no pueden ser idénticos en la tabla y su orden no es significativo.

La relación en la figura 5-3 está en la primera forma normal. Sin embargo, como hemos visto, las relaciones en la primera forma normal pueden tener anomalías de modificación. Para eliminarlas, separamos la relación en dos o más relaciones. Cuando lo hacemos, las nuevas relaciones están en otra forma normal —lo cual justamente depende de las anomalías que hemos eliminado, así como de aquellas a las que son vulnerables las nuevas relaciones.

SEGUNDA FORMA NORMAL

Para entender la segunda forma normal considere la relación ACTIVIDADES en la figura 5-4; tiene anomalías de modificación similares a las que examinamos anteriormente. Si eliminamos o borramos el tuple para Estudiante 175, perderemos el hecho de que squash cuesta \$50. Asimismo, no podemos registrar una actividad hasta que un estudiante se inscriba en ella. De esta forma, la relación sufre ambas anomalías: eliminación e inserción.

⁴R. Fagin "A Normal Form for Relational Databases That Is Based on Domains and Keys", *ACM Transactions on Database Systems*, septiembre de 1981, pp. 387-415.

El problema con esta relación es que tiene una dependencia que involucra sólo una parte de la llave, que es la combinación (EID, Actividad); pero la relación contiene una dependencia, Actividad $\rightarrow$ Cuota. El determinante de esta dependencia (Actividad) es sólo parte de la llave (EID, Actividad). En este caso, decimos que, en la tabla, Cuota es *parcialmente dependiente* de la llave. No debe haber anomalías de modificación si Cuota es dependiente de toda la llave. Para eliminar las anomalías debemos separar las relaciones en dos.

Este ejemplo conduce a la definición de la segunda forma normal: *una relación se encuentra en la segunda forma normal si todos los atributos que no son llaves son dependientes de todos los atributos de la llave*. De acuerdo con esta definición, si la relación tiene un solo atributo como su llave, entonces automáticamente está en la segunda forma normal. Debido a que la llave es sólo un atributo, en forma predeterminada cada atributo que no es llave depende de *todos* los atributos de la llave; no puede haber dependencia parcial. Por lo tanto, la segunda forma normal sólo se aplica a relaciones que tienen llaves compuestas.

ACTIVIDADES se puede descomponer para formar dos relaciones en la segunda forma normal. Las relaciones son las mismas que las de la figura 5-5, a saber: ACT-ESTUD y COSTO-ACT. Sabemos que las nuevas relaciones están en la segunda forma normal porque ambas tienen llaves de un solo atributo.

TERCERA FORMA NORMAL

Las relaciones en la segunda forma normal también pueden tener anomalías. Considere la relación VIVIENDA de la figura 5-7(a). La llave es EID y las dependencias funcionales son EID $\rightarrow$ Edificio y Edificio $\rightarrow$ Cuota. Estas dependencias surgen porque cada estudiante vive en un solo edificio y ese edificio tiene sólo una cuota. Cada uno de los que viven en Randolph Hall, por ejemplo, paga \$3200 por trimestre.

► FIGURA 5-7

Eliminación de la dependencia transitiva:
(a) relación con dependencia transitiva, y
(b) relaciones que eliminan la dependencia transitiva

VIVIENDA (EID, Edificio, Cuota)

Llave: EID

Dependencias

funcionales: Edificio $\rightarrow$ Cuota

EID $\rightarrow$ Edificio $\rightarrow$ Cuota

EID	Edificio	Cuota
100	Randolph	3200
150	Ingersoll	3100
200	Randolph	3200
250	Pitkin	3100
300	Randolph	3200

(a)

VIVIENDA-ESTUD (EID, Edificio)

Llave: EID

EID	Edificio
100	Randolph
150	Ingersoll
200	Randolph
250	Pitkin
300	Randolph

EDIF-CUOTA (Edificio, Cuota)

Llave: Edificio

Edificio	Cuota
Randolph	3200
Ingersoll	3100
Pitkin	3100

(b)

Puesto que EID determina Edificio y Edificio determina Cuota, entonces indirectamente $EID \rightarrow Cuota$. A un arreglo de dependencias funcionales como a éste se le llama **dependencia transitiva**, ya que EID determina Cuota a través del atributo Edificio.

La llave de VIVIENDA es EID, el cual es un atributo individual y, por lo tanto, la relación está en la segunda forma normal (tanto Edificio como Cuota están determinados por EID). A pesar de esto, VIVIENDA tiene anomalías debido a la dependencia transitiva.

¿Que pasaría si eliminamos el segundo tuple que se muestra en la figura 5-7(a)? Perderíamos no sólo el hecho de que Estudiante 150 vive en Ingersoll Hall, sino también que cuesta \$3100 vivir ahí. Ésta es una anomalía de eliminación. ¿Cómo podemos registrar el hecho de que Cuota para Carrigg Hall es de \$3500? No podemos, hasta que un estudiante decida mudarse ahí. Ésta es una anomalía de inserción.

Para eliminar las anomalías de una relación en la segunda forma normal debe suprimirse la dependencia transitiva, lo que conduce a una definición de la tercera forma normal: *una relación se encuentra en la tercera forma normal si está en la segunda forma normal y no tiene dependencias transitivas*.

La relación VIVIENDA se puede dividir en dos relaciones en la tercera forma normal. Esto se ha hecho para las relaciones VIVIENDA-ESTUD (EID, Edificio) y EDIF-CUOTA (Edificio, Cuota) en la figura 5-7(b).

La relación ACTIVIDAD en la figura 5-3 también tiene una dependencia transitiva. En ACTIVIDAD, EID determina Actividad y Actividad determina Cuota. Por lo tanto, ACTIVIDAD no está en la tercera forma normal. Descomponiendo ACTIVIDAD en las relaciones ACT-ESTUD (Actividad, EID) y COST-ACT (Cuota, Actividad) se eliminan las anomalías.

FORMA NORMAL BOYCE-CODD

Por desgracia, incluso las relaciones en la tercera forma normal pueden tener anomalías. Considere la relación ASESOR en la figura 5-8(a). Suponga que los requerimientos que sustentan esta relación son que un estudiante (EID) puede tener una o más especialidades (Especialidad), una Especialidad puede tener varios miembros de la facultad (Fnombre) como asesores, y un miembro de la facultad (Fnombre) asesora en una sola área de especialidad. También suponga que dos miembros de la facultad no pueden tener el mismo nombre.

Debido a que los estudiantes pueden tener varias Especialidades, EID no determina la Especialidad. Además, ya que los estudiantes pueden tener varios asesores, EID tampoco determina Fnombre. Entonces, EID, por sí mismo, no puede ser una llave.

La combinación (EID, Especialidad) determina Fnombre y la combinación (EID, Fnombre) determina la Especialidad. De ahí que cualquiera de las combinaciones pueda ser una llave. Dos o más atributos o un conjunto de atributos que puedan ser una llave se denominan **llaves candidatas**. Cualquiera de las candidatas que se seleccione como la llave se llama **llave primaria**.

Además de las llaves candidatas, hay otra dependencia funcional que debemos considerar: Fnombre determina la Especialidad (cualquier miembro de la facultad asesora en sólo una especialidad. Por lo tanto, al proporcionar Fnombre podemos determinar la Especialidad). Así, Fnombre es un determinante.

Por definición, ASESOR está en la primera forma normal y también en la segunda, puesto que no tiene un atributo llave (todos los atributos cuando menos son parte de una llave). Asimismo, está en la tercera forma normal porque no tiene una dependencia transitiva. A pesar de esto, tiene anomalías de modificación.

Suponga que el Estudiante 300 abandona la escuela. Si borramos el tuple Estudiante 300 se perderá el hecho de que Perls asesora en psicología. Ésta es una anomalía de eliminación. De igual forma, ¿cómo podemos almacenar el hecho de que Keynes asesora en economía? No podemos, hasta que un estudiante se especialice en economía. Ésta es una anomalía de inserción.

Situaciones como la anterior conducen a la definición de la forma normal Boyce-Codd (BCNF): *una relación está en BCNF si cada determinante es una llave candidata*. ASESOR no está en BCNF porque el determinante, Fnombre, no es una llave candidata.

FIGURA 5-8

Forma Normal Boyce-Codd:
(a) relación en la tercera forma normal, pero no en la forma normal Boyce-Codd, y (b) relaciones en la forma normal Boyce-Codd

ASESOR (EID, Especialidad, Fnombre)

Llave (primaria): (EID, Especialidad)

Llave (candidata): (EID, Fnombre)

Dependencias funcionales:

Fnombre $\rightarrow$ Especialidad

EID	Especialidad	Fnombre
100	Matemáticas	Cauchy
150	Psicología	Jung
200	Matemáticas	Riemann
250	Matemáticas	Cauchy
300	Psicología	Perls
300	Matemáticas	Riemann

(a)

ASESOR-ESTUD(EID-Fnombre)
Llave: EID,Fnombre

EID	Fnombre
100	Cauchy
150	Jung
200	Riemann
250	Cauchy
300	Perls
300	Riemann

ASESOR-MATERIA (Fnombre, Materia)
Llave: Fnombre

Fnombre	Materia
Cauchy	Matemáticas
Jung	Psicología
Riemann	Matemáticas
Perls	Psicología

(b)

Al igual que con los otros ejemplos, ASESOR puede desglosarse en dos relaciones que no tengan anomalías. Por ejemplo, las relaciones ASESOR-ESTUD (EID, Fnombre) y ASESOR-MATERIA (Fnombre, Materia) no tienen anomalías.

Las relaciones en BCNF no tienen anomalías con respecto a dependencias funcionales y esto parece eliminar las anomalías de modificación. Sin embargo, pronto se descubrió que las anomalías pueden surgir de otras situaciones diferentes a las dependencias funcionales.

CUARTA FORMA NORMAL

Considere la relación ESTUDIANTE de la figura 5-9, la cual muestra la relación entre estudiantes, especialidades y actividades. Suponga que los estudiantes pueden inscribirse en varias especialidades diferentes y participar en diversas actividades también diferentes. En un caso así, la única llave es la combinación de atributos (EID, Especialidad, Actividad). La Estudiante 100 se especializa en música y contabilidad y también participa en natación y tenis. El Estudiante 150 se especializa sólo en matemáticas y participa en carreras.

¿Cuál es la relación entre EID y Especialidad? No es una dependencia funcional porque los estudiantes pueden tener varias especialidades. Un solo valor de EID puede tener varios valores de Especialidad, y un solo valor de EID puede tener varios valores de Actividad.

Esta dependencia de atributos se llama **dependencia multivaluada**. Las dependencias multivaluadas conducen a anomalías de modificación. Para empezar, observe la redundancia de los datos en la figura 5-9. La Estudiante 100 tiene cuatro registros; cada uno muestra una de sus especialidades pareadas con una de sus actividades. Si los datos fueran almacenados con menos renglones —por ejemplo, sólo con dos tuples,

► FIGURA 5-9

Relaciones con dependencias multivaluadas

ESTUDIANTE (EID, Especialidad, Actividad)
Llave: (EID, Especialidad, Actividad)

Dependencias multivaluadas: $EID \twoheadrightarrow Especialidad$
 $EID \twoheadrightarrow Actividad$

EID	Especialidad	Actividad
100	Música	Natación
100	Contabilidad	Natación
100	Música	Tenis
100	Contabilidad	Tenis
150	Matemáticas	Carreras

uno para música y natación y otro para contabilidad y tenis— las implicaciones podrían ser engañosas. *Parecería* que la Estudiante 100 nadaba sólo cuando tenía la especialidad de música, y jugaba tenis sólo cuando tenía la especialidad de contabilidad. Pero esta interpretación no es lógica. Sus especialidades y actividades son completamente independientes de cualquier otra. Para evitar esta conclusión engañosa almacenamos todas las combinaciones de especialidad y actividades.

Suponga que debido a que la Estudiante 100 decide inscribirse en esquí, agregamos el tuple [100, MÚSICA, ESQUÍ], como se muestra en la figura 5-10(a). La relación en este punto implica que la estudiante 100 esquí cuando toma la especialidad de música, pero no cuando toma la especialidad de contabilidad. Con el fin de mantener la consistencia de los datos, debemos agregar también el renglón [100, CONTABILIDAD, ESQUÍ] como en la figura 5-10(b). Ésta es una anomalía de actualización —hay que hacer demasiadas actualizaciones para realizar un simple cambio en los datos.

En general, hay una dependencia multivaluada cuando una relación tiene cuando menos tres atributos, dos de los cuales son valores múltiples, y sus valores dependen sólo de un tercer atributo. En otras palabras, en la relación R (A, B, C) existe una depen-

► FIGURA 5-10

Relaciones ESTUDIANTE con anomalías de inserción:
(a) inserción de un solo tuple, y
(b) inserción de dos tuples

ESTUDIANTE (EID, Especialidad, Actividad)
Llave: (EID, Especialidad, Actividad)

EID	Especialidad	Actividad
100	Música	Esquí
100	Música	Natación
100	Contabilidad	Natación
100	Música	Tenis
100	Contabilidad	Tenis
150	Matemáticas	Carreras

(a)

EID	Especialidad	Actividad
100	Música	Esquí
100	Contabilidad	Esquí
100	Música	Natación
100	Contabilidad	Natación
100	Música	Tenis
100	Contabilidad	Tenis
150	Matemáticas	Carreras

(b)

► FIGURA 5-11

Eliminación de una dependencia multivaluada

ESPECIALIDAD-ESTUD (EID, Especialidad)
Llave: (EID, Especialidad)

EID	Especialidad
100	Música
100	Contabilidad
150	Matemáticas

ACT-ESTUD (EID, Actividad)
Llave: (EID, Actividad)

EID	Actividad
100	Esquí
100	Natación
100	Tenis
150	Carreras

dencia multivaluada si A determina valores múltiples de B; A determina valores múltiples de C, y B y C son independientes entre sí. Como vimos en el ejemplo anterior, EID determina valores múltiples de Especialidad y también valores múltiples de Actividad, pero Especialidad y Actividad son independientes una de otra.

Nuevamente nos remitiremos a la figura 5-9. Observe cómo se escriben las dependencias multivaluadas: $EID \twoheadrightarrow Especialidad$ y $EID \twoheadrightarrow Actividad$. Esto se lee: “EID multidetermina Especialidad y EID multidetermina Actividad”. Esta relación está en BCNF (2-NF porque todo esto es llave; 3NF porque no tiene dependencias transitivas, y BCNF porque no tiene determinantes que no son llaves). Sin embargo, como podemos ver, esto tiene anomalías: si un estudiante agrega otra especialidad debemos ingresar un tuple para ésta y parearlo con cada una de las actividades del estudiante. Lo mismo se aplica si un estudiante se inscribe en una nueva actividad. Si abandona una especialidad debemos eliminar cada uno de los registros que contienen esa especialidad. Si participa en cuatro actividades, debe haber cuatro tuples que contengan la especialidad que ha dejado y deberán borrarse.

Para evitar estas anomalías tenemos que eliminar las dependencias multivaluadas. Esto se hace construyendo dos relaciones, cada una almacena datos para uno solo de los atributos multivalor. Las relaciones resultantes no tienen anomalías. Éstas son ESPECIALIDAD-ESTUD (EID, Especialidad) y ACT-ESTUD (EID, Actividad), como se muestra en la figura 5-11.

A partir de las observaciones anteriores, definimos la cuarta forma normal de la siguiente manera: *una relación se encuentra en la cuarta forma normal si está en BCNF y no tiene dependencias multivaluadas*. Después de analizar la forma dominio-llave en este capítulo, describiremos las dependencias multivaluadas en otra forma más intuitiva.

QUINTA FORMA NORMAL

La quinta forma normal se refiere a dependencias que son más raras. Esto tiene que ver con relaciones que pueden dividirse en subrelaciones, como lo hemos hecho, pero que no se pueden reconstruir. La condición bajo la cual se origina esta situación no tiene un significado claro, intuitivamente hablando. No sabemos cuáles son los resultados de estas dependencias, ni si tienen alguna consecuencia práctica. Para más información acerca de la quinta forma normal consulte el trabajo de Date que citamos al principio de este capítulo.

► FORMA NORMAL DOMINIO-LLAVE

Cada una de las formas normales ya analizadas las identificaron aquellos investigadores que encontraron anomalías con algunas relaciones, las cuales estaban en una forma normal más baja: observaron que las anomalías de modificación en las relaciones de la segunda forma normal conducen a la definición de la tercera forma normal, y así sucesivamente. Aunque cada forma normal resuelve alguno de los problemas que han sido identificados con la anterior, nadie puede saber qué problemas aún no han sido detecta-

dos. En cada etapa se ha ido progresando mediante un diseño bien estructurado de bases de datos, pero nadie puede garantizar que no se puedan encontrar más anomalías. En esta sección estudiaremos una forma normal, la cual garantiza que no habrá anomalías de ningún tipo. Cuando planteamos las relaciones de esta manera, sabemos que no pueden ocurrir incluso las anomalías más raras asociadas con la quinta forma normal.

En 1981 Fagin publicó un importante artículo en el que definió la forma normal dominio-llave (DK/NF).⁵ Mostró que una relación en DK/NF no tiene anomalías de modificación y, además, que una relación sin anomalías de modificación debe estar en DK/NF. Este hallazgo establece un límite en la definición de las formas normales, y por lo tanto no es necesaria una forma normal más alta, al menos con el fin de eliminar anomalías de modificación.

Igualmente importante es que la DK/NF sólo implica los conceptos de llave y dominio, los cuales son fundamentales y clave para los profesionales de bases de datos. Se apoyan en los productos DBMS (o cuando menos deberían hacerlo). En cierto sentido, el trabajo de Fagin formaliza y justifica lo que muchos profesionales creían intuitivamente, pero no eran capaces de expresar en forma precisa.

DEFINICIÓN

El concepto DK/NF es muy simple: una relación está en DK/NF si cada restricción en la relación es una consecuencia lógica de la definición de llaves y dominios. Ponga mucha atención en los términos importantes: restricción, llave y dominio.

La **restricción** pretende ser muy extensa. Fagin la define como cualquier regla que gobierna valores estáticos de atributos, la cual es lo suficientemente precisa como para que podamos verificar si es o no verdadera. Las reglas de edición, las restricciones de intrarrelación e interrelación, las dependencias funcionales y la dependencia multivaluada son ejemplos de estas restricciones. Fagin excluye expresamente las restricciones correspondientes a cambios en los valores de los datos, o las restricciones que dependan del tiempo. Por ejemplo, la regla “el salario de los vendedores en el periodo actual no puede ser menor al que recibían en la etapa anterior” está excluida de la definición de restricción de Fagin. Exceptuando las restricciones dependientes del tiempo, la definición de Fagin es extensa e incluyente.

Una **llave** es un identificador único de un tuple, como ya lo hemos establecido. El tercer término importante en la definición de DK/NF es el **dominio**. En el capítulo 4 establecimos que el dominio es una descripción de los valores permitidos para un atributo. Consta de dos partes: una descripción física y una semántica o lógica. La descripción física constituye un conjunto de valores que el atributo puede tener, y la lógica se refiere al significado del atributo. La prueba de Fagin se aplica a ambas partes.

De manera informal, una relación está en DK/NF si al cumplir las restricciones de la llave y del dominio da como resultado que todas las restricciones se cumplan. Además, debido a que las relaciones en DK/NF no pueden tener anomalías de modificación, el DBMS puede prohibirlas imponiendo las restricciones de dominio y de llave.

Por desgracia, no se conoce algún algoritmo para convertir una relación a DK/NF; incluso no se sabe cuáles relaciones se pueden convertir a DK/NF. Encontrar o diseñar las relaciones DK/NF es más un arte que una ciencia.

A pesar de lo anterior, en el mundo práctico del diseño de bases de datos, la DK/NF es extremadamente útil para el diseño. Si podemos definir relaciones en una forma en que las restricciones sobre ellas sean consecuencias lógicas de dominio y llaves, entonces no habrá anomalías de modificación. En muchos diseños este objetivo se puede cumplir. Cuando no es así, las restricciones se deben construir dentro de la lógica de los programas de aplicación que procesan la base de datos. Veremos más acerca de esto en el presente capítulo y en el capítulo 10.

Los siguientes ejemplos ilustran la DK/NF.

⁵ *Ibid.*

► FIGURA 5-12

Ejemplo 1 de DK/NF

ESTUDIANTE (EID, Grado, Edificio, Cuota)

Llave: EID

Restricciones: Edificio $\rightarrow$ Cuota
EID no debe empezar con el dígito 1**EJEMPLO 1 DE LA FORMA NORMAL DOMINIO-LLAVE**

Considere la relación ESTUDIANTE en la figura 5-12, la cual contiene atributos EID, Grado, Edificio y Cuota. Edificio es el lugar donde los estudiantes viven y Cuota es la cantidad que pagan por vivir en ese edificio.

EID determina funcionalmente a los otros tres atributos; por lo tanto, EID es una llave. Suponga que también sabemos, con base en la definición de requerimientos, que Edificio $\rightarrow$ Cuota y que los EID no deben empezar con 1. Si podemos expresar esas restricciones como consecuencias lógicas de dominio y definiciones de llaves, podemos estar seguros que, de acuerdo con el teorema de Fagin, no habrá anomalías de modificación. En el presente ejemplo, esto será fácil.

Para imponer la restricción de que los números de estudiantes no empiecen con 1, sólo definimos el dominio para los números de estudiantes que incorporen esta restricción (figura 5-13). Imponer la restricción de dominio garantiza que se cumplirá.

A continuación necesitamos hacer de la dependencia funcional Edificio $\rightarrow$ Cuota una consecuencia lógica de las llaves. Si Edificio fuera un atributo de llave, Edificio $\rightarrow$ Cuota podría ser una consecuencia lógica de una llave. Por lo tanto, la cuestión es cómo convertir Edificio en una llave. No puede haber una llave en ESTUDIANTE porque más de uno viven en el mismo edificio, pero puede ser una llave de su propia relación. Así, definimos la relación EDIF-CUOTA con Edificio y Cuota como sus atributos. Edificio es la llave de esta relación. Una vez definida esta nueva relación, podemos eliminar Cuota de ESTUDIANTE. El dominio final y las definiciones de relación para este ejemplo aparecen en la figura 5-13.

Este es el mismo resultado que obtenemos cuando convertimos una relación de una 2NF a una 3NF para eliminar las dependencias transitivas. Sin embargo, en este caso el proceso fue simple y el resultado más sólido. Resultó más sencillo porque no necesitamos saber que eliminábamos una dependencia transitiva. Simplemente tuvimos

► FIGURA 5-13

*Definición de dominio-llave del ejemplo 1*Definiciones de dominio

EID	en	CDDD, donde C es un dígito decimal no = 1; D = dígito decimal
NiveldeGrado	en	{'FR', 'SO', 'JR', 'SN', 'GR'}
Edificio	en	CHAR(4)
Cuota	en	DEC(4)

Definiciones de relación y llave

ESTUDIANTE (EID, NiveldeGrado, Edificio)
Llave: EID

EDIF-CUOTA
(Edificio, Cuota)
Llave: Edificio

► FIGURA 5-14

Ejemplo 2 de DK/NF

```

PROFESOR (FID, Fnombre, Clase, EID, Snombre)
Llave: (FID, Clase, EID)
Restricciones:  FID → Fnombre
                  Fnombre → FID
                  FID → → Clase | EID
                  Fnombre → → Clase | EID
                  EID → FID
                  EID → Fnombre
                  EID → Snombre
                  FID debe empezar con 1; EID no debe empezar con 1
  
```

que encontrar maneras creativas de hacer que todas las restricciones fueran consecuencias lógicas de las definiciones de dominio y llave. El resultado fue más sólido porque cuando convertimos la relación a 3NF sólo sabíamos que existían menos anomalías que cuando estaba en 2NF. Para convertir la relación a DK/NF sabemos que ésta no tiene anomalías de modificación.

EJEMPLO 2 DE LA FORMA NORMAL DE DOMINIO-LLAVE

El siguiente ejemplo, más complicado, involucra la relación de la figura 5-14. La relación PROFESOR contiene datos acerca de los profesores, la materia que enseñan y los alumnos que asesoran. FID y Fnombre únicamente identifican a un profesor, y EID sólo a un estudiante, pero Enombre no necesariamente identifica a EID. Los profesores pueden impartir varias materias y asesorar a varios estudiantes, pero un estudiante es asesorado sólo por un profesor. Los FID empiezan con un 1, pero los EID no deben empezar con un 1.

Estas declaraciones se pueden expresar con mayor precisión mediante la dependencia funcional y los valores múltiples que se muestran en la figura 5-14. FID y Fnombre se determinan funcionalmente entre sí (en esencia, son equivalentes). FID y Fnombre multideterminan a Clase y a EID; EID determina funcionalmente a FID y a Fnombre, y EID, a Enombre.

En ejemplos más complicados como este es útil considerar DK/NF desde un punto de vista más intuitivo. Recuerde que la esencia de la normalización es que cada relación puede tener sólo un tema. Si lo consideramos desde esta perspectiva, hay tres temas en PROFESOR. Uno es la correspondencia entre FID y Fnombre; otro se refiere a las clases que el profesor imparte, y el tercero al número de identificación, nombre y asesor de un estudiante determinado.

La figura 5-15 muestra tres relaciones que reflejan esos temas. La relación FACULTAD representa la equivalencia de FID y Fnombre. FID es la llave y Fnombre es una llave alternativa cuyo significado es que ambos atributos son únicos en la relación. Debido a que ambos son llaves, la dependencia funcional $FID \rightarrow Fnombre$ y $Fnombre \rightarrow FID$ son consecuencias lógicas de llaves.

La relación PREPARACIÓN contiene la correspondencia de facultad y clases; muestra las materias que un profesor está preparado para impartir. La llave es la combinación (Fnombre, Clase). Se requieren ambos atributos en la llave porque un profesor puede impartir varias materias y una materia puede ser impartida por varios profesores. Por último, ESTUDIANTE representa el nombre del estudiante y del asesor para un EID en particular. Observe que cada una de estas relaciones tiene un tema individual. Éstas expresan todas las restricciones de la figura 5-14 como una consecuencia lógica de las definiciones de dominio y llaves. Estas relaciones están, por lo tanto, en DK/NF.

Observe que separando el tema PREPARACIÓN del tema ESTUDIANTE se han eliminado las dependencias multivaluadas. Cuando examinamos la cuarta forma normal, encontramos que con el fin de eliminar dependencias multivaluadas tuvimos que

► FIGURA 5-15

Definición dominio-llave del ejemplo 2

Definiciones de dominio

FID en CDDD, C = 1; D = dígito decimal
 Enombre en CHAR(30)
 Clase en CHAR(10)
 EID en CDDD, C es dígito decimal, no = 1;
 D = dígito decimal
 Enombre en CHAR(30)

Definiciones de relación y llave

FACULTAD (FID, Fnombre)
 Llave (primaria): FID
 Llave (candidata): Fnombre

 PREPARACIÓN (Fnombre, Clase)
 Llave: Fnombre, Clase

 ESTUDIANTE (EID, Enombre, Fnombre)
 Llave: EID

separar los atributos multivaluados en diferentes relaciones. Nuestro planteamiento aquí es separar una relación con varios temas en varias relaciones que tengan cada una un tema. Al hacer esto eliminamos la dependencia multivaluada. De hecho, llegamos a la misma solución usando ambos planteamientos.

EJEMPLO 3 DE LA FORMA NORMAL DOMINIO-LLAVE

El siguiente ejemplo se refiere a una situación que no fue tratada por ninguna de las otras formas normales, pero que en la práctica ocurre con frecuencia. Esta relación tiene una restricción entre los valores de los datos dentro de un tuple, la cual no es una dependencia funcional ni una dependencia multivaluada.

Considere las restricciones en la relación ESTUD-ASESOR en la figura 5-16. Contiene información acerca de un estudiante y de su asesor, o asesora. EID determina Enombre, FID, Fnombre y EstadodeGradodeFacultad y por lo tanto es la llave. FID y Fnombre identifican a un miembro de una sola facultad y son equivalentes entre sí, como en el ejemplo 2. Tanto FID como Fnombre determinan el EstadodeGradodeFacultad. Por úl-

► FIGURA 5-16

Ejemplo 3 de DK/NF

ESTUD-ASESOR (EID, Enombre, FID, Fnombre, EstadodeGradodeFacultad)

Llave: EID

Restricciones: FID → Fnombre
 Fnombre → FID
 FID y Fnombre → EstadodeGradodeFacultad
 Sólo los miembros de la facultad para graduados pueden asesorar a los estudiantes
 FID empieza con 1
 EID no debe empezar con 1
 EID de estudiantes graduados empieza con 9
 EstadodeGradodeFacultad = $\begin{cases} 0 & \text{para no graduados de la facultad} \\ 1 & \text{para graduados de la facultad} \end{cases}$

timo, el nuevo tipo de restricción es que sólo a los miembros de la facultad de graduados se les permite asesorar a los estudiantes.

Las restricciones de dominio son que EID no debe comenzar con 1, sino con 9 para estudiantes de la facultad para graduados, FID debe comenzar con 1, y EstadodeGrado-deFacultad es 0 para la facultad de no graduados y 1 para la facultad de graduados. Con estas definiciones de dominio, la restricción de que los estudiantes deben ser asesorados por los miembros de la facultad para graduados se puede expresar como una restricción en los valores de un renglón. Específicamente, si el EID comienza con 9, el valor de EstadodeGrado-deFacultad debe ser 1.

Para poner esta relación en DK/NF procederemos como en el ejemplo 2. ¿Cuáles son los temas básicos de esta relación? Hay uno concerniente al personal de la facultad, que relaciona FID, Fnombre, y el EstadodeGrado-deFacultad. Puesto que FID y Fnombre determinan el EstadodeGrado-deFacultad, ambos atributos pueden ser la llave y esta relación está en DK/NF (véase la figura 5-17).

Ahora considere los datos con respecto a estudiantes y asesores. Aunque en principio puede parecer que sólo hay un tema, el de asesoría, la restricción de que sólo los miembros de la facultad para graduados pueden asesorar a estudiantes implica otra cosa. En realidad, hay dos temas: asesoramiento a los graduados y asesoría a los no graduados. Así, la figura 5-17 contiene una relación G-ASE (Graduado-Asesor) para estudiantes graduados, y una relación NG-ASE para no graduados. Veamos las definiciones de dominio: EGID comienza con 9; Gfnombre es el Fnombre de un tuple de FACULTAD con EstadodeGrado-deFacultad igual a 1; y ENGID no debe empezar con 1 o 9. Todas las restricciones descritas en la figura 5-16 están implícitas en la llave y las definiciones de dominio en la figura 5-17. Estas relaciones están en DK/NF y no tienen anomalías de modificación.

Para resumir, en la figura 5-18 se enumeran las formas normales y se definen las características de cada una.

► FIGURA 5-17

Definición de dominio/Clave del ejemplo 3

Definición de dominio

FID	en CDDD, donde C = 1; D = dígito decimal
Fnombre	en CHAR (30)
EstadodeGrado-deFacultad	en [0, 1]
EGID	en CDDD, donde C = 9; D = dígito decimal; estudiante graduado
ENGID	en CDDD, DONDE C ≠ 1 y C ≠ 9; D = dígito decimal; estudiante no graduado
Enombre	en CHAR (30)

Definiciones adicionales de dominio

Gfnombre	en {Fnombre de FACULTAD, donde EstadodeGrado-deFacultad = 1}
----------	--

Definiciones de relación y llave

FACULTAD (FID, Fnombre, EstadodeGrado-deFacultad)
Llave: FID o Fnombre

G-ASESOR (EGID, Enombre, Gfnombre)
Llave: EGID

NG-ASESOR (ENGID, Enombre, Fnombre)
Llave: ENGID

► FIGURA 5-18

Resumen de Formas
Normales

Forma	Características que la definen
1NF	Cualquier relación.
2NF	Todos los atributos que no son llaves dependen por completo de cada llave.
3NF	No hay dependencias transitivas.
BCNF	Cada determinante es una llave candidata.
4NF	No hay dependencias multivaluadas.
5NF	No se describió en este análisis.
DK/NF	Todas las restricciones son consecuencias lógicas de los dominios y de las llaves.

► SÍNTESIS DE RELACIONES

En la sección anterior propusimos un diseño relacional a partir de una perspectiva analítica. Las preguntas que hicimos con respecto a una relación determinada fueron: ¿está en la forma correcta? ¿Tiene anomalías de modificación? Abordaremos el diseño relacional desde una perspectiva diferente: la sintética. A partir de dicha perspectiva nos preguntamos: “Considerando un conjunto de atributos con ciertas dependencias funcionales, ¿qué relaciones debemos formar?”

Primero, observemos que dos atributos, digamos A y B, pueden estar relacionados de tres formas:

1. Se determinan entre sí:

$A \rightarrow B$ y $B \rightarrow A$

Por lo tanto, A y B tienen una relación de atributos uno a uno.

2. Uno determina al otro.

$A \rightarrow B$, pero $B \not\rightarrow A$

Así, A y B tienen una relación muchos a uno.

3. No están relacionados funcionalmente.

$A \not\rightarrow B$ y $B \not\rightarrow A$

Entonces, A y B tienen una relación de atributos muchos a muchos.

RELACIONES DE ATRIBUTOS UNO A UNO

Si A determina a B, y B determina a A, los valores de los atributos tienen una relación uno a uno. Esto debe ser porque si A determina a B, la relación entre A y B es de muchos a uno. Sin embargo, también es cierto que si B determina a A la relación entre B y A debe ser de muchos a uno. Para que ambos enunciados sean verdad al mismo tiempo, la relación entre A y B debe ser realmente uno a uno (lo cual es un caso especial de muchos a uno), y la relación entre B y A es también realmente de uno a uno. Por lo tanto, la relación es uno a uno.

Este caso se ilustra mediante FID y Fnombre en los ejemplos 2 y 3 de la sección anterior, referente a la forma normal dominio-llave. Cada uno de estos atributos únicamente identifica a una persona de la facultad. Consecuentemente, un valor de FID corresponde exactamente a un valor de Fnombre y viceversa.

Se pueden esbozar tres enunciados equivalentes a partir del ejemplo de FID y Fnombre:

- Si dos atributos se determinan funcionalmente entre sí, la relación de sus valores de datos es uno a uno
- Si dos atributos identifican en forma única a la misma cosa (entidad u objeto), la relación de sus valores es uno a uno
- Si dos atributos tienen una relación uno a uno, se determinan funcionalmente entre sí

Cuando se crea una base de datos con atributos que tienen una relación uno a uno, ambos atributos deben aparecer juntos cuando menos en una relación. Otros atributos que son determinados funcionalmente por éstos (un atributo que es determinado funcionalmente por uno de ellos también lo es por los otros) pueden residir en la misma relación.

Considere FACULTAD (FID, Fnombre, EstadodeGradodeFacultad) en el ejemplo 3 de la sección anterior. FID y Fnombre se determinan entre sí. EstadodeGradodeFacultad también puede aparecer en esta relación porque está determinado por FID y Fnombre. Los atributos que no están determinados funcionalmente por éstos no pueden aparecer en una relación con ellos. Considere las relaciones FACULTAD y PREPARACIÓN en el ejemplo 2; tanto FID como Fnombre aparecen en FACULTAD, pero Clase (a partir de PREPARACIÓN) puede no ocurrir. Clase puede tener múltiples valores para un miembro de la facultad, de tal forma que Clase no depende de FID o Fnombre. Si agregamos Clase a la relación FACULTAD, la llave de FACULTAD necesitará ser cualquiera (FID, Clase) o (Fnombre, Clase). Sin embargo, en este caso FACULTAD no estaría en DK/NF porque las dependencias entre FID y Fnombre no estarían lógicamente involucradas en ninguna de las llaves posibles.

Estos enunciados se resumen en la primera columna de la figura 5-19, y las reglas de definición del registro se enumeran en la figura 5-20. Si A y B tienen una relación 1:1 pueden residir en la misma relación, digamos R. A determina a B y B determina a A. La llave de la relación puede ser A o B. Se puede agregar un nuevo atributo, C, a R, si A o B determinan funcionalmente a C.

Los atributos que tienen una relación uno a uno deben existir juntos con el fin de establecer su equivalencia (por ejemplo, FID de 198 se refiere al profesor Heart). Sin embargo, por lo general no es conveniente que aparezcan juntos en más de una relación, ya que esto provoca una duplicación de datos innecesaria. Con frecuencia uno o ambos de los atributos aparecen en otras relaciones. En el ejemplo 2, Fnombre ocurre tanto en PREPARACIÓN como en ESTUDIANTE. Aunque será posible colocar Fnombre en PREPARACIÓN y FID en ESTUDIANTE, por lo general es una práctica errónea, porque cuando los atributos se parean en esta forma se debe seleccionar a uno para que represente el par en todas las otras relaciones. En el ejemplo 2 se seleccionó Fnombre.

RELACIONES DE ATRIBUTOS MUCHOS A UNO

Si el atributo A determina a B, pero B no determina a A, la relación entre los valores de los datos es de muchos a uno. En la relación del asesor en el ejemplo 2, EID determina a FID. Varios estudiantes (EID) son asesorados por un miembro de la facultad (FID), pero cada estudiante es asesorado sólo por un miembro de la facultad. Ésta, entonces, es una relación de muchos a uno.

Para una relación que esté en DK/NF todas las restricciones deben estar incluidas en llaves, y por lo tanto cada determinante debe ser una llave. Si A, B y C están en la misma relación, y si A determina a B, entonces A debe ser la llave (lo cual significa que

► FIGURA 5-19

Resumen de los tres tipos de relaciones de atributos

	Tipo de relación del atributo		
	Uno a uno	Muchos a uno	Muchos a muchos
Definición de la relación*	R(A,B)	S(C,D)	T(E,F)
Dependencias	$A \rightarrow B$	$C \rightarrow D$	$E \twoheadrightarrow F$
	$B \rightarrow A$	$D \twoheadrightarrow C$	$F \twoheadrightarrow E$
Llave	A o B	C	(E,F)
Regla para agregar otro atributo	$A \text{ o } B \rightarrow C$	$C \rightarrow E$	$(E,F) \rightarrow G$

* Las letras usadas en estas definiciones de relaciones son iguales a las que se usan en la figura 5-20.

 FIGURA 5-20

Resumen de las reglas para construir relaciones

Con referencia a las relaciones de atributo uno a uno

- Los atributos que tienen una relación uno a uno deben aparecer juntos cuando menos en una relación. Llame a la relación R y a los atributos A y B
- A o B deben ser la llave de R
- Se puede agregar un atributo a R si está determinado funcionalmente por A o B
- Un atributo que no está determinado funcionalmente por A o B no puede agregarse a R
- A y B deben aparecer juntos en R , pero no deberán aparecer juntos en otras relaciones
- A o B deben usarse consistentemente para representar el par en las relaciones diferentes a R

Con referencia a las relaciones de atributos muchos a uno

- Los atributos que tienen una relación muchos a uno pueden existir juntos en una relación. Suponga que C determina a D en la relación S
- C debe ser la llave de S
- Puede agregarse un atributo a S si está determinado por C
- Un atributo que no está determinado por C no puede agregarse a S

Con referencia a las relaciones de atributos muchos a muchos

- Los atributos que tienen una relación muchos a muchos pueden existir juntos en una relación. Suponga que dos de estos atributos, E y F , residen juntos en la relación T
- La llave de T debe ser (E, F)
- Se puede agregar un atributo a T si está determinado por la combinación (E, F)
- No se puede agregar un atributo a T si no está determinado por la combinación (E, F)
- Si se agrega un nuevo atributo, G , se expande la llave a (E, F, G) , entonces el tema de la relación ha sido cambiado. G no pertenece a T o el nombre de T se debe cambiar para reflejar el nuevo tema

también determina a C). Si por el contrario (A, B) , determina a C , entonces (A, B) debe ser la llave. En este último caso, no se permite otra dependencia funcional, como por ejemplo que A determine a B .

Usted puede aplicar estos enunciados al diseño de bases de datos en la siguiente forma: cuando construya una relación, si A determina a B los otros atributos individuales que puede agregar a la relación también deben estar determinados por A . Por ejemplo, suponga que tiene que poner EID y Edificio juntos en una relación llamada ESTUDIANTE. Puede agregar cualquier otro atributo determinado por EID, por ejemplo Enombre, a esta relación. Pero si el atributo Cuota está determinado por Edificio, usted no puede añadirlo a esta relación. Sólo podrá agregar Cuota si $EID \rightarrow Cuota$.

Estos enunciados están resumidos en la columna central de la figura 5-19. Si C y D tienen una relación N:1, pueden residir juntos en una relación, digamos S . C determinará a D , pero D no determinará a C . La llave de S será C . Otro atributo, E , se podrá agregar a S sólo si C determina a E .

RELACIONES DE ATRIBUTOS MUCHOS A MUCHOS

Si A no determina a B y B no determina a A , la relación entre sus valores de datos es muchos a muchos. En el ejemplo 2, Fnombre y Clase tienen una relación muchos a muchos. Un profesor imparte muchas clases y una clase la imparten muchos profesores. En una relación muchos a muchos ambos atributos deben ser una llave de la relación. Por ejemplo, la llave de PREPARACIÓN en el ejemplo 2 es la combinación (Fnombre, Clase).

Cuando se construyen relaciones que tienen múltiples atributos como llaves, puede sumar nuevos atributos que sean funcionalmente dependientes de todos los atributos de la llave. CantidaddeVecesImpartida es funcionalmente dependiente de ambos (Fnombre, Clase) y se puede agregar a la relación. Sin embargo, OficinadeFacultad no puede añadirse porque sólo sería dependiente de Fnombre y no de Clase. Si se necesita almacenar OficinadeFacultad en la base de datos, debe adjuntarse en la relación concerniente a facultad, no en la que se refiere a preparaciones.

Estos enunciados se resumen en la columna derecha de la figura 5-19. Si E y F tienen una relación M:N, E no determina a F, y F no determina a E. Tanto E como F se pueden poner en la relación T y, si se hace así, la llave de T podrá ser la llave compuesta (E, F). Se puede agregar un nuevo atributo, G, a T si está determinado por (E, F). No se puede agregar a T si sólo está determinado por E o F.

Considere un ejemplo similar. Suponga que agregamos NúmerodeSalón a PREPARACIÓN. ¿NúmerodeSalón está determinado funcionalmente por la llave de PREPARACIÓN (Fnombre, Clase)? Lo más probable es que no, porque un profesor puede impartir determinada clase en varios salones diferentes.

La composición (Fnombre, Clase) y NúmerodeSalón tiene una relación M:N. Puesto que es así, las reglas en la figura 5-19 se pueden aplicar, pero con E representando (Fnombre, Clase) y F representando NúmerodeSalón. Ahora podemos integrar una nueva relación, T, con los atributos Fnombre, Clase y NúmerodeSalón. La llave es entonces (Fnombre, Clase, NúmerodeSalón). En este caso, hemos creado una nueva relación con un nuevo tema. Considere la relación T, que contiene nombres de las facultades, clases y cantidad de salones. Por lo tanto, el tema de esta relación no es PREPARACIÓN, sino QUIÉN-QUÉ-DÓNDE-HA-IMPARTIDO.

Cambiar el tema puede ser o no apropiado. Si NúmerodeSalón es importante, no se necesita cambiar el tema. En este caso, PREPARACIÓN es la relación equivocada y QUIÉN-QUÉ-DÓNDE-HA-IMPARTIDO es un tema más adecuado.

Por otra parte, dependiendo de los requerimientos del usuario, PREPARACIÓN puede ser completamente adecuado así como está. Si es el caso, entonces NúmerodeSalón pertenece a la base de datos y podrá ser localizado en una relación diferente —tal vez NÚMERO-SECCIÓN, SECCIÓN-CLASE o alguna combinación similar.

► DEPENDENCIAS MULTIVALUADAS, ITERACIÓN 2

El análisis de las relaciones de valores de atributos muchos a muchos puede hacer que el concepto de dependencias multivaluadas sea más fácil de comprender. El problema con la relación ESTUDIANTE (EID, Especialidad, Actividad) en la figura 5-9 es que tiene *dos* relaciones diferentes muchos a muchos —una entre EID y Especialidad y la otra entre EID y Actividad—. Queda claro que varias especialidades de estudiantes no tienen nada que ver con sus diferentes actividades. Sin embargo, poner ambas relaciones muchos a muchos en una sola hace que parezca como si existiera alguna asociación.

Especialidad y Actividad son independientes y no habría problema si un estudiante tuviera sólo una de cada una. EID determinaría funcionalmente Especialidad y Actividad, y la relación estaría en DK/NF. En este caso, tanto las relaciones entre Especialidad y EID como entre Actividad y EID serían muchos a una.

Otra forma de percibir la dificultad es examinar la llave (EID, Especialidad, Actividad). Puesto que ESTUDIANTE tiene relaciones muchos a muchos, todos los atributos tienen que estar en la llave. Ahora, ¿qué tema representa esta llave? Podemos decir que la combinación de los estudios y las actividades de los estudiantes. Pero esto no es un hecho singular, es plural. Un renglón de esta correspondencia sólo describe parte de la combinación, y para obtener la imagen completa necesitamos todos los renglones de un estudiante específico. *En general, un renglón puede tener todos los datos del tema de la relación.* Por ejemplo, un renglón de Cliente podría tener todos los datos que queremos acerca de un cliente en particular.

Considere PREPARACIÓN en el ejemplo 2 en la sección acerca de la forma normal dominio-llave. La llave es (Fnombre, Clase). Esto quiere decir que un profesor en particular está preparado para impartir determinada clase. Sólo necesitamos un renglón para obtener toda la información (la relación puede incluir CantidaddeVecesImpartida, CalificaciónPromediodelCurso, etc.) que tenemos con respecto a la combinación de ese profesor y esa clase. Consultar más renglones no proporcionará información adicional.

Como sabe, la solución al problema de restricción a dependencias multivaluadas es dividir la relación en dos, cada una con un solo tema. ESTUD-ESPECIALIDAD muestra la combinación de un estudiante y una especialidad. Todo lo que sabemos acerca de la combinación está en un solo renglón y no obtendremos información adicional si examinamos más renglones.

► OPTIMIZACIÓN

En este capítulo estudiamos los conceptos de normalización y mostramos cómo crear tablas que estén en DK/NF. El procedimiento que empleamos por lo general es el adecuado, pero a veces el resultado de la normalización no justifica el costo. Por último, veremos dos maneras en las que lo anterior puede suceder.

DESNORMALIZACIÓN

Como mencionamos, las relaciones normalizadas evitan anomalías de modificación, y en este sentido son preferibles a las relaciones sin normalizar. Sin embargo, si la valoramos bajo otros criterios, la normalización a veces no vale la pena.

Considere esta relación:

CLIENTE (NúmeroCliente, NombreCliente, Ciudad, Estado CódigoPostal)

donde NúmeroCliente es la llave.

Esta relación no está en DK/NF porque contiene la dependencia funcional CódigoPostal $\rightarrow$ (Ciudad, Estado), lo cual no está implícito en la llave NúmeroCliente. Por lo tanto, hay una restricción no implícita por la definición de llaves.

Esta relación se puede transformar en las siguientes relaciones DK/NF:

CLIENTE (NúmeroCliente, NombreCliente, CódigoPostal)

donde la llave es NúmeroCliente

CÓDIGOS (CódigoPostal, Ciudad, Estado)

donde la llave es CódigoPostal.

Estas dos tablas están en la forma normal dominio-llave, pero es probable que no representen el mejor diseño. La tabla sin normalizar probablemente es mejor porque sería más fácil de procesar y las desventajas de duplicar los datos de Ciudad y Estado no serían muy importantes.

Para otro ejemplo de desnormalización considere la relación

COLEGIO (NombreColegio, Decano, AsistenteDecano)

y suponga que un colegio tiene un decano y de uno a tres asistentes de decano. En este caso, la llave de la tabla es (NombreColegio, AsistenteDecano). Esta tabla no está en la forma normal dominio-llave porque la restricción NombreColegio $\rightarrow$ Decano no es una consecuencia lógica de las llaves de la tabla.

COLEGIO se puede normalizar en las relaciones:

DECANO (NombreColegio, Decano)

y

ASISTENTE-DECANO (NombreColegio, AsistenteDecano)

Pero ahora cada vez que una aplicación de la base de datos necesite obtener datos del colegio, deberá leer cuando menos dos renglones de datos, o posiblemente cuatro. Una alternativa para este diseño es colocar los tres AsistentedeDecano en la tabla de COLEGIO, cada uno en un atributo por separado. La tabla sería entonces:

COLEGIO1 (NombredelColegio, Decano, AsistentedeDecano1, AsistentedeDecano2, AsistentedeDecano3)

COLEGIO1 está en DK/NF porque todos sus atributos son funcionalmente dependientes de la llave NombredelColegio. Pero algo se ha perdido. Para ver qué es, suponga que quiere determinar los nombres de los COLEGIOs que tienen una asistente de decano llamada 'Mary Abernathy'. Para hacer esto, tendría que buscar este valor en cada una de las tres columnas de AsistentedeDecano. En su consulta aparecería algo similar a lo siguiente:⁶

```
SELECT      NombredelColegio
FROM        COLEGIO1
WHERE       AsistentedeDecano1 = 'Mary Abernathy' o
           AsistentedeDecano2 = 'Mary Abernathy' o
           AsistentedeDecano3 = 'Mary Abernathy'
```

Usando el diseño normalizado con ASISTENTE-DECANO, sólo se necesitaría establecer:

```
SELECT      NombredelColegio
FROM        ASISTENTE-DECANO
WHERE       AsistentedeDecano = 'Mary Abernathy'
```

En este ejemplo hay tres soluciones posibles, cada una con ventajas y desventajas. Elegir entre ellas es un arte, no hay una regla estricta que establezca cómo seleccionarlas. La mejor opción depende de las características de procesamiento de las aplicaciones que use la base de datos.

En resumen, a veces las relaciones no son normalizadas a propósito, o son normalizadas y después se desnormalizan. Con frecuencia se hace esto para mejorar el desempeño. Siempre que los datos de dos tablas individuales se deben combinar, los DBMS tienen que ejecutar trabajo extra. En la mayoría de los casos se requieren cuando menos dos lecturas en lugar de una.

REDUNDANCIA CONTROLADA

Una de las ventajas de las relaciones normalizadas es que la duplicación de datos se minimiza (sólo los valores de la llave aparecen en más de una relación). Sin embargo, por razones de desempeño a veces es apropiado duplicar intencionalmente los datos. Considere, por ejemplo, una aplicación de pedido que accese a la tabla ARTÍCULO, la cual tiene las siguientes columnas:

```
NúmerodelaPieza
NombredelaPieza
ColordelaPieza
Descripción delaPieza
FotografiadelPieza
CantidaddeManodeObra
CantidaddelPedido
PrecioEstándar
```

⁶Estos enunciados son ejemplos de SQL, un lenguaje relacional que analizaremos con detalle en el capítulo 9. Por el momento, sólo pensaremos en ellos de manera intuitiva; en el presente capítulo aprenderá acerca de su formato.

CostoEstándar
NombredelComprador

Suponga que NúmerodePieza es la llave y que la tabla está en DK/NF. También suponga que el atributo DescripcióndePieza es un campo memo potencialmente largo, y FotografiadePieza es una columna binaria de cuando menos 256K bytes de longitud.

La aplicación de procesamiento de pedidos necesitará acceder a esta tabla para obtener NombredelaPieza, ColordelaPieza, PrecioEstándar y CantidaddeManodeObra. Suponga que no se necesita DescripcióndePieza o FotografiadePieza. Dependiendo de las características del DBMS en uso, es posible que la presencia de estas dos grandes columnas haga que el proceso sea muy lento. Si es el caso, los diseñadores de la base de datos pueden decidir duplicar algunos de los datos en una segunda tabla que contenga sólo los necesarios para el procesamiento del pedido. Ellos pueden definir una tabla como PEDIDOARTÍCULO (NúmerodePieza, NombredelaPieza, ColordelaPieza, PrecioEstándar, CantidaddeManodeObra) que se use sólo para la aplicación del procesamiento del pedido.

En este caso, los diseñadores están creando un potencial para prevenir problemas serios de integridad de datos. Necesitarán desarrollar controles programáticos y manuales para asegurar que tales problemas no ocurran. Sólo crearán ese diseño si consideran que el incremento del desempeño vale la pena el costo de los controles y el riesgo de los problemas de integridad.

Otra razón para controlar la redundancia es crear tablas que se usan sólo con el propósito de elaborar reportes y apoyar la toma de decisiones. Trataremos estos temas en el capítulo 17.

► RESUMEN

El modelo relacional es importante por dos razones: se puede usar para expresar diseños de bases de datos independientes de un DBMS, y es la base para una categoría importante de productos DBMS. La normalización se puede usar como lineamiento para verificar la pertinencia y exactitud de las relaciones.

Una relación es una tabla en dos dimensiones que tiene entradas de un solo valor. Todas las entradas en una columna determinada son de la misma clase; las columnas tienen un nombre único, y el orden de éstas no es importante. Las columnas también se llaman atributos. No existen dos renglones idénticos en la tabla y el orden de los renglones no es importante. Los renglones también se llaman tuples. Los términos *tabla*, *archivo* y *relación* son sinónimos; *columna*, *campo* y *atributo* son sinónimos, y *renglón*, *registro* y *tuple* también son sinónimos.

Una dependencia funcional es una relación entre atributos. Es funcionalmente dependiente de X si su valor determina el de Y. Un determinante es un grupo de uno o más atributos en el lado izquierdo de una dependencia funcional. Por ejemplo, si X determina a Y, entonces X es el determinante. Una llave es un grupo de uno o más atributos que únicamente identifican un tuple. Cada relación tiene cuando menos una llave; debido a que cada renglón es único; en el caso más extremo la llave es el conjunto de todos los atributos en la relación. Aunque una llave es siempre única, el determinante en una dependencia funcional no necesita serlo. Si los atributos son o no llaves, y si éstas son o no atributos no lo determina un conjunto abstracto de reglas, sino la semántica del usuario.

Cuando se actualizan, algunas relaciones experimentan consecuencias no deseadas llamadas anomalías de modificación. Ocurre una anomalía de eliminación cuando se suprime un renglón y se pierde información de dos o más entidades. Ocurre una anomalía de inserción cuando la estructura relacional obliga a la adición de hechos sobre dos entidades al mismo tiempo. Las anomalías pueden ser eliminadas dividiendo la relación en dos o más relaciones.

Existen muchos tipos de anomalías de modificación. Las relaciones se pueden clasificar por los tipos de anomalías que eliminan. Estas clasificaciones se denominan formas normales.

Por definición, cada relación está en la primera forma normal. Está en la segunda forma normal si todos los atributos que no son llaves dependen de la llave completa. Una relación está en la tercera forma normal cuando se encuentra en la segunda forma normal y no tiene dependencias transitivas. Una relación está en la forma normal Boyce-Codd si cada determinante es una llave candidata. Cuando está en la cuarta forma normal es cuando se encuentra en la forma normal Boyce-Codd y no tiene dependencias multivaluadas. La definición de la quinta forma normal es intuitivamente oscura y por lo tanto no la establecemos.

Una relación está en la forma normal dominio-llave si cada restricción en la relación es una consecuencia lógica de la definición de dominios y llaves. Una restricción es cualquier restricción de los valores estáticos de los atributos cuya veracidad puede ser evaluada. Como lo establecimos, los dominios tienen una parte física y una semántica; sin embargo, en el contexto de DK/NF los dominios sólo se refieren a la descripción física.

Una manera informal de expresar la DK/NF es decir que cada relación debe tener un tema individual. Por ejemplo, esto puede referirse a PROFESORES o ESTUDIANTES, pero no a PROFESORES y ESTUDIANTES al mismo tiempo.

La normalización es un proceso de análisis de relaciones. Es posible construir relaciones mediante un proceso de síntesis considerando las relaciones entre los atributos. Si dos atributos se determinan funcionalmente entre sí, tienen una relación uno a uno. Si uno determina funcionalmente al otro, pero no a la inversa, los atributos tienen una relación uno a muchos. Si ninguno de los atributos determina a otro, entonces tienen una relación muchos a muchos. Estos hechos se pueden usar cuando se construyen relaciones, como se resume en la figura 5-20.

En algunos casos la normalización no es pertinente. Si una tabla se separa en dos o más se crean restricciones de integridad referencial. Si el costo del procesamiento extra de las dos tablas y sus restricciones integrales es más alto que el beneficio de evitar anomalías de modificación, entonces no se recomienda la normalización. Además, en algunos casos es preferible crear columnas repetidas en lugar de usar las técnicas de normalización estándar, y en otros casos se usa la redundancia controlada para mejorar el desempeño.

► PREGUNTAS DEL GRUPO I

- 5.1 ¿Qué restricciones se deben colocar en una tabla para que sea considerada una relación?
- 5.2 Defina los siguientes términos: *relación*, *tuple*, *atributo*, *archivo*, *registro*, *campo*, *tabla*, *renglón*, *columna*.
- 5.3 Defina *dependencia funcional*. Dé un ejemplo sobre dos atributos que tengan dependencia funcional, y otro de dos atributos que no tengan dependencia funcional.
- 5.4 Si EID determina funcionalmente a Actividad, ¿significa que sólo puede existir un valor de EID en la relación? ¿Por qué?
- 5.5 Defina *determinante*.
- 5.6 Proporcione un ejemplo de una relación que tenga una dependencia funcional en la cual el determinante posea dos o más atributos.
- 5.7 Defina *llave*.
- 5.8 Si EID es la llave de una relación, ¿es un determinante? ¿Puede un valor determinado de EID ocurrir más de una vez en la relación?
- 5.9 ¿Qué es una anomalía de eliminación? Dé un ejemplo diferente al que se menciona en este texto.
- 5.10 ¿Qué es una anomalía de inserción? Mencione un ejemplo diferente al de este texto.

- 5.11 Explique la relación de las siguientes formas normales: primera, segunda, tercera, Boyce-Codd, cuarta, quinta y de dominio-llave.
- 5.12 Defina la *segunda forma normal*. Mencione un ejemplo sobre la relación en 1NF, pero no en 2NF. Transforme la relación en relaciones en la forma 2NF.
- 5.13 Defina la *tercera forma normal*. Proporcione un ejemplo de la relación en 2NF, pero no en 3NF. Transforme la relación en relaciones en la forma 3NF.
- 5.14 Defina la forma *BCNF*. Dé un ejemplo sobre la relación en 3NF, pero no en BCNF. Transforme la relación en relaciones en la forma BCNF.
- 5.15 Defina la *dependencia multivaluada*. Mencione un ejemplo.
- 5.16 ¿Por qué las dependencias multivaluadas no son un problema en las relaciones con sólo dos atributos?
- 5.17 Defina la *cuarta forma normal*. Mencione un ejemplo de la relación en BCNF, pero no en 4NF. Transforme la relación en relaciones en 4NF.
- 5.18 Defina la *forma normal dominio-llave*. ¿Por qué es importante?
- 5.19 Transforme la siguiente relación a DK/NF. Establezca las suposiciones apropiadas acerca de dependencias funcionales y dominios.

EQUIPO (Fabricante, Modelo, Fecha de Adquisición, Nombre del Comprador, Teléfono del Comprador, Localización de la Planta, Ciudad, Estado, Código Postal)

- 5.20 Transforme la siguiente relación a DK/NF. Establezca las suposiciones apropiadas acerca de las dependencias funcionales y los dominios.

FACTURA (Número, Nombre del Cliente, Número de Cliente, Dirección del Cliente, Número del Artículo, Precio del Artículo, Cantidad de Artículo, Número de Vendedor, Nombre del Vendedor, Subtotal, Impuesto, Adeudo Total)

- 5.21 Conteste otra vez la pregunta 5.20, pero agregue el atributo Estado de los Impuestos del Cliente (0 si no está exento, 1 si está exento). También agregue la restricción de que no habrá impuesto si Estado de los Impuestos del Cliente = 1.
- 5.22 Dé un ejemplo, diferente al de este texto, en el cual juzgue que la normalización no vale la pena. Muestre las relaciones y justifique su diseño.
- 5.23 Explique dos situaciones en las que los diseñadores de las bases de datos intencionalmente deciden crear datos duplicados. ¿Cuál es el riesgo de tal diseño?

► PREGUNTAS DEL GRUPO II

- 5.24 Considere la siguiente definición de relación y los datos de muestra:

Relación PROYECTO

ProyectoID	Nombre del Empleado	Salario del Empleado
100A	Jones	64K
100A	Smith	51K
100B	Smith	51K
200A	Jones	64K
200B	Jones	64K
200C	Parks	28K
200C	Smith	51K
200D	Parks	28K

Proyecto(ProyectoID, Nombre del Empleado, Salario del Empleado)
Donde ProyectoID es el nombre de un proyecto de trabajo

NombredelEmpleado es el nombre de un empleado que trabaja en el proyecto

SalariodelEmpleado es el salario del empleado cuyo nombre es NombredelEmpleado

Suponga que todas las dependencias funcionales y restricciones son evidentes en estos datos, ¿cuál de los siguientes enunciados es cierto?

- ProyectoID $\rightarrow$ NombredelEmpleado
- ProyectoID $\rightarrow$ SalariodelEmpleado
- (ProyectoID, NombredelEmpleado) $\rightarrow$ SalariodelEmpleado
- NombredelEmpleado $\rightarrow$ SalariodelEmpleado
- SalariodelEmpleado $\rightarrow$ ProyectoID
- SalariodelEmpleado $\rightarrow$ (ProyectoID, NombredelEmpleado)

Conteste las siguientes preguntas:

- ¿Cuál es la llave de PROYECTO?
 - ¿Todos los atributos que no son llaves (si los hay) dependen por completo de las llaves?
 - ¿En cuál forma normal está PROYECTO?
 - Describa dos anomalías de modificación que tiene PROYECTO
 - ¿ProyectoID es un determinante?
 - ¿NombredelEmpleado es un determinante?
 - ¿(ProyectoID, NombredelEmpleado) es un determinante?
 - ¿SalariodelEmpleado es un determinante?
 - ¿Tiene esta relación una dependencia transitiva? Si es así, ¿cuál es?
 - Rediseñe esta relación para eliminar las anomalías de modificación
- 5.25 Considere la siguiente definición de relación y datos muestra:

Relación PROYECTO-HORAS

NombredelEmpleado	ProyectoID	TareasID	Teléfono	HorasTotales
Don	100A	B-1	12345	12
Don	100A	P-1	12345	12
Don	200B	B-1	12345	12
Don	200B	P-1	12345	12
Pam	100A	C-1	67890	26
Pam	200A	C-1	67890	26
Pam	200D	C-1	67890	26

HORAS-PROYECTO (NombredelEmpleado, ProyectoID, TareasID, Teléfono, HorasTotales) Donde NombredelEmpleado es el nombre de un empleado

ProyectoID es el nombre de un proyecto

TareasID es el nombre estándar de la tarea

Teléfono es el número telefónico del empleado

HorasTotales son las horas que ha trabajado el empleado en este proyecto

Suponga que todas las dependencias funcionales y restricciones son evidentes en estos datos, ¿cuáles de los siguientes enunciados son verdaderos?

- NombredelEmpleado $\rightarrow$ ProyectoID
- NombredelEmpleado $\rightarrow$ ProyectoID
- NombredelEmpleado $\rightarrow$ TareasID

- d. $\text{Nombre del Empleado} \twoheadrightarrow \text{TareasID}$
- e. $\text{Nombre del Empleado} \rightarrow \text{Teléfono}$
- f. $\text{Nombre del Empleado} \rightarrow \text{Horas Totales}$
- g. $(\text{Nombre del Empleado}, \text{ProyectoID}) \rightarrow \text{Horas Totales}$
- h. $(\text{Nombre del Empleado}, \text{Teléfono}) \rightarrow \text{TareasID}$
- i. $\text{ProyectoID} \rightarrow \text{TareasID}$
- j. $\text{TareasID} \rightarrow \text{ProyectoID}$

Conteste las siguientes preguntas:

- k. Enumere todos los determinantes
- l. ¿Esta relación contiene una dependencia transitiva? Si es así, ¿cuál es?
- m. ¿Esta relación contiene una dependencia multivalor? Si es así, ¿cuáles son los atributos no relacionados?
- n. Describa la anomalía de eliminación que presenta esta relación
- o. ¿Cuántos temas contiene esta relación?
- p. Rediseñe esta relación para eliminar las anomalías de modificación. ¿Cuántas relaciones usó? ¿Cuántos temas contiene cada una de sus nuevas relaciones?

5.26 Considere la siguiente definición de dominio, relación y llave:

Definiciones de dominio

Nombre del Empleado	en	CHAR(20)
Número Telefónico	en	DEC(5)
Nombre del Equipo	en	CHAR(10)
Localización	en	CHAR(7)
Costo	en	CURRENCY (MONEDA)
Fecha	en	YYMMDD (AAMMDD)
Hora	en	HHMM donde HH es el valor entre 00 y 23, y MM entre 00 y 59

Definiciones de relación, llave y restricción.

EMPLEADO (Nombre del Empleado, Número Telefónico)

Llave: Nombre del Empleado

Restricciones: $\text{Nombre del Empleado} \rightarrow \text{Número Telefónico}$

EQUIPO (Nombre del Equipo, Localización, Costo)

Llave: Nombre del Equipo

Restricciones: $\text{Nombre del Equipo} \rightarrow \text{Localización}$

$\text{Nombre del Equipo} \rightarrow \text{Costo}$

CITA (Fecha, Hora, Nombre del Equipo, Nombre del Empleado)

Llave: (Fecha, Hora, Nombre del Equipo)

Restricciones: $(\text{Fecha}, \text{Hora}, \text{Nombre del Equipo}) \rightarrow \text{Nombre del Empleado}$

- a. Modifique las definiciones para agregar esta restricción: un empleado no puede anotarse en más de una cita
- b. Defina horario nocturno para referirse a las horas entre las 21:00 y las 5:00. Agregue un atributo Tipo Empleado cuyo valor sea 1 si el empleado trabaja durante la noche. Cambie este diseño para forzar la restricción de que sólo los empleados que trabajan en la noche pueden programar citas nocturnas

► PREGUNTAS DEL PROYECTO FIREDUP

FiredUp alquila a un grupo de diseñadores de bases de datos (¡quienes deben estar muy motivados!) para que creen las siguientes relaciones con el fin de que dicha base sirva para hacer el seguimiento de sus estufas, reparaciones y datos de los clientes. Vea el proyecto al final de los capítulos 1 al 3 si necesita revisar las necesidades de los clientes. Para cada una de las siguientes relaciones, especifique llaves candidatas, dependencias funcionales, dependencias multivaluadas (si las hay). Justifique estas especificaciones a menos que sean muy obvias. Considerando sus especificaciones acerca de las llaves, ¿qué forma normal tiene cada relación? Transforme cada una en dos o más relaciones que estén en la forma normal dominio-llave. Indique la llave primaria de cada tabla, llaves candidatas, llaves externas y especifique cualquier restricción de integridad referencial.

Para contestar estas preguntas, suponga lo siguiente:

- El tipo de estufa y la versión determinan la capacidad del tanque
- Una estufa puede ser reparada muchas veces, pero nunca más de una en un día determinado
- Cada estufa tiene su propia factura de reparación
- Una estufa puede estar registrada con diferentes usuarios, pero nunca al mismo tiempo
- Una estufa tiene muchos componentes y cada uno se puede usar en varias estufas. Por lo tanto, FiredUp conserva los registros acerca de los tipos de partes o refacciones, tales como la *válvula del quemador*, y no acerca de las partes, como por ejemplo las válvulas del quemador número 41734 fabricado el 12 de diciembre del 2001

A. PRODUCTO1 (NúmeroSerie, Tipo, NúmeroVersión, CapacidaddelTanque, FechaFabricación, InicialesdelInspector)

B. PRODUCTO2 (NúmeroSerie, Tipo, CapacidaddelTanque, FechaReparación, NúmeroFacturadeReparación, CostodeReparación)

C. REPARACIÓN1 (NúmeroFacturadeReparación, FechaReparación, CostodeReparación, NombreReparador, NúmeroTeléfonoReparador)

D. REPARACIÓN2 (NúmeroFacturadeReparación, FechaReparación, CostodeReparación, NombreReparador, NúmeroTeléfonoReparador, NúmeroSerie, Tipo, CapacidaddelTanque)

E. REPARACIÓN3 (FechaReparación, CostodeReparación, NúmeroSerie, FechaFabricación)

F. ESTUFA1 (NúmeroSerie, NúmeroFacturadeReparación, NúmerodelComponente)

G. ESTUFA2 (NúmeroSerie, NúmeroFacturadeReparación, PropietarioRegistradoID)

Suponga que es necesario registrar al propietario de la estufa, aun cuando nunca haya sido reparada.

H. Considerando los supuestos de este caso, las relaciones y los atributos en los puntos A-G, y su conocimiento acerca de negocios pequeños, construya un conjunto de relaciones dominio-llave para FiredUp. Indique llaves primarias, llaves externas y restricciones interrelacionadas.

Diseño de bases de datos utilizando modelos de entidad-relación

En el capítulo 3 analizamos las especificaciones de modelos de datos usando el modelo de entidad-relación, y en el capítulo 5 estudiamos el modelo relacional y la normalización. En el presente capítulo reunimos estos temas para ilustrar la transformación de los requerimientos de los usuarios expresados en modelos de entidad-relación en el diseño de bases de datos relacionales. Estos diseños son independientes de cualquier DBMS en particular.

Este capítulo tiene tres secciones: en la primera mostramos técnicas para transformar modelos de datos de entidad-relación en diseños relacionales. Como podrá ver, la normalización desempeña un papel importante; en la segunda sección se aplican estas técnicas para transformar cuatro estructuras de datos comunes en las aplicaciones de bases de datos; por último, este capítulo concluye con un análisis de llaves sustitutas y de valores nulos.

► TRANSFORMACIÓN DE LOS MODELOS ENTIDAD-RELACIÓN EN DISEÑOS DE BASES DE DATOS RELACIONALES

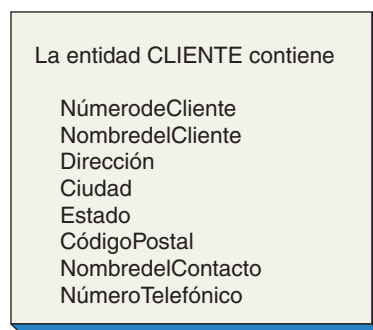
De acuerdo con el modelo entidad-relación, las cosas que los usuarios quieren registrar se representan mediante **entidades**, y las relaciones entre esas entidades se representan mediante **relaciones** definidas en forma explícita. Esta sección describe cómo transformar esas entidades y relaciones en términos del modelo relacional.

REPRESENTACIÓN DE ENTIDADES CON EL MODELO RELACIONAL

La representación de entidades por medio de un modelo relacional es directa. Comenzamos definiendo una relación para cada entidad. El nombre de la relación es el de la

► FIGURA 6-1

Representación de una entidad con una relación: (a) la entidad CLIENTE, y (b) la relación que representa la entidad CLIENTE



(a)

CLIENTE (NúmerodeCliente, NombredelCliente, Dirección, Ciudad, Estado, CódigoPostal, NombredelContacto, NúmeroTelefónico)

(b)

entidad y los atributos de la relación son los de la entidad. Entonces, examinamos cada relación de acuerdo con el criterio de normalización que vimos en el capítulo 5. Puede o no ser necesario cambiar este diseño inicial.

El ejemplo en la figura 6-1(a) es la entidad que se muestra en la figura 3-1. La entidad CLIENTE contiene los siguientes atributos: NúmerodeCliente, NombredelCliente, Dirección, Ciudad, Estado, CódigoPostal, NombredelContacto y NúmeroTelefónico. Para representar esta entidad con una relación, definimos una relación para la entidad y colocamos los atributos como columnas en la relación. Si sabemos qué atributo del modelo de datos identifica esta entidad, éste se convertirá en la llave de la relación. En caso contrario, debemos preguntar a los usuarios o investigar de otra manera los requerimientos para determinar qué atributo o atributos pueden definir una entidad. En este caso, suponemos que la llave es NúmerodeCliente. En esta figura, al igual que en otras, las llaves de las relaciones están subrayadas.

EL PAPEL DE LA NORMALIZACIÓN. Durante la fase de requerimientos, la única disposición que tiene una entidad es que es importante para el usuario. No se intenta determinar si ésta se ajusta o no a cualquiera de los criterios de normalización que se analizaron en el capítulo 5. Por lo tanto, cuando ya se definió la relación para una entidad, se deberá examinar de acuerdo al criterio de normalización.

Considere, por ejemplo, la relación CLIENTE en la figura 6-1(b). ¿Está en la forma normal dominio-llave (DK/NF)? Para saberlo, necesitamos conocer las restricciones de esta relación. Sin una descripción completa de los requisitos implícitos no conocemos todas las restricciones, como por ejemplo las de dominio. Pero podemos descubrir algunos de los requisitos a partir de los nombres de los atributos y de nuestro conocimiento sobre la naturaleza del negocio.

Primero, NúmerodeCliente determina todos los otros atributos, ya que los únicos valores de NombredelCliente, Dirección, Ciudad, Estado, CódigoPostal, NombredelContacto y NúmeroTelefónico se pueden determinar a partir de un valor específico de NúmerodeCliente. Sin embargo, hay restricciones que surgen de las otras dependencias funcionales. CódigoPostal determina Ciudad y Estado, y NombredelContacto determina NúmeroTelefónico. Para crear un conjunto de relaciones en DK/NF es necesario hacer que estas dependencias funcionales adicionales sean una consecuencia lógica de los dominios y las llaves, y esto lo podemos hacer mediante la definición de las tres relaciones que se muestran en la figura 6-2. Observe que la llave de CLIENTE es NúmerodeCliente, la de CÓDIGO-POSTAL-TABLA es CódigoPostal, y la llave de CONTACTO es NombredelContacto. Observe también las restricciones de integridad referencial.

El diseño en la figura 6-2 está en DK/NF y no hay anomalías de modificación. Esto es, podemos añadir nuevos códigos postales y contactos sin tener que agregar un clien-

► FIGURA 6-2

Representación de la entidad Cliente con relaciones en la forma normal dominio/llave

CLIENTE (Número de Cliente, Dirección, Código Postal, Nombre del Contacto)

CÓDIGO-POSTAL-TABLA (Código Postal, Ciudad, Estado)

CONTACTO (Nombre del Contacto, Número Telefónico)

Restricciones de integridad referencial:

El Código Postal en CLIENTE debe existir en Código Postal en CÓDIGO-POSTAL-TABLA
Nombre del Contacto en CLIENTE debe existir en Nombre del Contacto en CONTACTO

te con el nuevo código postal o contacto. Además, cuando se elimina el último cliente en un código postal no se pierden la ciudad y el estado. Pero, como señalamos al final del capítulo 5, la mayoría de los profesionales consideran que este diseño es demasiado puro; el hecho de dividir Código Postal, Ciudad y Estado haría que este diseño fuera difícil de usar. Por lo tanto, probablemente resultaría mejor dejar Código Postal, Ciudad y Estado en la relación CLIENTE.

¿Qué podemos decir acerca de CONTACTO? Si la relación entre un contacto y una compañía es 1:1, entonces se gana poco con colocar los datos del contacto en su propia tabla. La relación en la figura 6-1(b) es aceptable. Si la relación no es 1:1, entonces CONTACTO estaría en una entidad, con la relación apropiada en CLIENTE (la cual podría ser N:1 o 1:N) y revisando el modelo E-R adecuadamente.

En otros ejemplos es preferible el diseño DK/NF. Considere la entidad COMISIÓN-VENTAS en la figura 6-3(a). Si intentamos representar esta entidad con una relación, como se muestra en la figura 6-3(b), el resultado sería un desorden de atributos con muchas anomalías potenciales de modificación.

Esta relación obviamente contiene más de un tema. Si la examinamos veremos que contiene uno sobre vendedor, otro acerca de ventas durante algún periodo y otro tema relativo a los cheques de las comisiones de ventas. Las relaciones en DK/NF que representan esta entidad se muestran en la figura 6-3(c). Intuitivamente, este diseño parece superior al de la figura 6-3(b); es más directo y se ajusta mejor.

Para resumir el análisis, cuando se representa una entidad con el modelo relacional, el primer paso es construir una relación que tenga todos los atributos de las entidades como columnas. Después, la relación se examina en función del criterio de normalización. En muchas instancias el diseño se puede mejorar desarrollando conjuntos de relaciones en DK/NF.

Sin embargo, no siempre se prefieren las relaciones DK/NF. Si son artificiales y difíciles de trabajar, puede convenir un diseño que no sea DK/NF. El desempeño también puede ser un factor. Tener acceso a dos o tres relaciones para obtener los datos necesarios acerca de un cliente puede implicar una cantidad de tiempo prohibitiva.

Sin considerar nuestra decisión acerca de normalizar o no, deberíamos examinar cada relación de la(s) entidad(es) en función del criterio de normalización. Así que, si vamos a “pecar”, sería mejor tomar una decisión consciente y bien informada. En el proceso, también aprenderemos los tipos de modificación de anomalías a los cuales son vulnerables las relaciones.

REPRESENTACIÓN DE ENTIDADES DÉBILES. Las entidades débiles requieren un trato especial cuando se crea el diseño relacional. Recordemos que la existencia de una entidad débil depende de otra. Si la entidad débil tiene una existencia dependiente, pero no depende de un identificador, se puede representar usando las técnicas descritas en la última sección. La dependencia necesita registrarse en el diseño relacional para que ninguna aplicación provoque una entidad débil sin un padre apropiado (la entidad de la cual depende la entidad débil). Además, se deben implementar reglas del ne-

► FIGURA 6-3

Entidad con la normalización apropiada:
 (a) entidad COMISIÓN-VENTAS,
 (b) representación de COMISIÓN-VENTAS con una sola relación, y
 (c) representación de COMISIÓN-VENTAS con las relaciones forma normal dominio/llave

La entidad COMISIÓN-VENTAS contiene:

NúmerodeVendedor
 NombredelVendedor
 NúmeroTelefónico
 NúmerodeCheque
 FechadelCheque
 PeriododeComisión
 ComisiónTotaldeVentas
 CantidaddelaComisión
 CategoríaPresupuestal

(a)

COMISIÓN-VENTAS (NúmerodeVendedor, NombredelVendedor, NúmeroTelefónico, NúmerodeCheque, FechadelCheque, PeriododeComisión, ComisiónTotaldeVentas, CantidaddelaComisión, CategoríaPresupuestal)

Dependencias funcionales:

NúmeroCheque es llave

NúmerodeVendedor determina el NombredelVendedor, NúmeroTelefónico, CategoríaPresupuestal

(NúmerodeVendedor, PeriododeComisión) determina ComisiónTotaldeVentas, CantidaddelaComisión

(b)

VENDEDOR (NúmerodeVendedor, NombredelVendedor, NúmeroTelefónico, CategoríaPresupuestal)

VENDEDOR (NúmerodeVendedor, PeriododeComisión, ComisiónTotaldeVentas, CantidaddelaComisión)

CHEQUE-COMISIÓN (NúmeroCheque, FechadelCheque, NúmerodeVendedor, PeriododelaComisión)

Restricción de integridad referencial:

NúmerodeVendedor en VENTAS debe existir en NúmerodeVendedor en VENDEDOR

(NúmerodeVendedor, PeriododeComisión) en CHEQUE-COMISIÓN debe existir en (NúmerodeVendedor, PeriododeComisión) en VENTAS

(c)

gocio para que cuando se suprima el padre, la entidad débil también quede eliminada. Estas reglas se deben escribir en el diseño relacional.

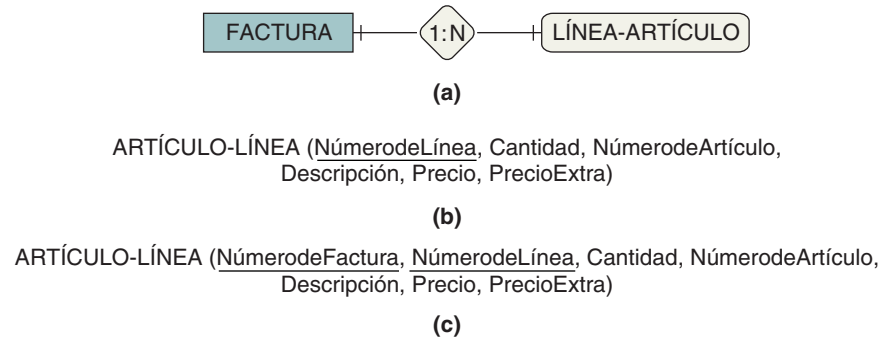
La situación anterior es un poco diferente si la entidad débil también es dependiente de un identificador. En la figura 6-4(a), ARTÍCULO-LÍNEA es una entidad débil dependiente de un identificador. Es débil porque su existencia lógica depende de FACTURA, y es dependiente de un identificador porque su identificador contiene el de FACTURA.

Cuando se crea una relación para una entidad dependiente de un identificador, debemos asegurarnos que tanto la llave del padre como la de la entidad aparezcan en la relación. Por ejemplo, considere qué pasaría si simplemente establecemos una relación para ARTÍCULO-LÍNEA y no incluimos la llave de FACTURA. Una relación como ésta se muestra en la figura 6-4(b). ¿Cuál es la llave de esta relación? Debido a que ARTÍCULO-LÍNEA es dependiente de un identificador no tiene una llave completa y, de hecho, esta relación podría tener renglones duplicados. (Lo anterior podría suceder si dos facturas tienen igual cantidad del mismo artículo en la misma línea.)

De esta forma, para una entidad débil dependiente de un identificador es necesario agregar la llave de la entidad padre a la relación de la entidad débil, y este atributo agregado sería parte de la llave de la entidad débil. En la figura 6-4(c) hemos agregado

► FIGURA 6-4

Representación relacional de una entidad débil: (a) ejemplo de entidad débil, (b) representación de la relación ARTÍCULO-LÍNEA con la llave incorrecta, y (c) relación ARTÍCULO-LÍNEA con la llave correcta



NúmerodeFactura, la llave de FACTURA, a los atributos en ARTÍCULO-LÍNEA. La llave de ARTÍCULO-LÍNEA es la compuesta {NúmerodeFactura, NúmerodeLínea}.

REPRESENTACIÓN DE RELACIONES TIENE-UN

Hay dos tipos de relaciones en el modelo E-R: las relaciones TIENE-UN entre entidades de tipos lógicos diferentes, y las relaciones ES-UN entre entidades que son subtipos de un tipo lógico común. En esta sección abordaremos las relaciones TIENE-UN; después analizaremos las relaciones ES-UN. Hay tres tipos de relaciones TIENE-UN: uno a uno, uno a muchos y muchos a muchos.

REPRESENTACIÓN DE RELACIONES UNO A UNO. La forma más simple de relación binaria es uno a uno (1:1), en la que una entidad de un tipo está relacionada únicamente con una entidad de otro tipo. En el ejemplo de EMPLEADO y AUTO, suponga que un empleado tiene asignado un automóvil y un automóvil está asignado a un empleado. En la figura 6-5 se muestra un diagrama E-R para esta relación.

La representación de una relación 1:1 con el modelo relacional es directa. Primero, cada entidad se representa con una relación y entonces la llave de una de las relaciones se coloca en la otra. En la figura 6-6(a), la llave de EMPLEADO se almacena en AUTO y en la figura 6-6(b) la llave AUTO se almacena en EMPLEADO.

Cuando la llave de la relación se almacena en una segunda relación, se llama **llave externa**. En la figura 6-6(a), NúmerodeEmpleado es una llave externa en AUTO, y en la figura 6-6(b) NúmerodeLicencia es una llave externa en EMPLEADO. En esta figura, las llaves externas se muestran en cursivas, pero algunas veces puede ver que las llaves externas se representan mediante una línea punteada. Aunque, en otros casos, las llaves externas no se señalan de ninguna manera en especial. En este texto, cuando hay peligro de confusión, se muestran las llaves externas en cursivas, pero la mayor parte del tiempo no tienen ninguna señal.

Para una relación 1:1, la llave de cualesquiera de las tablas se puede colocar como una llave externa en la otra tabla. En la figura 6-6(a) la llave externa NúmerodeLicencia se coloca en EMPLEADO. Con este diseño podemos navegar de EMPLEADO a AUTO, o de AUTO a EMPLEADO. En el primer caso tenemos un empleado y queremos saber acerca del automóvil que le fue asignado. Para obtener los datos del empleado utilizamos NúmerodeEmpleado para obtener los renglones que le corresponden, en EMPLEADO. De ahí obtenemos el NúmerodeLicencia del automóvil que tiene asignado. Entonces utilizamos este número para buscar los datos del automóvil en AUTO.

► FIGURA 6-5

Ejemplo de una relación 1:1



► FIGURA 6-6

Alternativas para la representación de relaciones 1:1: (a) colocación de la llave AUTO en EMPLEADO, y (b) colocación de la llave EMPLEADO en AUTO

EMPLEADO (NúmerodeEmpleado, NombredelEmpleado, NúmeroTelefónico. . . NúmerodeLicencia)

AUTO (NúmerodeLicencia, NúmerodeSerie, Color, Marca, Modelo. . .)

La integridad referencial contiene:

NúmerodeLicencia en EMPLEADO debe existir en NúmerodeLicencia en AUTO

(a)

EMPLEADO (NúmerodeEmpleado, NombredelEmpleado, NúmeroTelefónico. . .)

AUTO (NúmerodeLicencia, NúmerodeSerie, Color, Marca, Modelo. . . NúmerodeEmpleado)

Restricción de integridad referencial:

NúmerodeEmpleado en AUTO debe existir en NúmerodeEmpleado en EMPLEADO

(b)

Ahora consideremos la otra dirección. Suponga que tenemos un automóvil y deseamos saber a qué empleado le fue asignado. Usando el diseño en la figura 6-6(a) accedemos a la tabla EMPLEADO y buscamos el renglón que tiene el número de licencia. Los datos del empleado a quien le fue asignado el automóvil aparecen en ese renglón.

Realizamos acciones similares para dirigirnos a cualquier otra dirección para el diseño alternativo, en el cual la llave externa de NúmerodeEmpleado se coloca en AUTO. Usando este diseño, para cambiar de EMPLEADO a AUTO, accedemos directamente a la relación AUTO y buscamos el renglón en AUTO que tiene el número de empleado como valor de NúmerodeEmpleado. Para ir de AUTO a EMPLEADO buscamos el renglón en AUTO con un NúmerodeLicencia determinado. De este renglón extraemos el NúmerodeEmpleado y lo utilizamos para tener acceso a los datos del empleado en EMPLEADO. Aquí estamos usando el término *buscar* para decir que “encontramos un renglón que da un valor de una de sus columnas”. Más adelante, cuando analicemos los modelos DBMS le mostraremos cómo se hace.

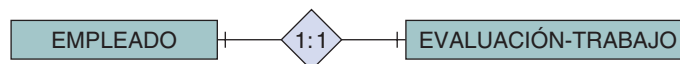
Aunque los dos diseños en la figura 6-6 son equivalentes conceptualmente hablando, pueden ser diferentes en cuanto a su funcionamiento. Por ejemplo, si una consulta en una dirección es más común que una consulta en otra, podemos inclinarnos por un diseño. También, si el producto DBMS es mucho más rápido en búsquedas de llaves primarias que en búsquedas de las llaves externas, también podemos preferir un diseño al otro.

RELACIONES 1:1 CUESTIONABLES. La figura 6-7 muestra otra relación 1:1, en la cual cada EMPLEADO tiene una EVALUACIÓN-TRABAJO y cada EVALUACIÓN-TRABAJO corresponde a un empleado en particular. Observe que las marcas verticales en la línea significan que la relación es obligatoria en ambas direcciones. Cuando la relación es 1:1 y es obligatoria en ambas direcciones, es común que los registros describan aspectos diferentes de la misma entidad, especialmente si, como en el caso en la figura 6-7, ambas tienen la misma llave. Cuando esto ocurre, por lo general los registros se deben combinar en una relación. Hay que aprender a considerar con reservas estas relaciones obligatorias 1:1.

Sin embargo, a veces la separación de una entidad en dos relaciones se puede justificar. Una justificación es el desempeño. Por ejemplo, suponga que los datos de EVALUACIÓN-TRABAJO son extensos y se usan con menos frecuencia que los otros datos del empleado. En estas circunstancias podemos almacenar apropiadamente EVALUA-

► FIGURA 6-7

Relación 1:1 sospechosa



CIÓN-TRABAJO en una tabla por separado, así las solicitudes más comunes para datos del empleado no evaluadas se podrán procesar rápidamente.

Una mayor seguridad sería la segunda justificación para separar en dos a una entidad lógica. Si el DBMS no tiene seguridad en el nivel de dato-artículo, se pueden necesitar los datos de EVALUACIÓN-TRABAJO para la separación, con el fin de evitar que los usuarios no autorizados tengan acceso a ellos. También puede ser preferible colocar EVALUACIÓN-TRABAJO en una tabla por separado y almacenarla en un disco al que puedan tener acceso sólo ciertos usuarios.

La conclusión de este análisis es que no todas las relaciones 1:1 son cuestionables, sino sólo aquellas que parecen describir diferentes aspectos de la misma entidad. Por ejemplo, la relación obligatoria 1:1 entre EMPLEADO y AUTO es bastante apropiada, porque cada relación describe una cuestión lógicamente diferente.

REPRESENTACIÓN DE RELACIONES UNO A MUCHOS. El segundo tipo de relación binaria es de uno a muchos (1:N), en la cual una entidad de un tipo se puede relacionar con muchas entidades de otro tipo. La figura 6-8 es un diagrama E-R de una relación uno a muchos entre profesores y estudiantes. En esta relación, PROFESOR se relaciona con muchos ESTUDIANTES que él o ella asesora. Como se estableció en el capítulo 3, el óvalo significa que la relación entre PROFESOR y ESTUDIANTE es opcional; por lo tanto, un profesor no necesita tener asesores. La barra que cruza la línea hasta el otro extremo significa que un renglón ESTUDIANTE debe corresponder a un renglón PROFESOR.

Los términos **padre** e **hijo** algunas veces se aplican a las relaciones 1:N. La relación padre está en *el lado uno* de la relación, y la relación hijo está en *el lado muchos*. En la figura 6-8(a) PROFESOR es la entidad padre y ESTUDIANTE es la entidad hijo.

La figura 6-8 muestra otras dos relaciones uno a muchos. En la figura 6-8(b), una entidad DORMITORIO corresponde a muchas entidades ESTUDIANTES, pero una entidad ESTUDIANTE corresponde a un DORMITORIO. Además, un dormitorio no tiene que tener estudiantes asignados, ni se requiere que un estudiante viva en un dormitorio.

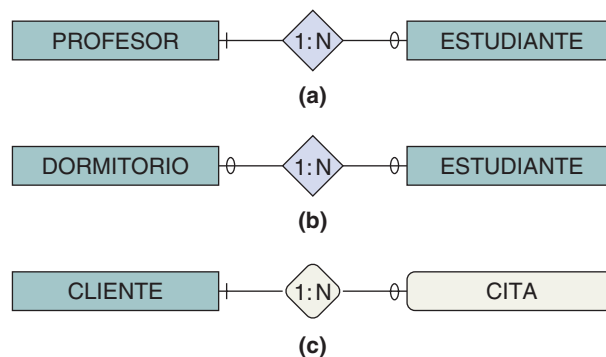
En la figura 6-8(c) un CLIENTE está relacionado con muchas entidades CITA, y una CITA en particular corresponde únicamente a un CLIENTE. Además, un CLIENTE puede tener o no una CITA, pero cada CITA debe corresponder a un CLIENTE.

Representar relaciones 1:N es simple y directo. Primero, cada entidad se representa mediante una relación, y después la llave de la relación que representa a la entidad padre se coloca en la relación que representa a la entidad hijo. De esta forma, para representar la relación ASESORADO de la figura 6-8(a), colocamos la llave de PROFESOR, Nombre del Profesor, en la relación ESTUDIANTE, como se muestra en la figura 6-9.

La figura 6-9 es un ejemplo de lo que a veces se denomina **diagrama de estructura de datos**, en el cual las relaciones se muestran mediante rectángulos con líneas que representan relaciones, y los atributos de la llave están subrayados. Un tenedor, o pata de gallo, en una línea de relación indica una relación de muchos.

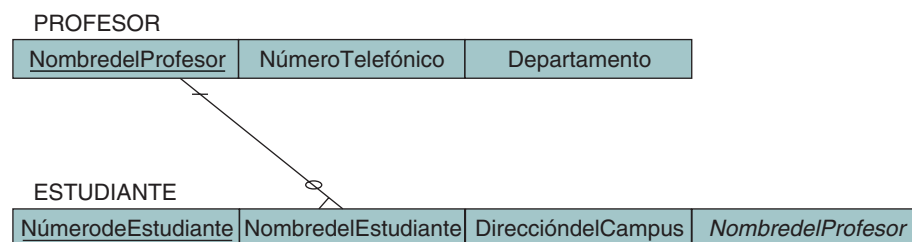
► FIGURA 6-8

Ejemplos de relaciones uno a muchos: (a) relación 1:N opcional a obligatoria, (b) relación 1:N opcional a opcional, y (c) relación 1:N con entidad débil



► FIGURA 6-9

Representación relacional de las entidades PROFESOR y ESTUDIANTE en la figura 6-8(a)



Restricción de integridad referencial:
NombredelProfesor en ESTUDIANTE debe existir en NombredelProfesor en PROFESOR

En la figura 6-9, el tenedor en el extremo de ESTUDIANTE en la línea de relación significa que puede haber muchos renglones de ESTUDIANTE por cada línea de PROFESOR. El que no aparezca un tenedor en el otro extremo significa que cada ESTUDIANTE puede ser asesorado, cuando mucho, por un PROFESOR. Al igual que con los diagramas E-R, las líneas intercaladas se usan para denotar las relaciones obligatorias y los óvalos indican las opcionales.

Observe que con NombredelProfesor almacenado como una llave externa en ESTUDIANTE, podemos procesar la relación en ambas direcciones. Con un NúmerodeEstudiante determinado, podemos buscar el renglón adecuado en ESTUDIANTE y obtener el nombre de su asesor en los datos del renglón. Para obtener los demás datos de PROFESOR usamos el nombre que obtuvimos en ESTUDIANTE con el fin de buscar el renglón adecuado en PROFESOR. Para determinar todos los estudiantes que asesora un miembro de la facultad en particular, buscamos todos los renglones de ESTUDIANTE que tengan el nombre del profesor como un valor para NombredelProfesor. Después los datos de los estudiantes se toman de esos renglones.

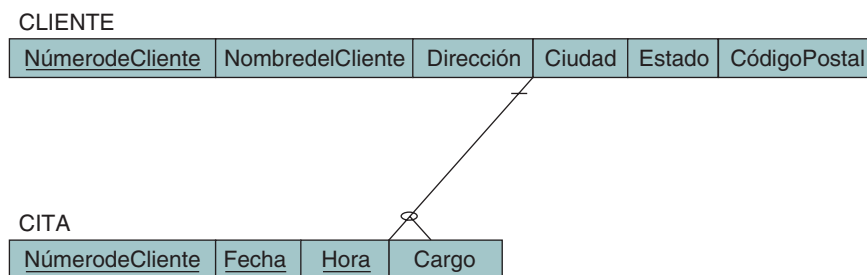
Compare esta situación con una representación de relaciones 1:1. En ambos casos almacenamos la llave de una relación como una llave externa en la segunda relación. Sin embargo, en una relación 1:1 no importa qué llave se mueva a la segunda relación. Pero sí tiene importancia en una relación 1:N. *La llave de la relación padre se debe colocar en la relación hijo.*

Para comprender mejor esto, observe qué pasaría si intentáramos colocar la llave del hijo en la relación padre (colocando NúmerodeEstudiante en PROFESOR). Debido a que los atributos en una relación sólo pueden tener un valor, en cada registro de PROFESOR sólo hay espacio para un estudiante. Por lo tanto, esta estructura no se puede usar para representar el lado “muchos” de la relación 1:N. De ahí que, para representar una relación 1:N debemos colocar la llave de la relación padre en la relación hijo.

La figura 6-10 muestra la representación de las entidades CLIENTE y CITA. Cada entidad se representa con una relación. CITA es una entidad débil dependiente de un

► FIGURA 6-10

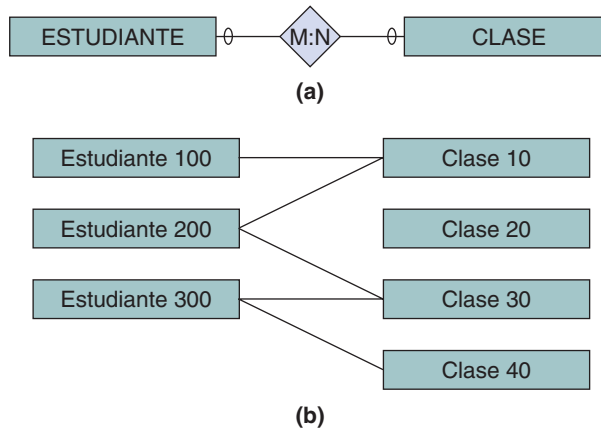
Representación relacional de la entidad débil de la figura 6-8(c)



Restricción de integridad referencial:
NúmerodeCliente en CITA debe existir en NúmerodeCliente en CLIENTE

► FIGURA 6-11

Ejemplo de una relación M:N:
(a) diagrama E-R de la relación ESTUDIANTE con CLASE, y (b) datos de muestra para la relación ESTUDIANTE con CLASE



identificador, por ende tiene una llave compuesta que consiste en la llave de la entidad de la cual depende cuando menos un atributo. Aquí la llave es (Número de Cliente, Fecha, Hora). Para representar la relación 1:N, normalmente agregaríamos la llave del padre al hijo. Sin embargo, en este caso la llave del padre (Número de Cliente) ya es parte del hijo, por lo que no es necesario agregarla.

REPRESENTACIÓN DE RELACIONES MUCHOS A MUCHOS. El tercero y último tipo de relación binaria es de muchos a muchos (M:N), en la cual una entidad de un tipo corresponde a muchas entidades del segundo tipo, y una entidad del segundo tipo corresponde a muchas entidades del primer tipo.

La figura 6-11(a) presenta un diagrama E-R de relación muchos a muchos entre estudiantes y clases. Una entidad ESTUDIANTE puede corresponder a muchas entidades CLASE, y la entidad CLASE puede corresponder a muchas entidades ESTUDIANTE. Observe que ambos participantes en la relación son opcionales: un estudiante no necesita estar inscrito en una clase, y la clase no necesita tener estudiantes. La figura 6-11(b) proporciona una muestra de datos.

Las relaciones muchos a muchos no pueden representarse directamente mediante relaciones en la misma forma que las relaciones uno a uno y uno a muchos. Para comprender por qué, intente usar la misma estrategia que aplicamos con las relaciones 1:1 y 1:N, colocando la llave de una relación como una llave externa en la otra relación. Primero, defina una relación para cada una de las entidades; nómbrelas ESTUDIANTE y CLASE. Ahora intente poner la llave de ESTUDIANTE (es decir, Número de Estudiante) en CLASE. Debido a que no se permiten valores múltiples en las celdas de una relación, sólo tenemos espacio para un Número de Estudiante, así no colocamos el registro de Número de Estudiante del segundo estudiante y de los demás.

Tendremos el mismo problema si intentamos colocar la llave de CLASE (digamos Número de Clase) en ESTUDIANTE. Podemos almacenar con facilidad el identificador de la primera clase en la cual está inscrito un estudiante; pero no tenemos espacio para almacenar el identificador de las clases adicionales.

La figura 6-12 muestra otra estrategia (*aunque es incorrecta*). En este caso, hemos almacenado un renglón en la relación CLASE para cada ESTUDIANTE inscrito en una clase, así que hay dos registros para Clase 10 y dos para Clase 30. El problema con este esquema es que duplicamos los datos de la clase y, por lo tanto, creamos anomalías de modificación. Necesitamos cambiar muchos renglones si, por ejemplo, se modifica el programa de la Clase 10. También considere las anomalías de inserción y de eliminación: ¿cómo podemos programar una clase nueva hasta que se haya inscrito un estudiante? ¿Qué pasaría si el Estudiante 300 abandonara la Clase 40? Obviamente esta estrategia no funcionaría.

► FIGURA 6-12

Representación incorrecta de una relación M:N

EID Otros datos de ESTUDIANTE	
100	...
200	...
300	...

ESTUDIANTE			
NúmerodeClase	HorariodeClase	Otros datos de clase	EID
10	10:00 MWF	...	100
10	10:00 MWF	...	200
30	3:00 TH	...	200
30	3:00 TH	...	300
40	8:00 MWF	...	300

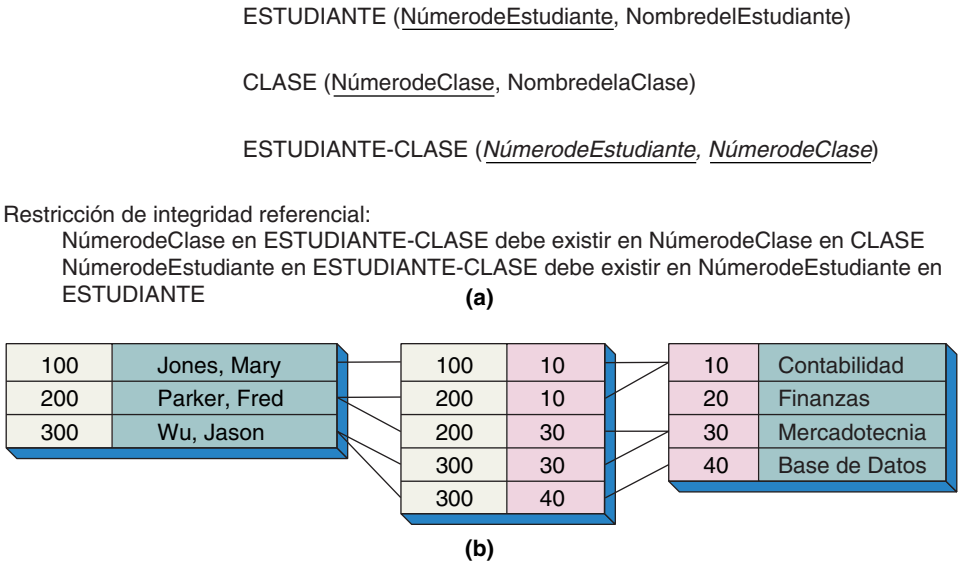
CLASE	
-------	--

La solución para este problema es crear una tercera relación que represente la propia relación. La relación ESTUDIANTE-CLASE se define en la figura 6-13(a). En la figura 6-13(b) se muestra un ejemplo. Estas relaciones se llaman **relaciones de intersección** porque cada renglón documenta la intersección de un estudiante particular con determinada clase. Observe que en la figura 6-13(b) hay un renglón en la relación de intersección para cada línea entre ESTUDIANTE y CLASE de la figura 6-11(b).

Los diagramas de la estructura de datos para la relación ESTUDIANTE-CLASE aparecen en la figura 6-14. La relación de CLASE con ESTUDIANTE-CLASE es 1:N, y la relación de ESTUDIANTE con ESTUDIANTE-CLASE también es 1:N. En esencia, hemos dividido la relación M:N en dos relaciones 1:N. La llave de ESTUDIANTE-CLASE es la combinación de las llaves de sus padres (EID, NúmerodeClase). La llave de una relación de intersección siempre es la combinación de las llaves de sus padres. También, observe que se requieren las dos relaciones de sus padres. Ahora debe existir un padre para cada valor de la llave en la relación de intersección.

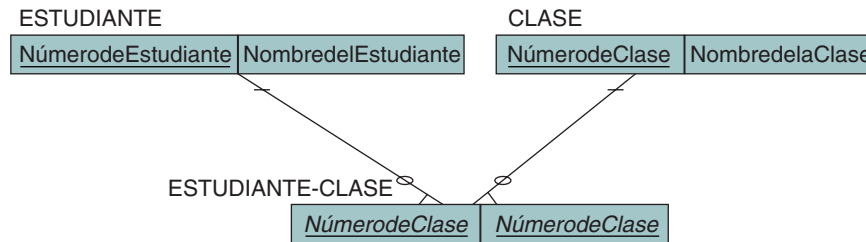
► FIGURA 6-13

Representación de una relación M:N: (a) relaciones necesarias para representar la relación ESTUDIANTE con CLASE, y (b) datos de ejemplo para la relación ESTUDIANTE con CLASE



► FIGURA 6-14

Diagrama de estructura de datos para la relación ESTUDIANTE con CLASE



REPRESENTACIÓN DE RELACIONES RECURSIVAS

Una **relación recursiva** es aquella entre entidades de la misma clase. Las relaciones recursivas no son esencialmente diferentes de otras relaciones TIENE-UN, y se pueden representar usando las mismas técnicas. Al igual que con una relación recursiva TIENE-UN, hay tres tipos de relaciones recursivas: 1:1, 1:N y N:M; la figura 6-15 muestra un ejemplo de cada una.

Considere primero la relación PATROCINADOR en la figura 6-15(a). Al igual que con una relación 1:1, una persona puede patrocinar a otra, y cada una es patrocinada como máximo por otra. La figura 6-16(a) contiene datos muestra de esta relación.

Para representar una relación recursiva 1:1 adoptamos un enfoque casi idéntico al de las relaciones regulares 1:1: podemos colocar la llave de la persona patrocinada en el renglón del patrocinador, o podemos colocar la llave del patrocinador en el renglón de la persona que es patrocinada. La figura 6-16(b) muestra la primera alternativa y la figura 6-16(c), la segunda. Ambas funcionan, y así la elección depende de aspectos del desempeño.

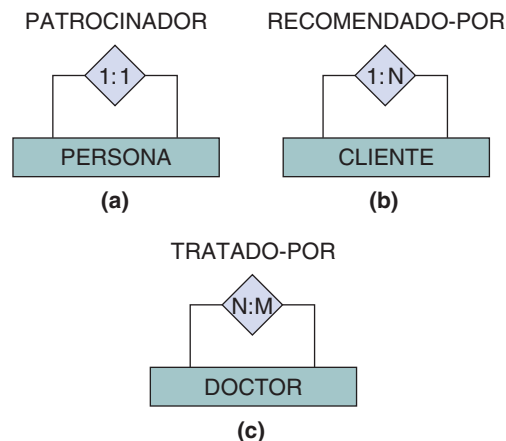
Esta técnica es idéntica a la de las relaciones no recursivas 1:1, excepto que los renglones de hijo y padre se ubican en la misma relación. Puede pensar el proceso de la forma siguiente: imagine que la relación es entre dos relaciones diferentes. Determine dónde van las llaves y después combine ambas relaciones en una sola.

Con el fin de ilustrar lo anterior, considere la relación RECOMENDADO-POR en la figura 6-15(b). Ésta es una relación 1:N, como se muestra en los datos de ejemplo en la figura 6-17(a). Cuando se colocan estos datos en una relación, un renglón representa al que recomienda, y el otro a quienes han sido recomendados. El renglón de quien recomienda adopta el papel de padre y los renglones de los recomendados, el papel de hijo. Al igual que con todas las relaciones 1:N, colocamos la llave del padre en el hijo. En la figura 6-17(b) se coloca el número del que recomienda en todos los renglones de los que han sido recomendados.

Ahora considere las relaciones recursivas M:N. La relación TRATADO-POR en la figura 6-15(c) representa la situación en la que los doctores se prescriben un tratamiento uno a otro. Los datos de ejemplo se muestran en la figura 6-18(a). Al igual que en las

► FIGURA 6-15

Ejemplo de relaciones recursivas:
(a) relación recursiva 1:1, (b) relación recursiva 1:N y (c) relación recursiva N:M



► FIGURA 6-16

Ejemplo de relación recursiva 1:1:
(a) datos de muestra para relación recursiva 1:1,
(b) primera alternativa para representar una relación 1:1, y
(c) segunda alternativa para representar una relación recursiva 1:1

Persona



(a)

Relación PERSONA1

Persona	PersonaPatrocinada
Jones	Smith
Smith	Parks
Parks	nulo
Myrtle	Pines
Pines	nulo

Restricción de integridad referencial:
PersonaPatrocinada en PERSONA1
debe existir en Persona, en PERSONA1

(b)

Relación PERSONA2

Persona	PersonaPatrocinadaPor
Jones	nulo
Smith	Jones
Parks	Smith
Myrtle	nulo
Pines	Myrtle

Restricción de integridad referencial:
PersonaPatrocinadaPor en PERSONA2
debe existir en Persona, en PERSONA2

(c)

► FIGURA 6-17

Ejemplo de una relación recursiva 1:N:
(a) datos de muestra para la relación RECOMENDADO-POR, y
(b) representación de una relación recursiva 1:N mediante una relación

Número de Cliente	Recomendación de estos clientes
100	200, 400
300	500
400	600, 700

(a)

Relación CLIENTE

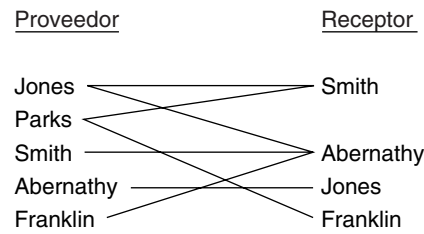
NúmerodeCliente	DatosdelCliente	RecomendadoPor
100	...	nulo
200	...	100
300	...	nulo
400	...	100
500	...	300
600	...	400
700	...	400

Restricción de integridad referencial:
RecomendadoPor en CLIENTE debe existir
en NúmerodeCliente, en CLIENTE

(b)

FIGURA 6-18

Ejemplo de una relación recursiva M:N: (a) datos de muestra para la relación TRATADO-POR, y (b) representación de una relación recursiva M:N mediante relaciones



(a)

Relación DOCTOR	
Nombre	Otros Atributos
Jones	...
Parks	...
Smith	...
Abernathy	...
O'Leary	...
Franklin	...

Relación TRATAMIENTO-INTERSECCIÓN

Doctor	Paciente
Jones	Smith
Parks	Smith
Smith	Abernathy
Abernathy	Jones
Parks	Franklin
Franklin	Abernathy
Jones	Abernathy

Restricciones de integridad referencial:

Doctor en TRATAMIENTO-INTERSECCIÓN debe existir en Nombre, en DOCTOR

Paciente en TRATAMIENTO-INTERSECCIÓN debe existir en Nombre, en DOCTOR

(b)

otras relaciones M:N, debemos crear una tabla de intersección que muestre los pares de renglones relacionados. El nombre del doctor en la primera columna es el que prescribió el tratamiento, y el del doctor en la segunda columna es aquel que lo recibió. Esta estructura se muestra en la figura 6-18(b).

Las relaciones recursivas son representadas de la misma manera que las otras relaciones. Sin embargo, los renglones de las tablas pueden representar dos funciones diferentes. Algunos son los renglones padre y los otros, los renglones hijo. Si se supone que una llave es la del padre y si el renglón no tiene padre, su valor será nulo. Si se supone que una llave es la llave hijo y el renglón no tiene hijo, su valor será nulo.

REPRESENTACIÓN DE RELACIONES TERNARIAS Y DE ORDEN SUPERIOR

Las relaciones ternarias se representan usando las técnicas que se acaban de describir, pero con frecuencia hay una consideración especial que necesita estar documentada como una regla del negocio. Considere, por ejemplo, las entidades PEDIDO, CLIENTE y VENEDORES. En la mayoría de los casos podemos tratar esta relación ternaria como dos relaciones binarias por separado.

Por ejemplo, suponga que un PEDIDO tiene un solo CLIENTE, pero que un CLIENTE puede tener muchos PEDIDOS. Por lo tanto, esta relación es binaria N:1. De manera si-

milar, suponga que PEDIDO sólo tiene un VENDEDOR y un VENDEDOR tiene muchos PEDIDOS. Esta relación también es binaria N:1.

Ambas relaciones se pueden representar usando las técnicas antes descritas. Representamos lo primero colocando la llave de CLIENTE en PEDIDO y lo segundo, colocando la llave de VENDEDOR en PEDIDO. De esta manera, hemos tratado la relación ternaria entre PEDIDO:CLIENTE:VENDEDOR como dos relaciones binarias por separado.

Sin embargo, suponga que el negocio tiene una regla que establece que cada CLIENTE puede realizar pedidos sólo con un VENDEDOR en particular. En este caso, la relación ternaria PEDIDO:CLIENTE:VENDEDOR está limitada por una relación binaria adicional N:1 entre CLIENTE y VENDEDOR. Para representar la restricción necesitamos agregar las llaves de VENDEDOR a CLIENTE. Las tres relaciones serán las siguientes:

PEDIDO (NúmeroPedido, atributos sin datos llave, NúmeroCliente, NúmeroVendedor)

CLIENTE (NúmeroCliente, atributos sin datos llave, NúmeroVendedor)

VENDEDOR (NúmeroVendedor, atributos sin datos llave)

La restricción de que sólo un VENDEDOR llame a un CLIENTE en particular significa que únicamente ciertos valores de NúmeroCliente y NúmeroVendedor pueden existir juntos en PEDIDO. Por desgracia, no hay forma de expresar esta restricción usando el modelo relacional. Debe estar documentado en el diseño; sin embargo, se debe cumplir ya sea por procedimientos almacenados (stored procedures) o mediante programas de aplicación. Véase la figura 6-19(a).

Otros tipos de restricciones binarias son: NO DEBE y DEBE CUBRIR. En una restricción NO DEBE, la relación binaria indica aquellas combinaciones que no se permite que ocurran en la relación ternaria. Por ejemplo, la relación ternaria RECETA:MEDICAMENTO:CLIENTE puede estar restringida por una relación binaria en la tabla

► FIGURA 6-19

Ejemplos de restricciones binarias en las relaciones ternarias: (a) ejemplo de una restricción binaria DEBE CUBRIR, sobre una relación ternaria

Tabla VENDEDOR

NúmeroVendedor	Otros datos sin llave
10	
20	
30	

Tabla CLIENTE

NúmeroCliente	Otros datos sin llave	NúmeroVendedor
1000		10
2000		20
3000		30

Restricción binaria DEBE

Tabla PEDIDO

NúmeroPedido	Otros datos sin llave	NúmeroVendedor	NúmeroCliente
100		10	1000
200		20	2000
300		10	1000
400		30	3000
500			2000

Sólo se permiten 20 aquí

(a)

► FIGURA 6-19

(Continuación)

((b) Ejemplo de una restricción binaria NO DEBE sobre una relación ternaria

Tabla MEDICAMENTO

NúmerodeMedicamento	Otros datos sin llave
10	
20	
30	
45	
70	
90	

Tabla ALERGIA

NúmerodeCliente	NúmerodeMedicamento	Otros datos sin llave
1000	10	
1000	20	
2000	20	
2000	45	
3000	30	
3000	45	
3000	70	

↪ Restricción binaria NO DEBE ↩

Tabla RECETA

NúmerodeReceta	Otros datos sin llave	NúmerodeMedicamento	NúmerodeCliente
100		45	1000
200		10	2000
300		70	1000
400		20	3000
500			2000

No pueden aparecer aquí ni 20 ni 45 ↪
(b)

ALERGIA, lo cual indica los medicamentos que no debe tomar un paciente. Véase la figura 6-19(b).

En una restricción DEBE CUBRIR, la relación binaria indica todas las combinaciones que tienen que aparecer en la relación ternaria. Por ejemplo, considere la relación AUTO:REPARACIÓN:TAREA. Suponga que una reparación consta de un número de TAREAs, las cuales deben ser llevadas a cabo para que tal REPARACIÓN se concrete. En este caso, en la relación AUTO-REPARACIÓN, cuando a un automóvil se le ha asignado una REPARACIÓN todas las TAREAs para que sea reparado deben aparecer como renglones en esta relación. Véase la figura 6-19(c).

Ninguno de los tres tipos de restricciones binarias analizadas se pueden representar en el diseño relacional. En lugar de esto, todas las relaciones deben ser tratadas como una combinación de relaciones binarias. Sin embargo, las restricciones se deben documentar como parte del diseño.

REPRESENTACIÓN DE RELACIONES ES-UN (SUBTIPOS)

La estrategia para representar subtipos, o relaciones ES-UN, difiere de la estrategia que se usa en una relación TIENE-UN. Considere el ejemplo de CLIENTE con los atributos NúmerodeCliente, NombredelCliente y CantidadAdeudada. Suponga que hay tres subtipos de CLIENTE, a saber, CLIENTE-PERSONA, CLIENTE-SOCIEDAD y CLIENTE-CORPORACIÓN, con los siguientes atributos:

► FIGURA 6-19

(Continuación)

(c) Ejemplo de una restricción binaria DEBE CUBRIR, con respecto a una relación ternaria

Tabla REPARACIÓN

NúmerodeReparación	Otros datos sin llave
10	
20	
30	
40	

Tabla TAREA

NúmerodeTarea	Otros datos sin llave	NúmerodeReparación
1001		10
1002		10
1003		10
2001		20
2002		20
3001		30
4001		40

Restricción binaria DEBE CUBRIR

Tabla AUTO-REPARACIÓN

NúmerodeFactura	NúmerodeReparación	NúmerodeTarea	Otros datos sin llave
100	10	1001	
200	10	1002	
300	10	1003	
400	20	2001	
500	20		

aquí debe aparecer el 2002

(c)

CLIENTE-PERSONA: Dirección, NúmerodeSeguroSocial

CLIENTE-SOCIEDAD: NombredelSocioAdministrador, Dirección, NúmerodelIdentificaciónFiscal

CLIENTE-CORPORACIÓN: PersonadeContacto, NúmeroTelefónico, NúmerodelIdentificaciónFiscal

Para representar esta estructura mediante relaciones, definimos una para el subtipo (CLIENTE) y otra para cada subtipo. Después, colocamos cada atributo del subtipo en la relación respectiva y cada uno de los atributos de los subtipos en las relaciones que los representan. En este punto, las relaciones de subtipo no tienen una llave. Para crear una, agregamos la llave del subtipo, o NúmerodeCliente, para cada uno de los subtipos. La lista final de relaciones es:

CLIENTE (NúmerodeCliente, NombredelCliente, CantidadAdeudada)

CLIENTE-PERSONA (NúmerodeCliente, Dirección, NúmerodeSeguroSocial)

CLIENTE-SOCIEDAD (NúmerodeCliente, NombredelSocioAdministrador, Dirección, NúmerodelIdentificaciónFiscal)

CLIENTE-CORPORACIÓN (NúmerodeCliente, PersonadeContacto, NúmeroTelefónico, NúmerodelIdentificaciónFiscal)

Observe que con esta estructura la relación entre un renglón en CLIENTE y un renglón en uno de los subtipos es 1:1. El cliente no tiene más de un renglón en una relación de subtipo, y cada subtipo corresponde únicamente a un renglón del subtipo. Dependiendo de las restricciones de la aplicación, es posible que un renglón en CLIENTE corresponda a múltiples renglones, cada uno de un subtipo diferente. Pero ningún renglón de CLIENTE puede corresponder a más de un renglón en la *misma* relación de subtipo.

Es posible que uno o más de los subtipos tengan una llave propia. Por ejemplo, la aplicación puede buscar o llamar a un `NúmerodeClienteCorporación` que sea distinto al `NúmerodeCliente`. En este caso, la llave de `CLIENTE-CORPORACIÓN` es `NúmerodeClienteCorporación`. Debido a que la relación entre `CLIENTE` y `CLIENTE-CORPORACIÓN` es 1:1, se puede establecer colocando la llave de una en la otra. Con frecuencia se considera un mejor diseño colocar la llave de la relación supertipo en la llave de la relación subtipo. Para este caso, la estructura de `CLIENTE-CORPORACIÓN` es:

`CLIENTE-CORPORACIÓN` (`NúmerodeClienteCorporación`, `NúmerodeCliente`, `PersonadeContacto`, `NúmeroTelefónico`, `NúmerodeIdentificaciónFiscal`).

► EJEMPLO DE DISEÑO

La figura 6-20(a) es una copia del diagrama E-R que se presentó en la figura 3-9 del capítulo 3. Contiene todos los elementos básicos usados en los diagramas E-R. Para representar este diagrama por medio de relaciones, comenzamos por establecer una relación para cada entidad. Supongamos que las llaves son las siguientes:

RELACIÓN	LLAVE
EMPLEADO	<code>NúmerodeEmpleado</code>
INGENIERO	<code>NúmerodeEmpleado</code>
AUTOBÚS	<code>NúmerodeLicencia</code>
SERVICIO	<code>NúmerodeFactura</code>
CLIENTE	<code>NúmerodeCliente</code>
CLIENTE-SERVICIO	(<code>NúmerodeFactura</code> , <code>NúmerodeCliente</code>)
CERTIFICACIÓN-INGENIERO	(<code>NúmerodeEmpleado</code> , <code>NombredeCertificación</code>)
CERTIFICACIÓN	<code>NombredeCertificación</code>

El siguiente paso es confrontar cada una de estas relaciones con el criterio de normalización. El ejemplo no indica qué atributos deben ser representados, así que no podemos determinar las restricciones. Supondremos que estas relaciones están en DK/NF, aunque en la práctica necesitaríamos verificar esa suposición con la lista de atributos y restricciones. Por ahora, nos concentraremos en la representación de las relaciones. Las relaciones y sus atributos llaves (incluyendo llaves externas) están listados en la figura 6-20(b).

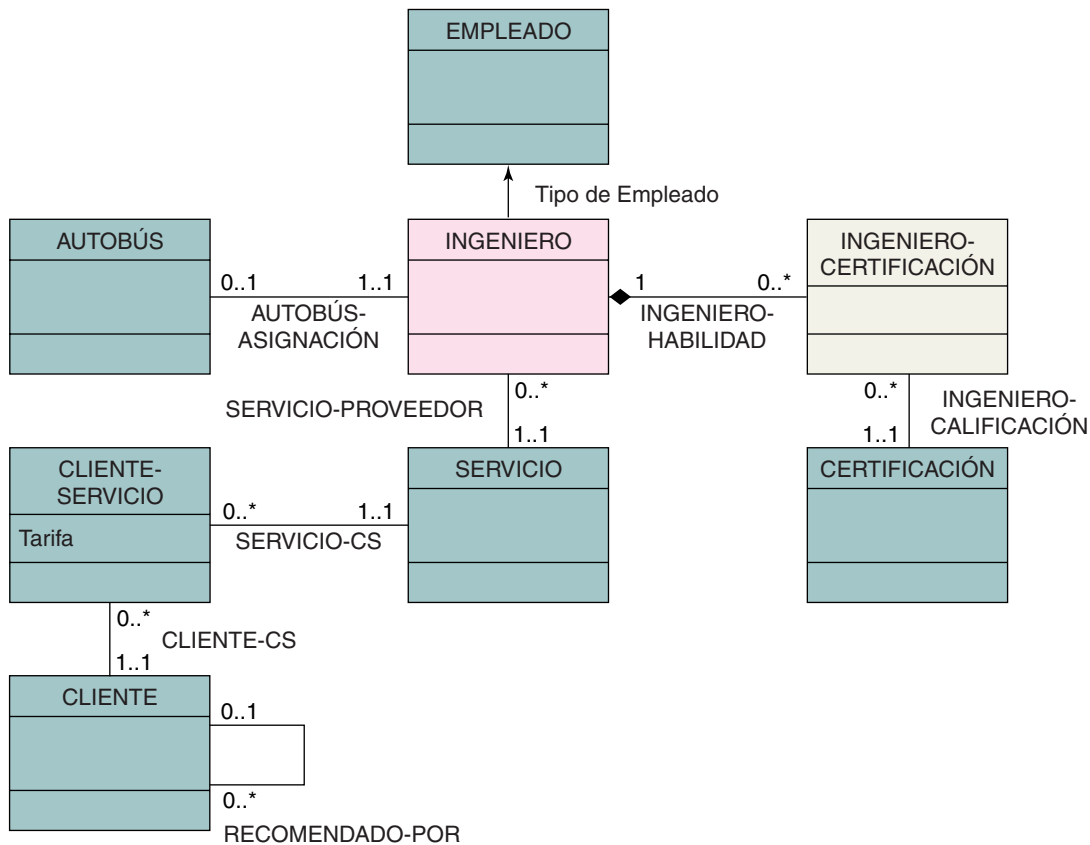
La relación entre `EMPLEADO` e `INGENIERO` ya está representada porque las relaciones tienen la misma llave, `NúmerodeEmpleado`. `INGENIERO` y `AUTOBÚS` tienen una relación 1:1 y así pueden estar relacionados por la colocación de la llave de una en la otra. Debido a que un autobús debe ser asignado a un empleado, no habrá valores nulos si se coloca `NúmerodeEmpleado` en `AUTOBÚS`, y así lo haremos.

Para la relación 1:N entre `INGENIERO` y `SERVICIO` colocamos la llave de `INGENIERO` (el padre) en `SERVICIO` (el hijo). La relación entre `SERVICIO` y `CLIENTE` es M:N, así que debemos crear una relación de intersección. Debido a que esta relación tiene un atributo, `Tarifa`, éste se agrega a la relación de intersección, `CLIENTE-SERVICIO`. Para la relación recursiva 1:N, `RECOMENDADO-POR`, agregamos el atributo `RecomendadoPor` a `CLIENTE`. El nombre *RecomendadoPor* implica, correctamente, que la llave del padre —el cliente que recomienda— se colocó en la relación.

Debido a que `INGENIERO-CERTIFICACIÓN` es dependiente de un identificador de `INGENIERO`, sabemos que `NúmerodeEmpleado` debe ser parte de su llave; así, la llave es compuesta (`NúmerodeEmpleado`, `NombredeCertificación`). La relación de dependencia es 1:N y se realizará por `NúmerodeEmpleado`. Por último, la relación entre `CERTIFICACIÓN` e `INGENIERO-CERTIFICACIÓN` es 1:N, así que normalmente agregaría-

► FIGURA 6-20

Representación relacional de un ejemplo de diagrama E-R: (a) diagrama E-R del capítulo 3, y
(b) relaciones necesarias para representar este diagrama E-R



(a)

EMPLEADO (NúmerodeEmpleado, otros atributos sin llave de EMPLEADO . . .)

INGENIERO (NúmerodeEmpleado, otros atributos sin llave de INGENIERO . . .)

AUTOBÚS (NúmerodeLicencia, otros atributos sin llave de AUTOBÚS, *NúmerodeEmpleado*)

SERVICIO (NúmerodeFactura, otros atributos sin llave de SERVICIO, *NúmerodeEmpleado*)

CLIENTE (NúmerodeCliente, otros atributos sin llave de CLIENTE, *RecomendadoPor*)

CLIENTE-SERVICIO (NúmerodeFactura, NúmerodeCliente, Tarifa)

INGENIERO-CERTIFICACIÓN (NúmerodeEmpleado, NombredCertificación, otros atributos sin llave de INGENIERO-CERTIFICACIÓN)

CERTIFICACIÓN (NombredCertificación, otros atributos sin llave de CERTIFICACIÓN)

(b)

mos la llave de CERTIFICACIÓN (el padre) a INGENIERO-CERTIFICACIÓN. Pero ésta ya es parte de la relación, así que no es necesario agregarla.

Estudie este ejemplo para asegurarse de que comprende los diferentes tipos de relaciones y cómo se expresan en términos de las relaciones. Todos los elementos del

modelo E-R se representan en la figura 6-20. Véase la pregunta 6.40 referente a las restricciones de integridad referencial.

► ÁRBOLES, REDES Y LISTAS DE MATERIALES

Aunque ni el modelo E-R ni el de objeto semántico hacen cualquier suposición acerca de los patrones de relaciones entre entidades, algunos se repiten muy a menudo, así que les han asignado nombres especiales. Estos patrones son árboles, redes simples, redes complejas y listas de materiales. Introducimos el concepto de estos patrones aquí, en el contexto del modelo E-R.

ÁRBOLES

Un **árbol**, o **jerarquía**, como a veces se le llama, es una estructura de datos en la cual sus elementos sólo tienen relaciones de uno a muchos con otro. Cada uno de los elementos tiene cuando mucho un padre. La figura 6-21 es un ejemplo de un árbol. De acuerdo con la terminología estándar, cada elemento se llama **nodo**, y las relaciones entre los elementos, **ramas**. El nodo en la parte superior del árbol se llama **raíz** (¡qué metáfora, en la naturaleza la raíz de los árboles está en la parte inferior!). En la figura 6-21 el nodo 1 es la raíz del árbol.

Cada nodo de un árbol, excepto la raíz, tiene un **padre**, el nodo inmediato superior. Así, el nodo 2 es el padre del nodo 5; el nodo 4 es el padre del nodo 8, y así sucesivamente. Como ya se mencionó, los árboles se distinguen de otras estructuras de datos en que cada nodo tiene cuando mucho un padre. Decimos que máximo un padre porque el nodo raíz no tiene padre.

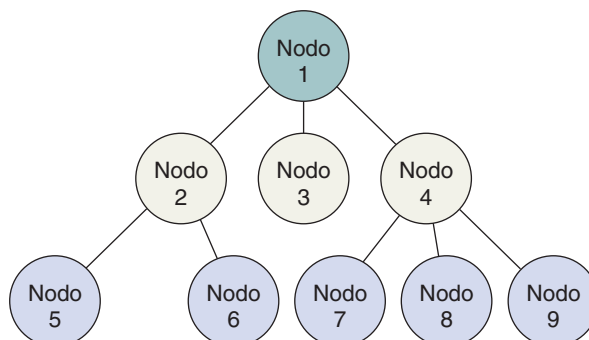
Los descendientes de un nodo se llaman **hijos**. En general, no hay un límite en el número de hijos que puede tener un nodo. El nodo 2 tiene dos hijos, los nodos 5 y 6; el nodo 3 no tiene hijo; y el nodo 4 tiene tres hijos, los nodos 7, 8 y 9. Los nodos que tienen el mismo padre se llaman **gemelos**, o **hermanos**. Por ejemplo, los nodos 5 y 6 son gemelos o hermanos.

La figura 6-22(a) ilustra un árbol de entidades en el que se pueden apreciar varias relaciones uno a muchos entre entidades en un sistema universitario. Las universidades constan de muchos departamentos, los que a su vez tienen muchos profesores y muchos empleados administrativos. Por último, los profesores asesoran a muchos estudiantes que han recibido muchas calificaciones. Hay seis tipos diferentes de entidades en esta estructura, pero todas las relaciones son 1:N.

Para representar un árbol de entidades usando el modelo relacional, simplemente aplicamos los conceptos descritos en las secciones anteriores de este capítulo. Primero transformamos cada entidad en una relación. Después examinamos las relaciones generadas en función del criterio de normalización y si es necesario se subdividen. Repre-

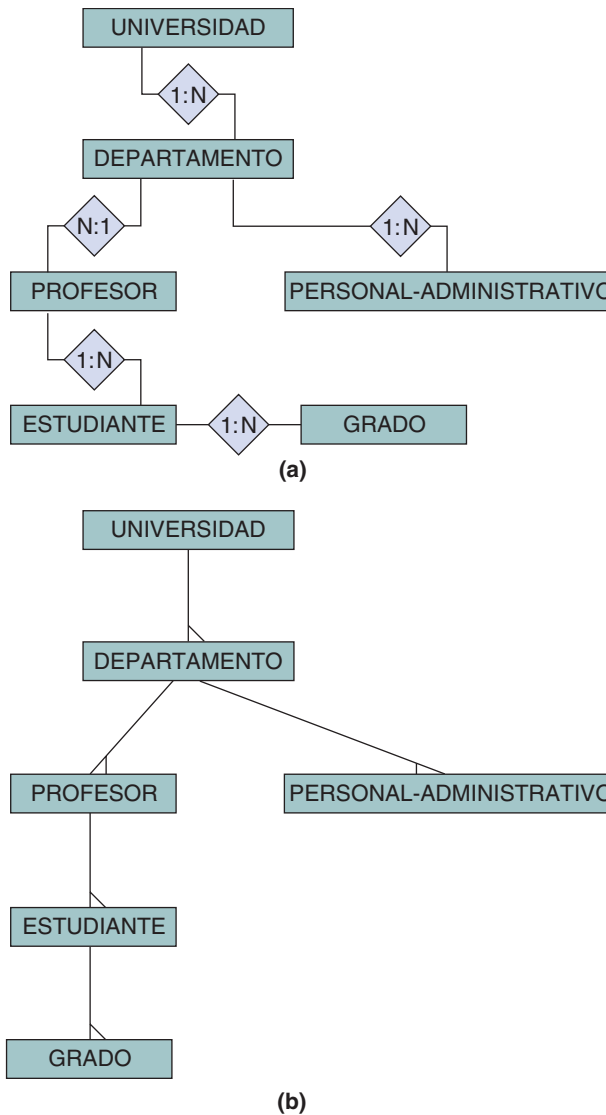
► FIGURA 6-21

Ejemplo de un árbol



► FIGURA 6-22

Representación de un árbol por medio de relaciones: (a) árbol compuesto por entidades y (b) representación de este árbol mediante relaciones



sentamos las relaciones 1:N almacenando la llave del padre en el hijo. La figura 6-22(b) es un diagrama de estructura de datos correspondiente al árbol de la figura 6-22(a).

En resumen, una jerarquía, o árbol, es un conjunto de registros organizados de tal forma que las relaciones son 1:N. Todos los registros tienen exactamente un padre, excepto la raíz. Una jerarquía se puede representar mediante un conjunto de relaciones usando los métodos antes descritos. Las jerarquías son comunes en los negocios, especialmente en aplicaciones de manufactura.

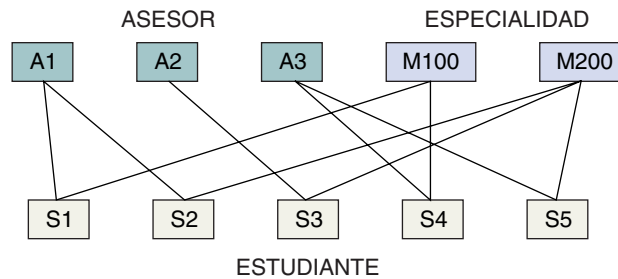
REDES SIMPLES

Una **red simple** es también una estructura de datos de elementos que sólo tienen relaciones de uno a muchos. Sin embargo, en una red simple los elementos pueden tener más de un padre, siempre y cuando sean de tipos diferentes. Por ejemplo, en la red simple que se muestra en la figura 6-23, cada entidad ESTUDIANTE tiene dos padres, una entidad ASESOR y una entidad ESPECIALIDAD. La estructura de datos que se muestra en la figura 6-23 no es un árbol porque las entidades ESTUDIANTE tienen más de un padre.

La figura 6-24(a) muestra la estructura general de esta red simple. Observe que las relaciones son de uno a muchos, pero que ESTUDIANTE tiene dos padres. En esta figu-

► FIGURA 6-23

Ejemplo de una red simple



ra, los registros padre están en la parte superior y los registros hijo, debajo de ellos. Este arreglo es conveniente pero no esencial. Puede ver una red simple representada con padres al lado o debajo del hijo. Puede identificar redes simples en estos arreglos por el hecho de que un registro de tipo simple participa como un hijo en dos (o más) relaciones uno a muchos.

Para representar una red simple de entidades con el modelo relacional seguimos los procedimientos descritos anteriormente. Primero, transformamos cada entidad en una relación y si es necesario normalizamos las relaciones. Después representamos cada relación 1:N almacenando la llave de la relación padre en la relación hijo. El resultado de este proceso para la red en la figura 6-24(a) se muestra en la figura 6-24(b).

REDES COMPLEJAS

Una **red compleja** es una estructura de datos de elementos en la cual cuando menos una de las relaciones es de muchos a muchos. La red compleja en la figura 6-25(a) ilustra la relación entre facturas, artículos de línea, partes y distribuidores. Dos de las tres relaciones son 1:N y la tercera es M:N. Debido a que hay cuando menos una relación de muchos a muchos, a esta estructura se le llama red compleja.

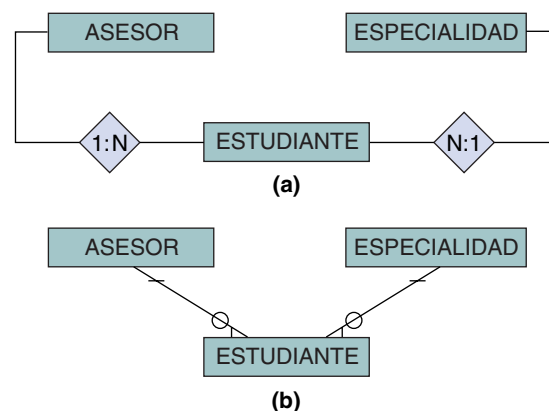
Como analizamos, las relaciones M:N no tienen representación directa en el modelo relacional. En consecuencia, antes de que se pueda almacenar esta estructura en forma relacional debemos definir una relación de intersección. En la figura 6-25(b) la relación de intersección es Proveedor-Parte.

LISTA DE MATERIALES

Una **lista de materiales** es una estructura de datos especial que ocurre con frecuencia en aplicaciones de manufactura. De hecho, estas estructuras dieron mayor impulso al desarrollo de la tecnología de bases de datos en la década de 1960.

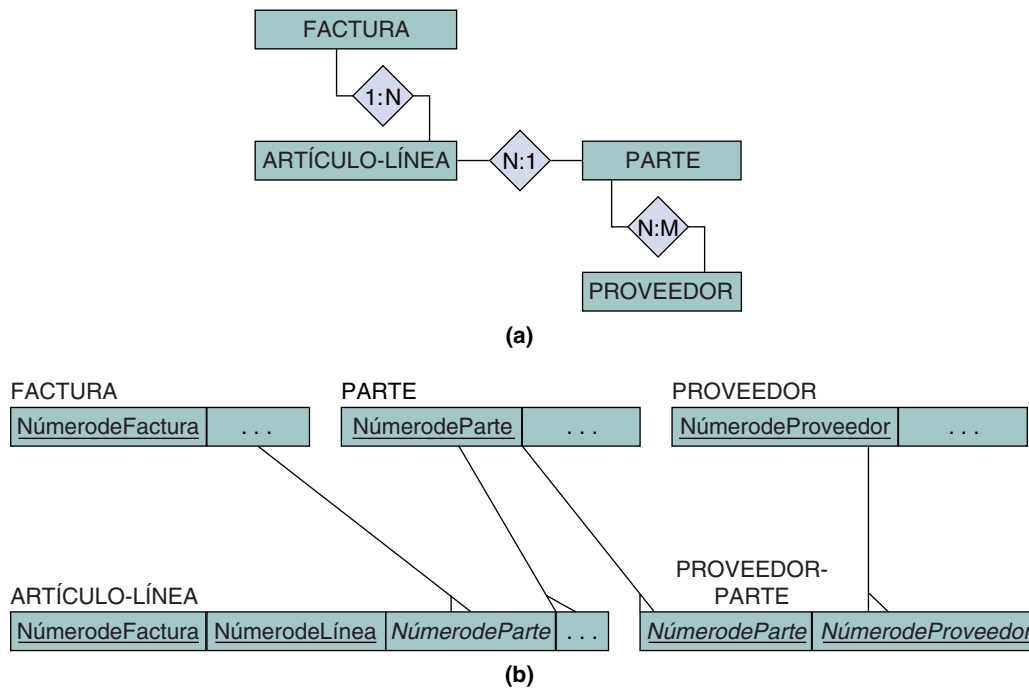
► FIGURA 6-24

Representación de una red simple por medio de relaciones: (a) red simple compuesta de entidades, y (b) su representación mediante relaciones



► FIGURA 6-25

Representación de una red compleja por medio de relaciones: (a) red compleja compuesta de entidades, y (b) su representación mediante relaciones

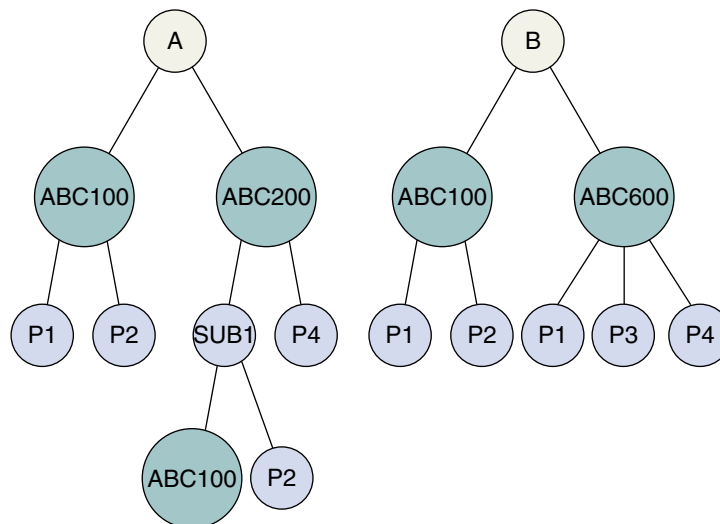


La figura 6-26 es un ejemplo de una lista de materiales, la cual muestra las partes que constituyen los productos. Cuando se observa desde el punto de vista de un producto específico, por ejemplo el producto A, esta estructura de datos es una jerarquía. Debido a que se puede usar una parte en más de un producto, esta estructura en realidad es una red. Por ejemplo, la refacción o parte ABC100 tiene dos padres: el producto A y el producto B.

Una lista de materiales se puede representar de varias formas por medio de relaciones. La más común es considerarla como una relación recursiva M:N. Una parte (producto, ensamble, subensamble, etc.) contiene muchos elementos. Al mismo tiempo, puede

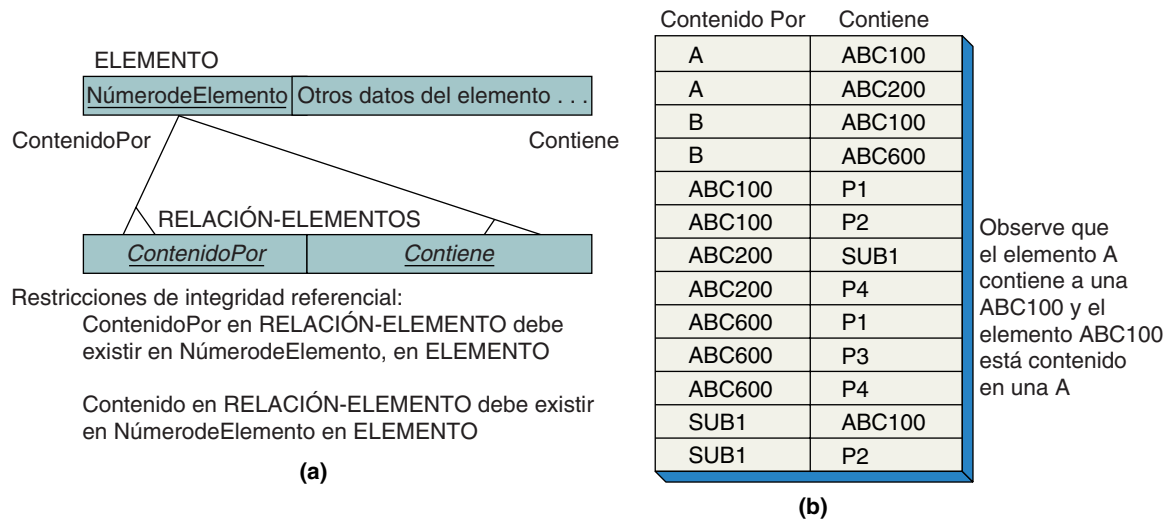
► FIGURA 6-26

Ejemplo de una lista de materiales



► FIGURA 6-27

Representación de una lista de materiales con relaciones: (a) relaciones que representan una lista de materiales, y (b) datos para la relación de intersección RELACIÓN-ELEMENTOS



ser que muchos elementos la contengan. La figura 6-27(a) muestra la estructura de datos general de la relación recursiva M:N, y la figura 6-27(b) proporciona un ejemplo de la relación de intersección creada para representar esta lista de materiales.

Antes de concluir el presente capítulo necesitamos desarrollar dos temas importantes: llaves sustitutas y valores nulos.

LLAVES SUSTITUTAS

Una **llave sustituta** es un identificador único proporcionado por el sistema, que se usa como la llave primaria de una relación. Los valores de una llave sustituta no tienen significado para los usuarios y normalmente están ocultos en las formas y los reportes. El DBMS no permitirá que se cambie el valor de una llave sustituta.

Hay dos razones para el uso de las llaves sustitutas: una pragmática y otra filosófica. Consideremos la primera.

LA RAZÓN PRAGMÁTICA. Suponga que hay una relación M:N entre las dos tablas siguientes:

MARISCO (NombredelMarisco, Tamaño, Color, Descripción)

y

PLAYA (NombredelaPlaya, Tipo, OrientacióndelaOla, TemperaturadelaCorrientedeAgua)

La relación entre estas dos tablas es de M:N porque un marisco puede ser encontrado en muchas playas, y una playa puede tener muchos tipos de mariscos. Suponga que los nombres de los mariscos son un dato de texto como *costata Cirtopleura* y están modelados con un dominio físico de texto 50. Además suponga que los nombres de la playa también están en un texto, como *Isla Perla*, *Playa Embrujada*, *Extremo Este*, y están modelados con un dominio físico de texto 75. Obviamente los índices que necesitaremos crear para reforzar la particularidad de estas columnas serán muy extensos. (Véase el apéndice A para mayor información acerca del uso de índices.)

Sin embargo, el mayor problema está en la tabla de intersección que representaría la relación M:N. Las dos columnas de esta tabla son (NombredelMarisco, NombredelaPlaya), las cuales se representan como texto 50 y texto 75, respectivamente. Por ejemplo, los datos e índices de esta tabla de intersección serán enormes. Si la base de datos

de los mariscos contiene 1000 variedades, que aparecen en un promedio de 100 playas, la tabla de intersección tendrá 100000 renglones de 125 caracteres de texto. Para hacer más complicado el problema, todos estos datos duplican NombredelMarisco, en MARISCO, y NombredelaPlaya, en PLAYA.

El almacenaje y procesamiento de estos índices se puede reducir considerablemente mediante la definición de una llave sustituta en las tablas MARISCO y PLAYA. Para lo anterior se necesita definir nuevas columnas en MARISCO y PLAYA, por ejemplo MariscoID y PlayaID, respectivamente. Estas columnas se definen en el DBMS como tipo AutoNumérico (o algo similar, dependiendo de la marca del DBMS). Cuando se define una columna de esta manera, cada vez que el DBMS crea un nuevo renglón se origina un nuevo y único valor para esa columna del nuevo renglón.

Así, un diseño corregido para estas tablas usando una llave sustituta sería:

MARISCO (MariscoID, NombredelMarisco, Tamaño, Color, Descripción)

PLAYA (PlayaID, NombredelaPlaya, Tipo, OrientacióndelaOla, TemperaturadelaCorrientedeAgua)

y

MARISCO_PLAYA_INT (MariscoID, PlayaID)

Los valores de la llave sustituta que son creados, por lo general, son enteros de 32 bites, o algo similar. Tales valores son compactos y fáciles de indizar y darán como resultado no sólo una drástica reducción del espacio en archivo, sino también un mejor desempeño.

Así, por razones pragmáticas, cada vez que una tabla tenga una llave primaria con texto extenso, o una llave compuesta de elementos grandes, considere el uso de una llave sustituta.

LA RAZÓN FILOSÓFICA. La razón filosófica para el uso de una llave sustituta es que sirve para mantener la identidad de la entidad. Cuando el usuario almacena un renglón en una tabla representa algo, que debe existir hasta que él lo elimine. La existencia de ese algo no debería depender de la presencia o la ausencia de ciertos valores de datos. Puesto que el usuario no puede cambiar las llaves sustitutas, y que garantizan su particularidad, representan la identidad de un renglón (o entidad).

Para aclarar lo anterior, considere las dos tablas en la figura 6-28(a), las cuales usan las llaves de datos más que las llaves sustitutas. Hay una relación de 1:N de ASESOR a ESTUDIANTE. La relación se realiza mediante la llave externa NombredelAsesor en ESTUDIANTE. Ahora, supongamos que Sesiones del Profesor quiere cambiar el nombre por el de Johnson. Suponga que se admite el cambio y que nuestra aplicación es demasiado fina como para diseminar ese cambio a través de una llave externa; así todos los valores de Sesión cambian a Johnson en NombredelAsesor en ESTUDIANTE. Los datos aparecerán como se muestra en la figura 6-28(b).

Tenemos un desorden. Hemos sobrecargado el nombre Johnson de tal forma que es imposible determinar a cuáles estudiantes asesora en Contabilidad, y a cuáles en Leyes. Lo que es peor, esta confusión es resultado de nuestro propio diseño, no se debe a ningún problema atribuible a los usuarios.

Quizás usted piense que el problema radica en que NombredelAsesor se debería definir como único, lo cual de hecho evitaría el problema; pero surge la pregunta: ¿quiénes somos, como creadores de bases de datos, para decir que los asesores no pueden tener nombres iguales? Si la semántica del negocio permite que los asesores tengan el mismo nombre, entonces, nuestro deber no es construir una base de datos que prohíba los nombres duplicados. Deformar el comportamiento de los usuarios en beneficio de la base de datos significa un mal diseño.

Considere este mismo escenario con llaves sustitutas. La figura 6-29(a) muestra los datos antes del cambio de nombre, y la figura 6-29(b) los muestra después del cambio. No hay confusión con las llaves porque las llaves sustitutas son usadas para representar la relación.

► FIGURA 6-28

Ejemplo de llave confusa

Relación ASESOR	
Nombre	Departamento
Johnson	Contabilidad
Eldridge	Leyes
Sesiones	Leyes

Relación ESTUDIANTE		
Nombre	Especialidad	NombredelAsesor
Franklin	Leyes	Eldridge
Jefferson	Contabilidad	Johnson
Washington	Leyes	Sesiones
Lincoln	Leyes	Sesiones

(a) Relaciones ASESOR y ESTUDIANTE sin llaves sustitutas, antes del cambio

Relación ASESOR	
Nombre	Departamento
Johnson	Contabilidad
Eldridge	Leyes
Johnson	Leyes

Relación ESTUDIANTE		
Nombre	Especialidad	NombredelAsesor
Franklin	Leyes	Eldridge
Jefferson	Contabilidad	Johnson
Washington	Leyes	Johnson
Lincoln	Leyes	Johnson

(b) Relaciones ASESOR y ESTUDIANTE sin llaves sustitutas, después del cambio

Hay muchas maneras de expresar la noción de que las llaves sustitutas protegen la identidad. El mundo de la programación orientada a objetos también crea este mismo argumento en diferentes formas para la programación orientada a objetos. Todos los argumentos provienen del hecho de que en tanto un renglón tenga un identificador sin cambios que exista mientras el renglón se conserve, entonces la identidad de éste nunca se perderá.

Sin embargo, este beneficio tiene un costo. En el diseño de la figura 6-28(a) se puede observar un renglón ESTUDIANTE y saber inmediatamente quién es el asesor. Con datos clave como éstos no hay que hacer una búsqueda en la tabla ASESOR para determinar el nombre de éste, como sucede con los datos de la figura 6-29(b). También, si una base de datos intercambia datos con otras que empleen los datos clave, entonces el uso de llaves sustitutas puede crear problemas.

¿LLAVES SUSTITUTAS O NO? ¿Cuál es la respuesta? Los expertos difieren. Mientras que ninguno argumenta razones pragmáticas para usarlas, algunos dicen que su uso debería limitarse a casos en los que se deben usar, como en el ejemplo de los mariscos. Otros dicen que las llaves sustitutas se deberían usar todas al mismo tiempo, y que los datos llave no deberían emplearse nunca.

Mi propia experiencia es usar las llaves sustitutas casi todo el tiempo. Titubeo en cuanto a su uso en tablas con datos llave que son fáciles de indexar en forma natural, como por ejemplo en una tabla de PRODUCTO que tiene una columna NúmerodeProducto con valores únicos enteros. También, algunas veces no defino las llaves sustitutas en tablas que normalmente se utilizan para intercambiar datos con otras bases de datos.

Esta política significa que algunas bases de datos que he diseñado tienen una mezcla de datos y llaves sustitutas. Esto no me gusta y puede resultar confuso. En cierta

► FIGURA 6-29

Ejemplo de una llave no confusa con llaves sustitutas

Relación ASESOR		
AsesorID	Nombre	Departamento
1	Johnson	Contabilidad
2	Eldridge	Leyes
3	Sesiones	Leyes

Relación ESTUDIANTE			
EstudianteID	Nombre	Especialidad	AsesorFK
20	Franklin	Leyes	2
21	Jefferson	Contabilidad	1
22	Washington	Leyes	3
23	Lincoln	Leyes	3

(a) Las relaciones ASESOR y ESTUDIANTE con llaves sustitutas, antes del cambio

Relación ASESOR		
AsesorID	Nombre	Departamento
1	Johnson	Contabilidad
2	Eldridge	Leyes
3	Johnson	Leyes

Relación ESTUDIANTE			
EstudianteID	Nombre	Especialidad	AsesorFK
20	Franklin	Leyes	2
21	Jefferson	Contabilidad	1
22	Washington	Leyes	3
23	Lincoln	Leyes	3

(b) Las relaciones ASESOR y ESTUDIANTE con llaves sustitutas, después del cambio

medida, esta confusión se puede evitar nombrando siempre las columnas de las llaves sustitutas (surrogate key). Normalmente uso un esquema titulado *TableNameID* o *TableName_SK* para las columnas de llaves sustitutas.

Analice este aspecto con su profesor, el cual sin duda tendrá otras ideas y opiniones. Es una decisión artística el hecho de afirmar que, en mi opinión, es necesario hacer una tabla a la vez.

VALORES NULOS

Un **valor nulo** es un valor de atributo que nunca se ha asignado. El problema de los valores nulos es que son ambiguos. Un valor nulo puede significar: (a) que el valor es desconocido, (b) que no es apropiado, o (c) que el valor se acepta en blanco. Por ejemplo, considere el atributo *FechaDeDeceso* en una relación *CLIENTE*. ¿Qué significa un valor nulo para *FechaDeDeceso*? Podría significar que los usuarios no saben si el cliente está vivo o no; o también que el cliente es una corporación y que la *FechaDeDeceso* no es apropiada; o podría significar ambas cosas: que el cliente es una persona y está viva.

Existen varias formas de eliminar estas ambigüedades. La primera es no permitir las. Definir el atributo como se necesita, lo cual está bien siempre y cuando en la mente de los usuarios el atributo verdadero se requiera. Sin embargo, los usuarios se molestarán si se ven obligados a dar un valor de *ColorPreferido* del Cliente, si no es esencial para el funcionamiento de su negocio.

En el capítulo 3 vimos algunas formas de eliminar los valores nulos inapropiados, y definir los subtipos. Definir PACIENTE-MASCULINO y PACIENTE-FEMENINO como subtipos de PACIENTE, por ejemplo, evitaría que los pacientes masculinos tuvieran que proporcionar el número de embarazos, y que a las mujeres se les preguntara sobre la condición de su próstata. Sin embargo, esta solución es costosa, ya que obliga a la definición de dos nuevas tablas y se requiere unirlas para contar con todos los datos de PACIENTE.

Una tercera solución sería definir cada atributo con un valor inicial que sea reconocido en blanco. Un atributo de texto, por ejemplo, podría dar el valor inicial (*desconocido*). Los usuarios podrían subsecuentemente dar el valor (*no apropiado*) para no apropiado, cuando el valor sea inapropiado. Lo anterior sería más eficaz si estas elecciones aparecieran en cajas de texto desplegables (véase el capítulo 10). Mientras que esta solución funciona para atributos de texto, el problema persiste para atributos numéricos, fechas, booleanos y otros atributos que no son de texto. Por supuesto, una solución es modelarlos como datos de texto, así se puede introducir el valor (*desconocido*) y (*no apropiado*). Sin embargo, en ese caso tendrá que escribir su propio código editado para asegurarse de que se introducen números válidos, o datos. También tendrá que proyectar los valores en el código del programa antes de efectuar operaciones numéricas o de datos.

Algunas veces la mejor solución es no hacer nada con respecto a los valores nulos. Si los usuarios pueden enfrentar la ambigüedad, o si la solución es muy costosa, sólo documente el hecho de que existe el problema y trabájelo. También, consulte el capítulo 9 para obtener más información acerca de las consecuencias de todos los valores nulos que se unen en las operaciones.

► RESUMEN

Para transformar los modelos de datos entidad-relación, cada entidad se representa mediante una relación. Los atributos de la entidad se convierten en los de la relación. Una vez que se ha creado la relación, se le debe examinar en función del criterio de normalización y, si es necesario, dividirla en dos o más relaciones.

Existen tres tipos de relaciones binarias TIENE-UN en el modelo E-R: 1:1, 1:N, y N:M. Para representar una relación 1:1, colocamos la llave de una relación en la otra. Las relaciones uno a uno a veces indican que dos relaciones han sido definidas en la misma entidad y se deben combinar en una sola.

Para representar una relación 1:N colocamos la llave del padre en el hijo. Por último, para representar una relación M:N, creamos una relación de intersección que contiene las llaves de las otras dos.

Las relaciones recursivas son aquellas en las que los participantes surgen de la misma clase de entidad. Hay tres tipos: 1:1, 1:N, y N:M, los cuales se representan de la misma manera que las relaciones no recursivas. Para las relaciones 1:1 y 1:N agregamos una llave sustituta que simboliza la entidad. Para una recursión N:M creamos una tabla de intersección que representa la relación M:N.

Las relaciones ternarias y de orden superior se pueden tratar como combinaciones de las relaciones binarias. Sin embargo, si se hace así, cualquiera de las restricciones binarias en las relaciones ternarias también se deben representar en el diseño. Debido a que no es posible imponer la restricción para el diseño relacional, se debe documentar como una regla del negocio. Ocurren tres tipos de estas restricciones: DEBE, NO DEBE y DEBE CUBRIR.

Las entidades de subtipo y supertipo (relaciones ES-UN) también se representan mediante relaciones. Una de ellas se define a través de la entidad supertipo, y otras por cada subtipo. Las llaves de las relaciones normalmente son las mismas y las relaciones entre los renglones se definen a través de estas llaves. Si no son las mismas, la llave de la relación subtipo se puede colocar en la relación supertipo, o la llave de la relación supertipo se coloca en la relación subtipo. Con frecuencia, la llave de la relación supertipo se coloca en la relación subtipo.

Las relaciones binarias se pueden combinar para formar tres clases de estructuras más complejas. Un árbol es un conjunto de tipos de registros en el que cada uno tiene exactamente un padre, con excepción de la raíz. En una red simple, los registros pueden tener padres múltiples, pero éstos deben ser de diferentes tipos. En una red compleja los registros tienen padres múltiples del mismo tipo. Otra forma de expresar lo anterior es que en una red compleja cuando menos una de las relaciones binarias es M:N.

Una lista de materiales es una estructura de datos que con frecuencia se ve en aplicaciones de manufactura. Estas estructuras se pueden representar mediante relaciones recursivas M:N.

Las llaves sustitutas son identificadores únicos que proporciona el sistema, las cuales se usan como las llaves primarias de una relación. Se usan por razones pragmáticas para reducir el tamaño de las llaves y de las llaves externas, así como para mejorar el desempeño. También se usan por razones filosóficas que sirven para mantener la identidad de las entidades. En general, se recomiendan estas llaves.

Un valor nulo es un valor que no ha sido asignado a un atributo. Dichos valores son ambiguos. Pueden significar que el valor es desconocido, que no es apropiado o que se acepta en blanco. Se pueden evitar mediante el requerimiento de los valores de los atributos, por el uso de subtipos, y mediante la distribución de valores sustitutos iniciales. También se pueden ignorar si la ambigüedad presente no es problema para los usuarios.

► PREGUNTAS DEL GRUPO I

- 6.1 Explique cómo se transforman en relaciones las entidades E-R.
- 6.2 ¿Por qué es necesario examinar las relaciones transformadas a partir de entidades, en función del criterio de normalización? ¿Bajo qué condiciones se deben alterar las relaciones si no son DK/NF? ¿En qué casos no se debe hacer?
- 6.3 Explique en qué difiere la representación de las entidades débiles de las entidades fuertes.
- 6.4 Enumere los tres tipos de relaciones binarias y dé un ejemplo sobre cada uno. No use los ejemplos de este texto.
- 6.5 Defina *llave externa* y dé un ejemplo.
- 6.6 Muestre dos formas diferentes para representar la relación 1:1 en su respuesta a la pregunta 6.4. Use diagramas de estructura de datos.
- 6.7 A partir de sus respuestas a la pregunta 6.6, describa un método para obtener datos acerca de las entidades, dando la llave de otra. Describa un método para obtener datos acerca de la segunda entidad, proporcionando la llave de la primera. Describa las respuestas para las alternativas que dio con respecto a la pregunta 6.6.
- 6.8 ¿Por qué algunas relaciones 1:1 se consideran cuestionables? ¿Bajo qué condiciones las relaciones 1:1 deben combinarse en una sola?
- 6.9 Defina los términos *padre* e *hijo* y dé un ejemplo sobre cada uno.
- 6.10 Muestre cómo se representa la relación 1:N en su respuesta a la pregunta 6.4. Use un diagrama de estructura de datos.
- 6.11 En su respuesta a la pregunta 6.10, describa un método con el que se obtengan datos para todos los hijos, considerando la llave del padre. Describa un método que proporcione datos para el padre, considerando la llave del hijo.
- 6.12 Para una relación 1:N explique por qué se debe colocar la llave del padre en el hijo, en lugar de colocar la del hijo en el padre.

- 6.13 Dé ejemplos sobre relaciones binarias 1:N, distintos a los de este texto, para:
- Una relación opcional a opcional
 - Una relación opcional a obligatoria
 - Una relación obligatoria a opcional
 - Una relación obligatoria a obligatoria

Ejemplifique sus respuestas usando diagramas de estructura de datos.

- 6.14 Muestre cómo se representa la relación N:M en su respuesta a la pregunta 6.4. Use un diagrama de estructura de datos.
- 6.15 Para su respuesta a la pregunta 6.14 describa un método con el que se obtengan los hijos para una entidad, considerando la llave de la otra. También describa un método para obtener el hijo de la segunda entidad, a partir de la llave de la primera.
- 6.16 ¿Por qué no es posible representar relaciones N:M con la misma estrategia que se usa para representar las relaciones 1:N?
- 6.17 Explique el significado del término *relación de intersección*.
- 6.18 Defina tres tipos de relaciones binarias recursivas y proporcione un ejemplo de cada una.
- 6.19 Muestre cómo representar la relación recursiva 1:1 en su respuesta a la pregunta 6.18. ¿En qué difiere de la representación de una relación no recursiva 1:1?
- 6.20 Muestre cómo representar la relación recursiva 1:N en su respuesta a la pregunta 6.18. ¿En qué difiere de la representación de las relaciones no recursivas 1:N?
- 6.21 Muestre cómo representar la relación recursiva M:N en su respuesta a la pregunta 6.18. ¿En qué difiere esto de la representación de las relaciones no recursivas M:N?
- 6.22 Explique cómo usar las relaciones binarias para representar una relación ternaria. El ejemplo que proporcione deberá ser diferente a los expuestos.
- 6.23 En su respuesta a la pregunta 6.22, defina una restricción binaria para la relación ternaria y explique cómo representarla. Puesto que la restricción no puede ser obligatoria en el modelo relacional, ¿qué se debe hacer?
- 6.24 Proporcione ejemplos de restricciones binarias NO DEBE y DEBE CUBRIR, diferentes a los de este texto.
- 6.25 Mencione un ejemplo de un supertipo y dos o más subtipos, y muestre cómo se representa usando relaciones.
- 6.26 Defina *árbol*, *red simple* y *red compleja*.
- 6.27 Mencione un ejemplo de una estructura de árbol, diferente a los de este texto, y muestre cómo se representa por medio de relaciones.
- 6.28 Dé un ejemplo sobre red simple, diferente a los de este texto, y muestre cómo se representa por medio de relaciones.
- 6.29 Proporcione un ejemplo de una red compleja, diferente a los de este texto, y muestre cómo se representa por medio de relaciones.
- 6.30 ¿Qué es una lista de materiales? Dé un ejemplo diferente a los de este texto, y muestre cómo representar su ejemplo mediante relaciones.
- 6.31 Defina *llave sustituta* y describa dos razones para usarla.
- 6.32 Describa una situación diferente a la de este texto, sobre la cual existan buenas razones pragmáticas para usar llaves sustitutas.
- 6.33 Explique el enunciado “La llave sustituta sirve para mantener la identidad de la entidad”. Explique por qué es importante.

- 6.34 ¿Cuáles son las tres posibles interpretaciones de los valores nulos?
- 6.35 Describa tres formas diferentes de evitar los valores nulos.
- 6.36 ¿Cuándo no representan un problema los valores nulos?

► PREGUNTAS DEL GRUPO II

- 6.37 Transforme el diagrama E-R para el Club de baile Jefferson (figura 3-19) en relaciones. Exprese su respuesta con un diagrama de estructura de datos y muestre las restricciones de integridad referencial.
- 6.38 Transforme el diagrama E-R para Paseos en barco San Juan (figura 3-21) en relaciones. Exprese su respuesta con un diagrama de estructura de datos y muestre las restricciones de integridad referencial.
- 6.39 Algunas de las relaciones en la figura 6-19 no son DK/NF. Identifíquelas y explique por qué no. ¿Qué forma normal tienen? ¿Cómo se puede justificar este diseño? ¿Cómo podría imponer la base de datos las restricciones binarias?
- 6.40 Establezca todas las restricciones de integridad referencial para las relaciones de la figura 6-20(b).

► PROYECTOS

- A. Termine el proyecto A que está al final del capítulo 3, si aún no lo ha hecho. Transforme su diagrama E-R en un conjunto de relaciones. Si cualquiera de sus relaciones no está en DK/NF, justifique su decisión de crear relaciones sin normalizar.
- B. Termine el proyecto B del final del capítulo 3, si no lo ha concluido. Transforme su diagrama E-R en un conjunto de relaciones. Si sus relaciones no están en DK/NF, justifique su decisión de haberlas creado sin normalizar.
- C. Termine el proyecto C al final del capítulo 3, si todavía no lo ha concluido. Transforme su diagrama E-R en un conjunto de relaciones. Si no están en DK/NF, justifique su decisión de crear relaciones sin normalizar.

► PREGUNTAS DEL PROYECTO FIREDUP

Si aún no lo ha hecho, cree diagramas E-R para las preguntas A y C en el proyecto FiredUp al final del capítulo 3.

- A. Transforme el diagrama entidad-relación de la pregunta A al final del capítulo 3 en un conjunto de relaciones en forma normal dominio/llave. Para cada relación, especifique la llave primaria, las llaves candidatas, si las hay, y las llaves externas. Especifique todas las restricciones de integridad referencial. Si es necesario, establezca y justifique las suposiciones con respecto a las semánticas subrayadas de la aplicación.
- B. Ajuste su respuesta a la pregunta A para permitir relaciones sin normalizar si las considera apropiadas. Justifique cualquiera de las relaciones sin normalizar que tenga. Si es necesario, establezca y justifique las suposiciones registrando las semánticas subrayadas de la aplicación.
- C. Transforme el diagrama entidad-relación de la pregunta C al final del capítulo 3 en un conjunto de relaciones, preferentemente en forma normal dominio-llave. Si cualquiera de sus relaciones no están en dicha forma, explique por qué. Para cada relación

especifique la llave primaria, las llaves candidatas, si las hay, y la llave externa. Detalle todas las restricciones de la interrelación.

D. Ajuste su respuesta a la pregunta anterior para suponer que la casa, el fax y el teléfono celular representan por separado atributos de valor simple. ¿Es un mejor diseño que el de la respuesta a la pregunta C? Explique por qué.

Diseño de bases de datos con modelos de objeto semántico

Este capítulo analiza la transformación de modelos de objeto semántico en diseños de bases de datos relacionales. Primero, describimos la transformación de cada uno de los siete tipos comunes de objetos semánticos. Después, ilustramos esos conceptos mostrando la modelación de objeto semántico y la presentación relacional de varios objetos del mundo real. Puesto que la mejor forma de aprender sobre este tema es trabajar con ejemplos propios, le sugerimos que realice los proyectos que se encuentran al final de este capítulo.

► TRANSFORMACIÓN DE OBJETOS SEMÁNTICOS EN DISEÑOS DE BASES DE DATOS RELACIONALES

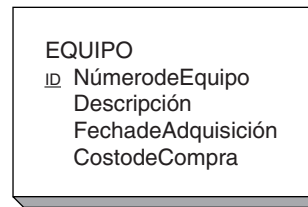
En el capítulo 4 abordamos el modelo de datos de objeto semántico y definimos siete tipos de objetos semánticos. En esta sección presentamos métodos para transformar cada uno de éstos en relaciones. Cuando se trabaja con objetos semánticos, los problemas de normalización son menos comunes que con los modelos E-R, porque la definición de los objetos semánticos normalmente separa los temas semánticos en grupos de atributos o en objetos. De esta manera, cuando se transforma un objeto en relaciones, éstas por lo general están en DK/NF o muy cerca de la forma normal dominio/llave.

OBJETOS SIMPLES

La figura 7-1 ilustra la transformación de un objeto simple en una relación. Recuerde que un objeto simple no tiene atributos multivaluados ni atributos de objeto. Por lo tanto, los objetos simples se pueden representar en la base de datos mediante una sola relación.

► FIGURA 7-1

Representación relacional del ejemplo de objeto simple: (a) diagrama del objeto EQUIPO, y (b) relación que representa a EQUIPO



(a)

EQUIPO (NúmerodeEquipo, Descripción, FechadeAdquisición, CostodeCompra)

(b)

La figura 7-1(a) ejemplifica un objeto simple, EQUIPO, que se puede representar mediante una sola relación, como se muestra en la figura 7-1(b). Cada atributo del objeto se define como un atributo de la relación, y el que lo identifica, NúmerodeEquipo, se convierte en el atributo llave de la relación, que se denota subrayando NúmerodeEquipo en la figura 7-1(b).

La transformación general de objetos simples se ilustra en la figura 7-2. El objeto OBJETO1 es transformado en la relación R1. El atributo que identifica las instancias de OBJETO1 es O1; se convierte en la llave de la relación R1. Los datos que no son llave se representan mediante elipsis (. . .) en ésta y en las siguientes figuras.

Debido a que una llave es un atributo que identifica de forma única un renglón de una tabla, sólo pueden transformarse en llave los identificadores únicos —aquellos con ID subrayada—. Si no hay un identificador único en el objeto, entonces se debe crear uno, mediante la combinación de los atributos existentes para formar un identificador único, o mediante la definición de una llave sustituta.

OBJETOS COMPUESTOS

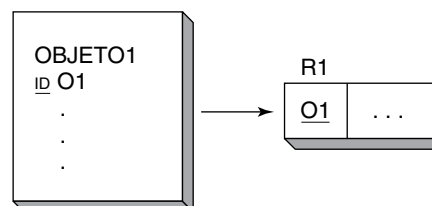
Un objeto compuesto es aquel que tiene uno o más atributos simples multivaluados, o grupo de atributos, pero que no tiene atributos de objeto. La figura 7-3(a) muestra un ejemplo de objeto compuesto: CUENTA-HOTEL. Para representarlo se crea una relación para el objeto base, CUENTA-HOTEL, y otra para la repetición del atributo en grupo, CargoDiario. Este diseño relacional se muestra en la figura 7-3(b).

En la llave de CARGO-DIARIO, NúmerodeFactura está subrayado porque es parte de esa llave, y está en cursivas porque también es una llave externa (es una llave de CARGO-DIARIO). FechadeCargo está subrayada porque es parte de la llave de CARGO-DIARIO, pero no está en cursivas porque no es una llave externa.

En general, los objetos compuestos se transforman mediante la definición de una relación para el objeto mismo y otra relación por cada atributo multivaluado. En la figura 7-4(a), el objeto OBJETO1 contiene dos grupos de atributos multivaluados, cada uno de los cuales está representado por una relación en el diseño de la base de datos. La llave de cada una de estas tablas es el compuesto del identificador del objeto, más el

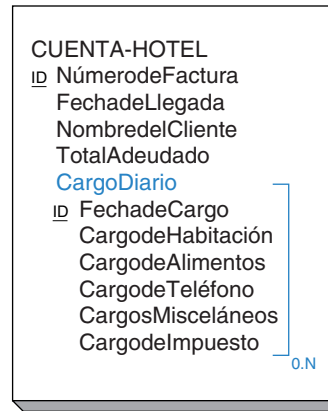
► FIGURA 7-2

Transformación general de objeto simple en una relación



► FIGURA 7-3

Representación relacional de ejemplo de objeto compuesto: (a) diagrama del objeto CUENTA-HOTEL, y (b) representación de relaciones CUENTA-HOTEL



(a)

CUENTA-HOTEL (Número de Factura, Fecha de Llegada, Nombre del Cliente, Total Adeudado)

CARGO-DIARIO (Número de Factura, Fecha de Cargo, Cargo de Habitación, Cargo de Alimentos, Cargo de Teléfono, Cargos Misceláneos, Cargo del Impuesto)

Restricción de integridad referencial:

Número de Factura en CARGO-DIARIO debe existir en
Número de Factura en CUENTA-HOTEL

(b)

identificador del grupo. Así, la representación de OBJETO1 es una relación R1 con la llave O1, una relación R2 con llave (O1, G1), y una relación R3 con llave (O1, G2).

La cardinalidad mínima del objeto con el grupo se especifica mediante la cardinalidad mínima del atributo de grupo. En la figura 7-4(a), la mínima del Grupo1 es 1 y para el Grupo2 es 0. Estas cardinalidades se muestran mediante una marca horizontal (en R2) y una elipse (en R3) en el diagrama de estructura de datos. La cardinalidad mínima del grupo de objetos siempre está predeterminada como 1, porque un grupo no puede existir sin el objeto que lo contiene. Estas cardinalidades mínimas se muestran mediante marcas horizontales en las líneas de relación en R1.

Como se observó en el capítulo 4, los grupos pueden estar anidados. La figura 7-4(b) muestra un objeto en el que el Grupo2 está anidado dentro del Grupo1. Cuando esto ocurre, la relación que representa al grupo anidado se subordina a la relación que representa al grupo que la contiene. En la figura 7-4(b) la relación R3 se subordina a la relación R2. La llave de R3 es la llave de R2, la cual es (O1, G1), más el identificador del Grupo2, que es G2; por lo tanto, la llave de R3 es (O1, G1, G2).

Asegúrese de que ha comprendido por qué las llaves en la figura 7-4(b) están estructuradas de esa manera. También observe que unos atributos están subrayados y en cursivas, y otros sólo subrayados, debido a que algunos atributos son locales y llaves externas, y otros sólo son llaves locales.

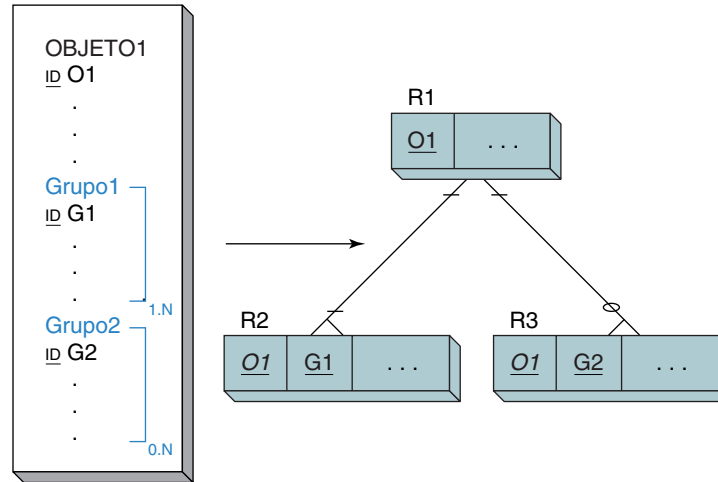
OBJETOS COMBINADOS

La representación relacional de objetos combinados es similar a la de entidades. De hecho, los objetos combinados y las entidades son en muchas formas completamente similares.

Como mencionamos en el capítulo 4, un objeto —OBJETO1— puede contener una o muchas instancias de un segundo objeto —OBJETO2— y OBJETO2 puede contener

► FIGURA 7-4

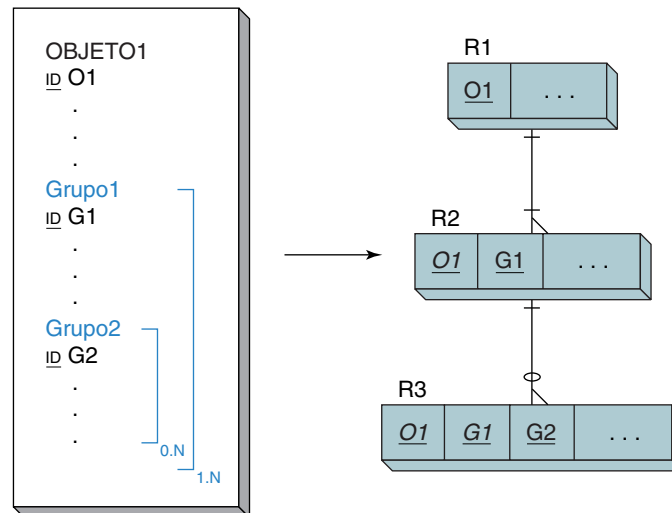
Transformación
general de objetos
compuestos:
(a) objeto compuesto
con grupos
separados, y
(b) objeto compuesto
con grupos anidados



Restricciones de integridad referencial:

O1 en R2 debe existir en O1 en R1
O1 en R3 debe existir en O1 en R1

(a)



Restricciones de integridad referencial:

O1 en R2 debe existir en O1 en R1
(O1, G1) en R3 debe existir en (O1, G1) en R2

(b)

una o muchas instancias del primer objeto, OBJETO1. Esto conduce a los tipos de objeto que se muestran en la figura 7-5.

Todas estas relaciones implican algunas variaciones de relaciones uno a uno, uno a muchos, o muchos a muchos. Específicamente, la relación de OBJETO1 con OBJETO2

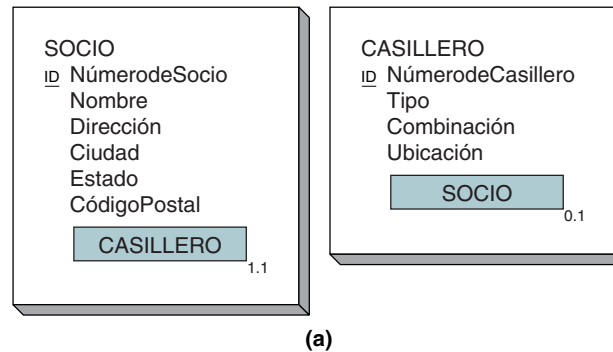
► FIGURA 7-5

Cuatro tipos de
objetos compuestos

Objeto2 Puede Contener	Objeto1 Puede contener	
	Uno	Muchos
Uno	1:1	1:N
Muchos	M:1	M:N

► FIGURA 7-6

Ejemplo de representación relacional de objetos combinados 1:1: (a) ejemplo de objetos combinados 1:1 y (b) su representación



SOCIO (NúmeroDeSocio, Nombre, Dirección, Ciudad, Estado, CódigoPostal, *NúmeroDeCasillero*)

CASILLERO (NúmeroDeCasillero, Tipo, Combinación, Ubicación)

Restricción de integridad referencial:

NúmeroDeCasillero en SOCIO debe existir en
NúmeroDeCasillero en CASILLERO

(b)

puede ser 1:1, 1:N, o N:M, siempre y cuando la relación de OBJETO2 con OBJETO1 sea 1:1, 1:M, o M:N. Para representar cualquiera de éstos, sólo necesitamos direccionar estos tres tipos de relaciones.

REPRESENTACIÓN UNO A UNO DE OBJETOS COMBINADOS

Considere la asignación de un CASILLERO a un SOCIO de un club deportivo. Un CASILLERO se asigna a un SOCIO y cada SOCIO tiene sólo un CASILLERO. La figura 7-6(a) muestra los diagramas de objeto. Para representar estos objetos con relaciones, definimos una relación por cada objeto, y, con relaciones de entidades de 1:1, colocamos la llave de cualquier relación en la otra relación. Así que podemos colocar la llave de SOCIO en CASILLERO o la llave de CASILLERO en SOCIO. La figura 7-6(b) muestra la ubicación de la llave de CASILLERO en SOCIO. Observe que NúmeroDeCasillero está subrayado en CASILLERO porque es la llave, y está en cursivas en SOCIO porque es una llave externa en SOCIO.

En general, para una relación 1:1 entre OBJETO1 y OBJETO2 definimos una relación por cada objeto, R1 y R2. Después, colocamos la llave de cualquiera de las dos relaciones (O1 o O2) como una llave externa en la otra relación, como se ilustra en la figura 7-7.

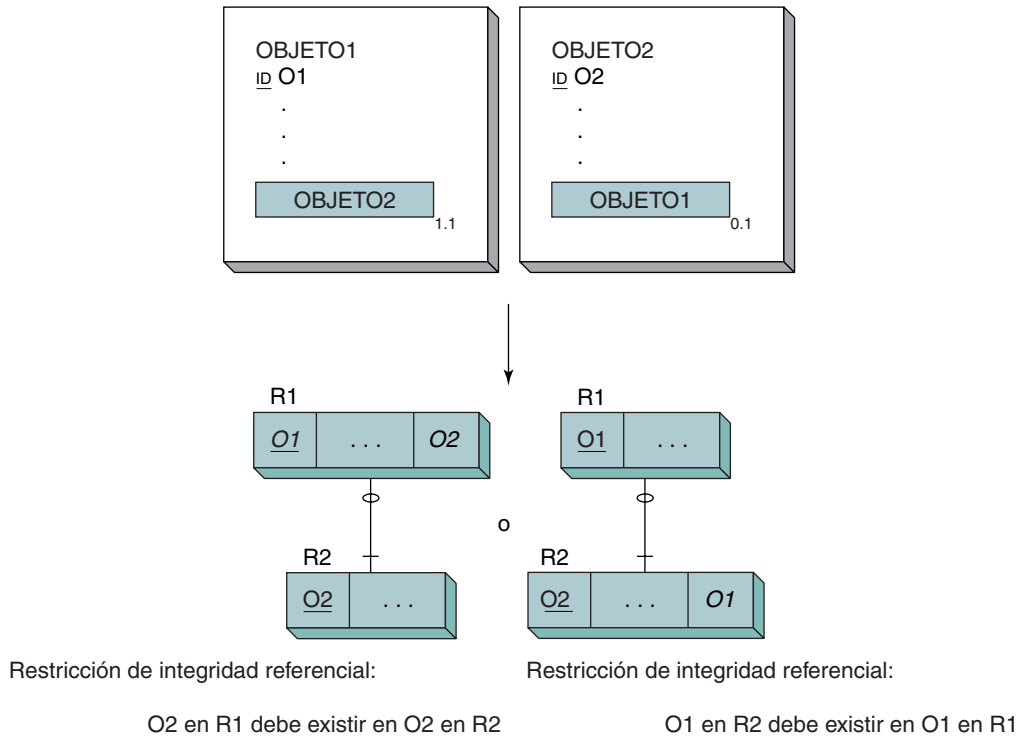
REPRESENTACIÓN DE RELACIONES UNO A MUCHOS Y MUCHOS A UNO

Ahora considere relaciones 1:N y N:1. La figura 7-8(a) muestra un ejemplo de un objeto de relación 1:N entre EQUIPO y REPARACIÓN. Un artículo de EQUIPO puede tener muchas REPARACIONes pero una REPARACIÓN se puede relacionar sólo con un artículo de EQUIPO.

Los objetos en la figura 7-8(a) se representan mediante las relaciones en la figura 7-8(b). Observe que la llave del padre (el objeto a un lado de la relación) se coloca en el hijo (el objeto en los diferentes lados de la relación).

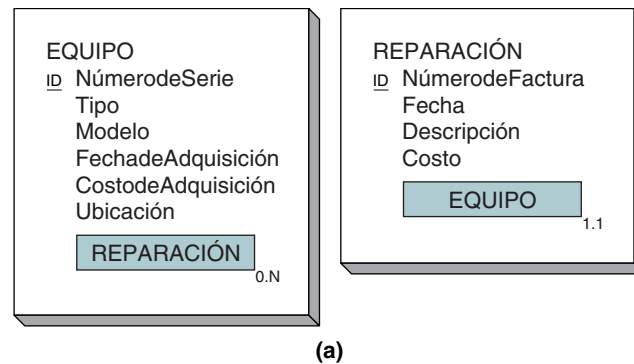
► FIGURA 7-7

Transformación general de objetos combinados 1:1



► FIGURA 7-8

Ejemplo de representación relacional de objetos combinados de 1:N: (a) ejemplo de objetos combinados 1:N, y (b) su representación



EQUIPO (NúmeroSerie, Tipo, Modelo, Fecha de Adquisición, Costo de Adquisición, Ubicación)

REPARACIÓN (NúmeroFactura, Fecha, Descripción, Costo, NúmeroSerie)

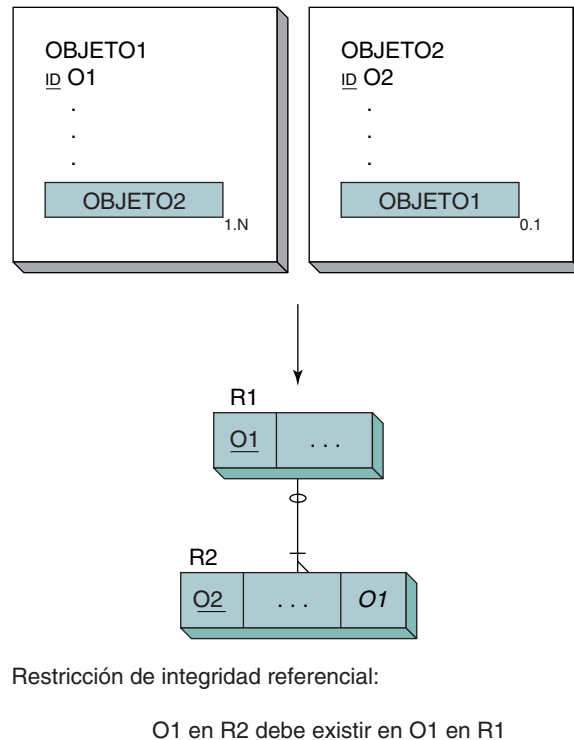
Restricción de integridad referencial:

NúmeroSerie en REPARACIÓN debe existir en
NúmeroSerie en EQUIPO

(b)

► FIGURA 7-9

*Transformación
general de objetos
combinados 1:N*



La figura 7-9 muestra la transformación general de objetos combinados 1:N. El objeto OBJETO1 contiene muchos objetos OBJETO2, y el objeto OBJETO2 contiene sólo un OBJETO1. Para representar esta estructura por medio de relaciones, se representa cada objeto con una relación y se coloca la llave del padre en el hijo. De esta forma, en la figura 7-9 el atributo O1 se coloca en R2.

Si OBJETO2 contuviera muchos OBJETOs1 y OBJETO1 sólo un OBJETO2, usaríamos la misma estrategia, pero se invertirían las funciones de R1 y R2. Así que sería mejor colocar O2 en R1.

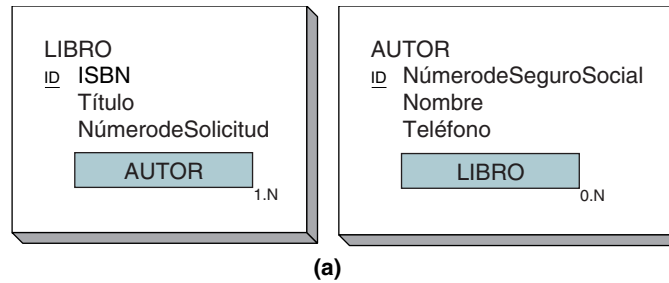
Las cardinalidades mínimas en cualquier caso están determinadas por las cardinalidades mínimas de los atributos del objeto. En la figura 7-9 el OBJETO1 requiere cuando menos un OBJETO2, pero OBJETO2 no necesariamente requiere un OBJETO1. Estas cardinalidades se muestran en el diagrama de estructura de datos con un óvalo al lado de la relación R1, y con una marca horizontal al lado de la relación R2. Estos valores de cardinalidades mínimas son simplemente ejemplos; cualquier objeto, o ambos, podría tener una cardinalidad de 0, 1, o algún otro número.

REPRESENTACIÓN DE RELACIONES MUCHOS A MUCHOS

Finalmente, considere las relaciones M:N. Igual que con las relaciones de entidad M:N, definimos tres relaciones: una para cada objeto, y una tercera relación de intersección. Esta última representa la relación de los dos objetos y consta de las llaves de ambos padres. La figura 7-10(a) muestra la relación M:N entre LIBRO y AUTOR. La figura 7-10(b) describe las tres relaciones que representan estos objetos: la relación de intersección LIBRO, AUTOR y LIBRO-AUTOR-INT. Observe que LIBRO-AUTOR-INT no tiene datos llave. Ambos atributos, ISBN y NúmerodeSeguroSocial, están subrayados y en cursivas porque son llaves locales y externas.

► FIGURA 7-10

Representación relacional del ejemplo de objeto combinado N:M: (a) objetos LIBRO y AUTOR, y (b) su representación relacional



LIBRO (ISBN, Título, NúmerodeSolicitud)

AUTOR (NúmerodeSeguroSocial, Nombre, Teléfono)

LIBRO-AUTOR-INT (ISBN, NúmerodeSeguroSocial)

Restricciones de integridad referencial:

ISBN en LIBRO-AUTOR-INT debe existir en
ISBN en LIBRO

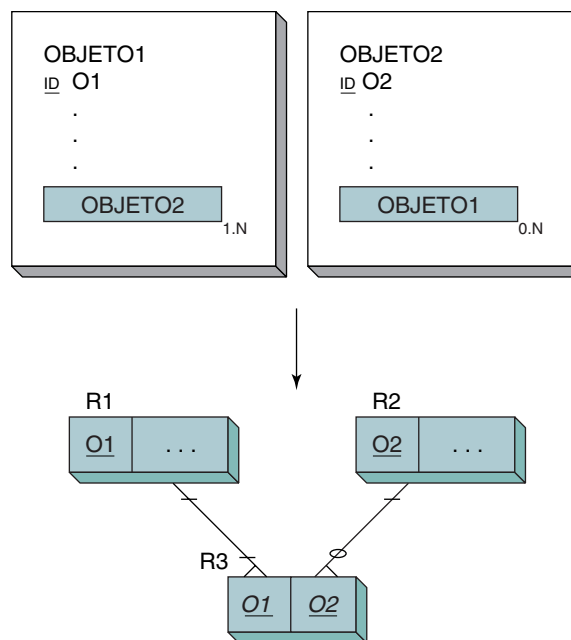
NúmerodeSeguroSocial en LIBRO-AUTOR-INT debe existir en
NúmerodeSeguroSocial en AUTOR

(b)

En general, para los dos objetos que tienen una relación M:N, definimos una relación R1 para OBJETO1, una R2 para OBJETO2 y una R3 para la relación de intersección. El esquema general se muestra en la figura 7-11. Observe que los atributos de R3 son sólo O1 y O2. Para los objetos compuestos M:N, R3 nunca tendrá datos sin llave. La importancia de este enunciado será obvia cuando comparemos relaciones combinadas M:N con relaciones de asociación.

► FIGURA 7-11

Transformación general de objetos combinados M:N en relaciones



Restricciones de integridad referencial:

O1 en R3 debe existir en O1 en R1
O2 en R3 debe existir en O2 en R2

Cuando se considera la cardinalidad mínima siempre se requieren los padres de la relación de intersección. Las cardinalidades mínimas de las relaciones en la relación de intersección se determinan mediante las cardinalidades mínimas de los vínculos de los objetos. En la figura 7-11, por ejemplo, un renglón en R1 requiere un renglón en R3 porque la cardinalidad mínima de OBJETO2 en OBJETO1 es 1. De manera similar, un renglón en R2 no requiere un renglón en R3 porque la cardinalidad mínima de OBJETO1 en OBJETO2 es 0.

OBJETOS HÍBRIDOS

Los objetos híbridos pueden transformarse en diseños relacionales usando una combinación de las técnicas para los objetos compuestos y combinados. La figura 7-12(a) muestra PEDIDO-VENTAS, un objeto híbrido y objetos relacionados. Para representar este objeto por medio de relaciones establecemos una para el mismo objeto y otra para cada uno de los objetos contenidos en CLIENTE y VENDEDOR. Entonces, al igual que con un objeto compuesto, establecemos una relación para el grupo de valor múltiple, que es ArticulodeLínea. Puesto que este grupo contiene otro objeto, ARTÍCULO, también establecemos una relación para ARTÍCULO. Todas las relaciones uno a muchos se representan mediante la colocación de la llave de la relación padre en la relación hijo, como se muestra en la figura 7-12(b).

El ejemplo en la figura 7-12 es engañosamente simple. Como mencionamos en el capítulo 4, en realidad hay cuatro instancias de objetos híbridos, los cuales se resumen en la figura 7-13.

Los casos 3 y 4 son más comunes que los 1 y 2, así que los consideramos primero. En la figura 7-14 OBJETO1 muestra dos grupos: el Grupo1 ilustra el caso 3 y el Grupo2, el caso 4.

Grupo1 tiene una cardinalidad máxima de N, lo que significa que puede haber muchas instancias del Grupo1 en el OBJETO1. Además, puesto que OBJETO2 está marcado como ID único, quiere decir que un OBJETO2 en particular puede aparecer sólo en una de las instancias del GRUPO1 dentro de un OBJETO1. Así, OBJETO2 actúa como un identificador para GRUPO1 dentro de OBJETO1.

(Cabe aclarar que PEDIDO-VENTAS en la figura 7-12 ilustra este caso. ARTÍCULO es un identificador de ArticulodeLínea, así que un ARTÍCULO determinado puede aparecer en ArticulodeLínea sólo en un PEDIDO; pero un ARTÍCULO puede aparecer en muchos PEDIDOS.)

Considere la representación relacional de Grupo1 en la figura 7-14. Se crea una relación, R1, para OBJETO1, y una relación, R2, para OBJETO2. Además, se crea una tercera relación, R-G1, para Grupo1. La relación entre R1 y R-G1 es 1:N, así que colocamos la llave de R1 (que es O1) en R-G1; la relación entre R2 y R-G1 también es 1:N, así que colocamos la llave de R2 (la cual es O2) en R-G1. Puesto que un OBJETO2 sólo puede aparecer una vez con un valor particular de OBJETO1, el compuesto (O1, O2) es único para R-G1 y puede convertirse en la llave de esa relación.

Ahora consideremos el Grupo2. OBJETO3 no identifica a Grupo2, por lo que OBJETO3 puede aparecer en la mayoría de las instancias de Grupo2 en el mismo OBJETO1. (En la figura 7-12 PEDIDO-VENTAS estaría en igual circunstancia si ARTÍCULO no fuera ID único en ArticulodeLínea. Esto significa que un ARTÍCULO podría aparecer muchas veces en el mismo PEDIDO.) Debido a que OBJETO3 no es el identificador del Grupo2, suponemos que algún otro atributo, G2, es el identificador.

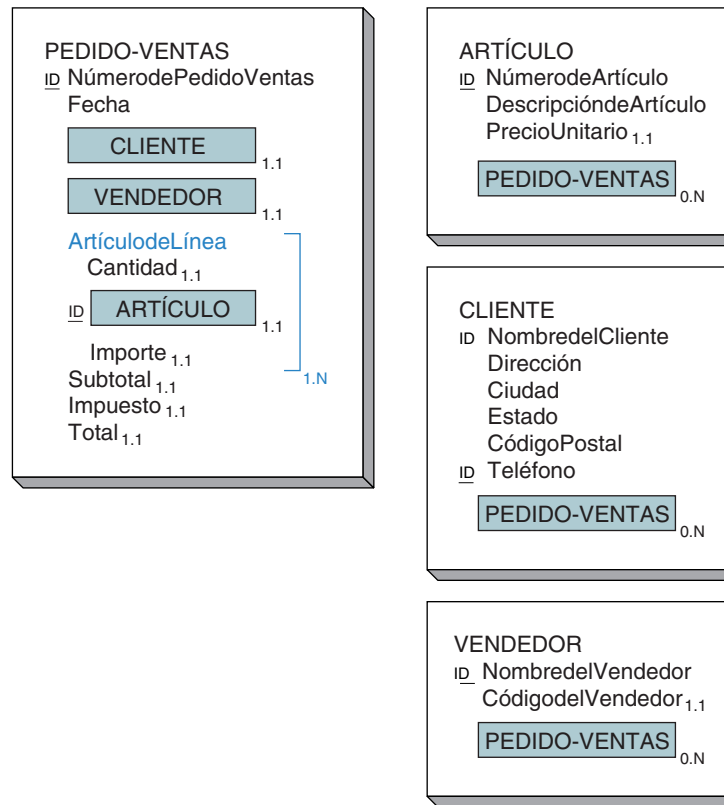
En la figura 7-14 creamos una relación R3 para OBJETO3 y otra, R-G2, para Grupo2. La relación entre R1 y R-G2 es 1:N, así que se coloca la llave de R1 (que es O1) en R-G2. La relación entre R3 y R-G2 es también 1:N, por lo cual se coloca la llave de R3 (que es O3) en R-G2.

Sin embargo, a diferencia de Grupo1, ahora (O1, O3) no puede ser la llave de R-G2 porque un O3 puede estar pareado muchas veces con un O1 determinado. Así, el compuesto (O1, O3) no es único para R-G2, por lo que la llave de R-G2 debe ser (O1-G2).

El caso 1 es similar al 3, excepto en la restricción de que un OBJETO2 se puede relacionar únicamente con un OBJETO1. Las relaciones en la figura 7-14 todavía fun-

► FIGURA 7-12

Representación relacional del ejemplo de objeto híbrido: (a) ejemplo de objeto híbrido, y (b) representación relacional de PEDIDO-VENTAS y objetos relacionados



(a)

PEDIDO-VENTAS (Número de Pedido Ventas, Fecha, Subtotal, Impuesto, Total, Teléfono, Nombre del Vendedor)

CLIENTE (Nombre del Cliente, Dirección, Ciudad, Estado, Código Postal, Teléfono)

VENDEDOR (Nombre del Vendedor, Código del Vendedor)

ARTÍCULO-LÍNEA (Número de Pedido de Ventas, Número de Artículo, Cantidad, Importe)

ARTÍCULO (Número de Artículo, Descripción de Artículo, Precio Unitario).

Restricciones de integridad referencial:

Nombre del Vendedor en PEDIDO-VENTAS debe existir en
Nombre del Vendedor en VENDEDOR

Teléfono en PEDIDO-VENTAS debe existir en
Teléfono en CLIENTE

Número de Pedido Ventas en ARTÍCULO-LÍNEA debe existir en
Número de Pedido Ventas en PEDIDO-VENTAS

Número de Artículo en ARTÍCULO-LÍNEA debe existir en
Número de Artículo en ARTÍCULO

(b)

cionarán, pero debemos agregar la llave de R1 (que es O1) a R2 y establecer la restricción de que (O1, O2) de R-G1 debe ser igual a (O1, O2) de R2.

El caso 2 es similar al caso 4, excepto por la restricción de que un OBJETO 3 puede estar relacionado sólo con un OBJETO1. Nuevamente, las relaciones en la figura 7-14 fun-

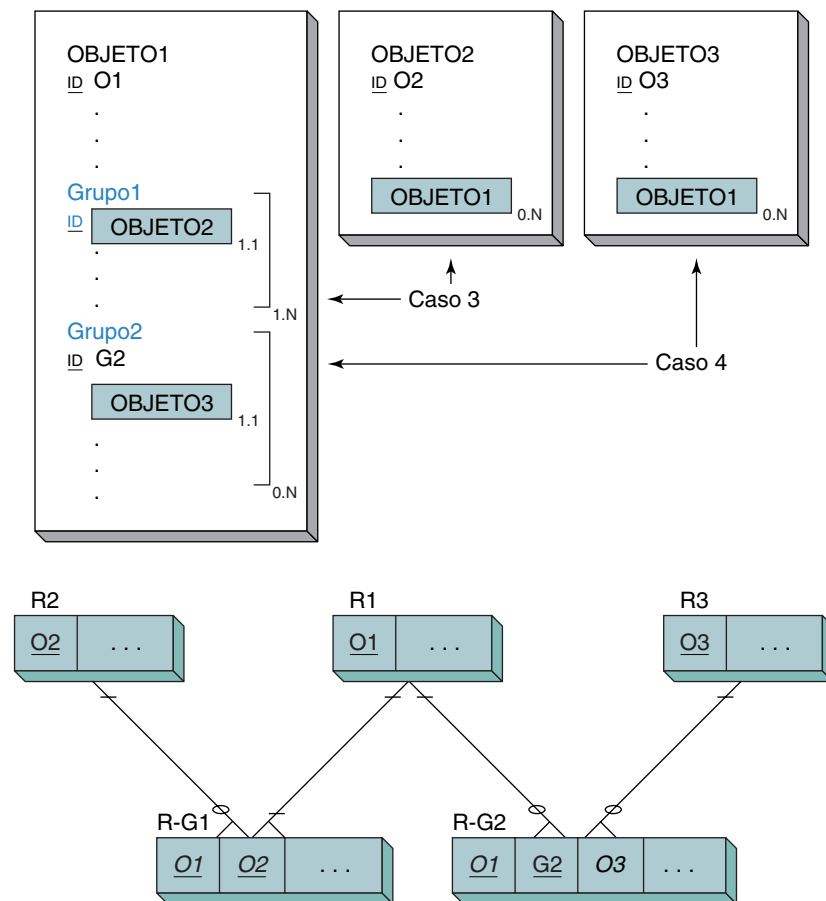
► FIGURA 7-13

Cuatro casos de cardinalidad de objetos híbridos

Caso	Descripción	Ejemplo
1	OBJETO2 se relaciona con una instancia de OBJETO1 y sólo aparece en una instancia de grupo dentro de ese objeto.	ARTÍCULO se relaciona con un PEDIDO y sólo puede aparecer en un ArtículosdeLínea de ese PEDIDO.
2	OBJETO2 se relaciona con una instancia de OBJETO1 y es posible que aparezca en muchas instancias de grupo dentro de ese objeto.	ARTÍCULO se relaciona con un PEDIDO y puede aparecer en muchos ArtículosdeLínea de ese PEDIDO.
3	Es posible que OBJETO2 se relacione con muchas instancias de OBJETO1 y sólo aparezca en una instancia de grupo dentro de cada objeto.	ARTÍCULO se relaciona con muchos PEDIDOS y sólo puede aparecer en un ArtículosdeLínea de ese PEDIDO.
4	Es posible que OBJETO2 se relacione con muchas instancias de OBJETO1 y quizá aparezca en muchas instancias dentro de esos objetos.	ARTÍCULO se relaciona con muchos PEDIDOS y puede aparecer en muchos ArtículosdeLínea de ese PEDIDO.

► FIGURA 7-14

Transformación general de objetos híbridos en relaciones



Restricciones de integridad referencial:

- O1 en R-G1 debe existir en O1 en R1
- O2 en R-G1 debe existir en O2 en R2
- O1 en R-G2 debe existir en O1 en R1
- O3 en R-G2 debe existir en O3 en R3

cionarán, pero debemos agregar la llave de R1 (que es O1) a R3, y establecer la restricción (O1, O3) de que R-G2 es un subconjunto de (O1, O3) en R3 (véanse las preguntas 7.7 y 7.8).

OBJETOS DE ASOCIACIÓN

Un objeto de asociación es aquel que une dos objetos. Es un caso especial de objetos combinados que ocurre con mayor frecuencia en las situaciones de asignación. La figura 7-15(a) muestra un objeto VUELO que liga a un AVIÓN con un PILOTO.

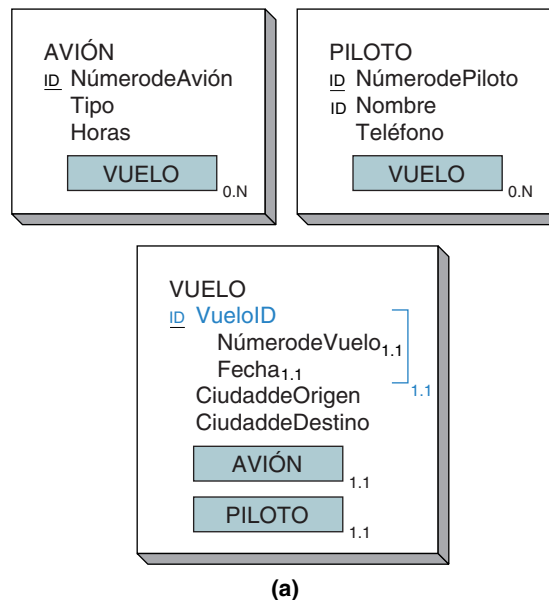
Para representar los objetos de asociación definimos una relación para cada uno de los tres objetos, y después representamos las relaciones entre los objetos empleando una de las estrategias que se usaron con los objetos combinados. Por ejemplo, en la figura 7-15(b) se define una relación para AVIÓN, una para PILOTO y otra para VUELO. Las relaciones entre VUELO y AVIÓN, y entre VUELO y PILOTO son 1:N, así que ponemos las llaves del padre en los hijos. En este caso, colocamos la llave de AVIÓN y la de PILOTO en VUELO.

VUELO contiene una llave propia. Aunque tiene llaves externas, sólo son atributos y no parte de la llave de VUELO. Pero éste no siempre es el caso. Si VUELO no tuviera una llave propia, ésta sería la combinación de las llaves externas de los objetos asociados. En ese caso la combinación sería {NúmerodeAvión, NúmerodePiloto, Fecha}.

En general, cuando se transforman en relaciones las estructuras del objeto de asociación, definimos una relación por cada uno de los objetos que participan en ésta. En

► FIGURA 7-15

Representación relacional de ejemplo de objeto de asociación: (a) objeto de asociación VUELO y objetos relacionados, y (b) representación relacional de los objetos AVIÓN, PILOTO y VUELO



(a)

AVIÓN (NúmerodeAvión, Tipo, Horas)

PILOTO (NúmerodePiloto, Nombre, Teléfono)

VUELO (NúmerodeVuelo, Fecha, CiudaddeOrigen, CiudaddeDestino, NúmerodeAvión, NúmerodePiloto)

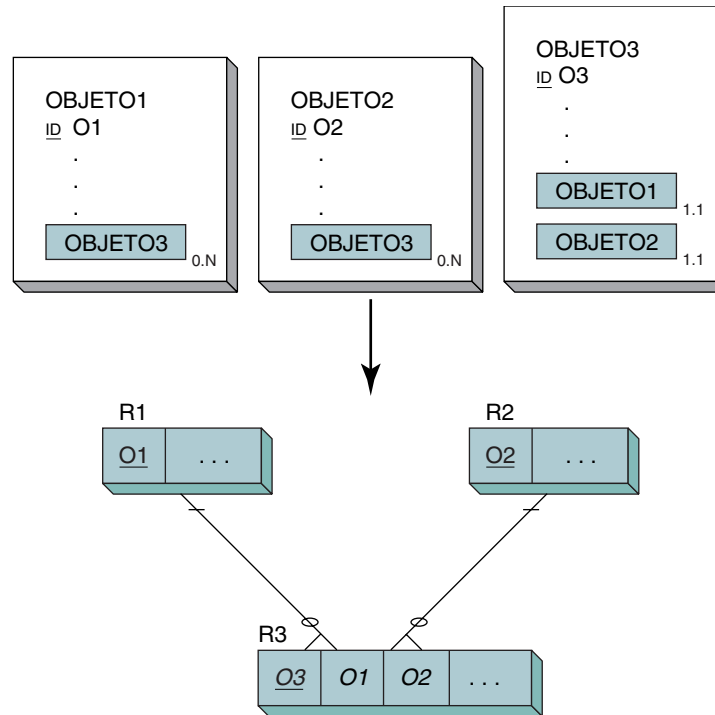
Restricción de integridad referencial:

NúmerodeAvión en VUELO debe existir en NúmerodeAvión en AVIÓN
NúmerodePiloto en VUELO debe existir en NúmerodePiloto en PILOTO

(b)

► FIGURA 7-16

Transformación general de los objetos de asociación dentro de las relaciones



Restricción de integridad referencial:

O1 en R3 debe existir en O1 en R1
O2 en R3 debe existir en O2 en R2

la figura 7-16, OBJETO3 asocia a OBJETO1 y OBJETO2. En este caso, definiremos R1, R2 y R3 como se muestra. Las llaves de cada una de las relaciones padre, O1 y O2, aparecen como atributos de la llave externa en R3, la relación que representa el objeto de asociación. Si el objeto de asociación no tiene un atributo único que lo identifique, la combinación de los atributos de R1 y R2 será usado para crear un identificador único.

Observe la diferencia entre la relación de asociación de la figura 7-16 y la de intersección en la figura 7-11. La diferencia principal es que la tabla de asociación lleva datos que representan algún aspecto de la combinación de los objetos. La relación de intersección no incluye datos, sólo existe para especificar qué objetos tienen una relación entre sí.

OBJETOS PADRE Y OBJETOS SUBTIPO

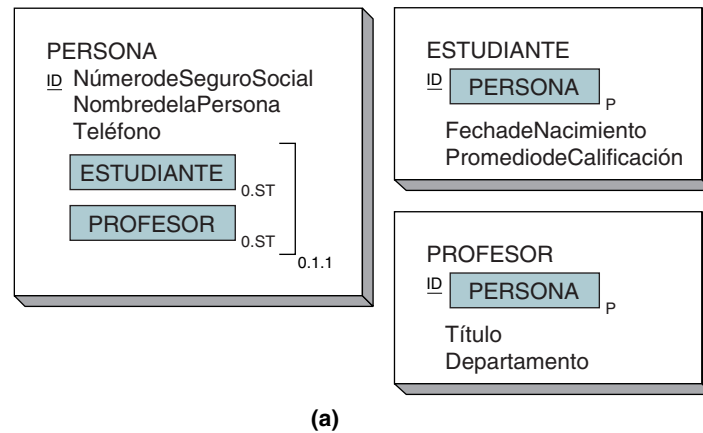
Los objetos padre (también llamados *supertipo*) y los objetos subtipo se representan de manera similar a las entidades padre y subtipo. Definimos una relación por el objeto padre y uno por cada objeto de subtipo. La llave de cada una de estas relaciones es la llave del padre.

La figura 7-17(a) muestra un objeto padre, PERSONA, que incluye dos subtipos mutuamente excluyentes, ESTUDIANTE y PROFESOR. La figura 7-17(b) muestra una representación relacional de estos tres objetos. Cada uno está representado mediante una tabla, y la llave de todas las tablas es la misma.

Sin embargo, las relaciones en la figura 7-17(b) tienen un problema: el programa de aplicación todavía necesita buscar en ambas tablas, ESTUDIANTE y PROFESOR, para determinar el tipo de PERSONA. Si se encuentra una entrada en ESTUDIANTE la persona es un estudiante; pero si se encuentra en PROFESOR, es un profesor. Éste es un

► FIGURA 7-17

Representación del ejemplo padre y subtipos: (a) padre PERSONA y subtipos ESTUDIANTE y PROFESOR, (b) representación relacional de padre y subtipos, y (c) representaciones alternativas de la relación padre



PERSONA (NúmerodeSeguroSocial, NombredePersona, Teléfono)

ESTUDIANTE (NúmerodeSeguroSocial, FechadeNacimiento, PromediodeCalificación)

PROFESOR (NúmerodeSeguroSocial, Título, Departamento)

Restricciones de integridad referencial:

NúmerodeSeguroSocial en ESTUDIANTE debe existir en
NúmerodeSeguroSocial en PERSONA

NúmerodeSeguroSocial en PROFESOR debe existir en
NúmerodeSeguroSocial en PERSONA

PERSONA1 (NúmerodeSeguroSocial, NombredePersona, Teléfono, TipodePersona)

PERSONA2 (NúmerodeSeguroSocial, NombredePersona, Teléfono, TipodeEstudiante,
TipodeProfesor)

modo indirecto y posiblemente lento para determinar el tipo de persona y si, como puede suceder, PERSONA no es de ninguno de los tipos, ambas tablas habrán sido consultadas sin razón alguna. Debido a este problema, a veces se coloca un tipo de atributo indicador en la tabla padre.

La figura 7-17(c) muestra dos variaciones de un indicador de tipo. En la primera variación, la relación PERSONA1, el tipo de objeto se almacena en el atributo TipodePersona. Los valores posibles de este atributo son "Ninguno", "ESTUDIANTE", o "PROFESOR". La aplicación se obtendría del valor de este atributo y, por lo tanto, determinaría si un subtipo existe o no, y si existe, de qué tipo es.

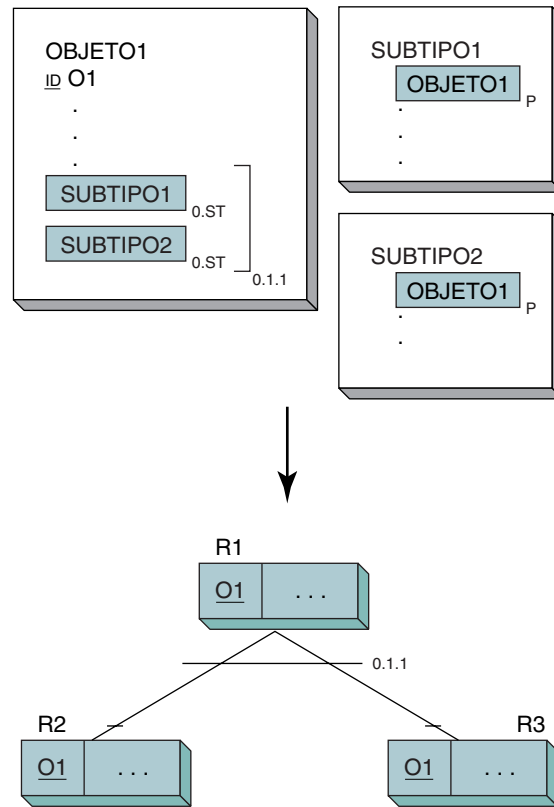
Una segunda posibilidad se aprecia en la relación PERSONA2, a la cual se han agregado dos atributos: uno para TipodeEstudiante y otro para TipodePersona. Cada uno de los atributos es una variable booleana; los valores permitidos son verdaderos o falsos. Observe que, como sucede en este caso, si una persona sólo puede ser de un tipo, y si entonces uno de estos valores es verdadero, el otro debe ser falso.

En general, los diseños del tipo PERSONA1 son mejores cuando los subtipos son mutuamente excluyentes. Los diseños del tipo PERSONA2 son mejores cuando los subtipos no son excluyentes.

Un esquema general de la representación de los subtipos se muestra en la figura 7-18. Se crea una relación para padre y otra para cada uno de los subtipos. La llave de todas las relaciones es el identificador del padre. Todas las relaciones entre el padre y el subtipo son 1:1. Observe la barra horizontal en las líneas de la relación y la presencia

► FIGURA 7-18

*Transformación
general de los objetos
padre-subtipo en
relaciones*



Restricción de integridad referencial:

O1 en R2 debe existir en O1 en R1
O1 en R3 debe existir en O1 en R1

de la cardinalidad del grupo de subtipo. El valor mostrado, 0.1.1, significa que no se requirió subtipo, pero si se presenta, cuando mucho se permite uno de los subtipos.

(Recuerde que en general el formato de la cardinalidad del grupo es **r.m.n.**, donde **r** es un valor booleano verdadero o falso, dependiendo de si se requiere o no el grupo subtipo; **m** es el número mínimo de subtipos que debe tener un valor dentro del grupo, y **n** es el número máximo de subtipos que puede tener un valor dentro del grupo. Por lo tanto, en un grupo de cinco subtipos, la cardinalidad de 1.2.4 indica que se requiere el grupo de subtipos, que cuando menos dos subtipos deben tener un valor y que un máximo de cuatro subtipos pueden tener un valor.)

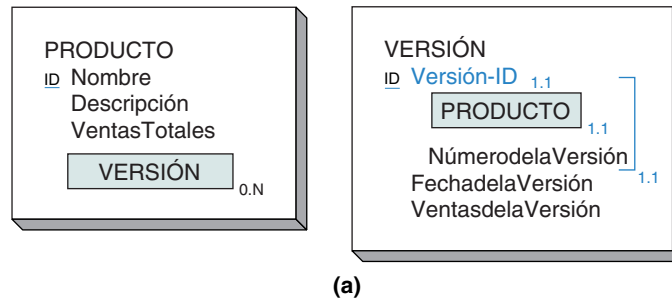
OBJETOS ARQUETIPO-VERSIÓN

Los objetos arquetipo-versión son objetos combinados que modelan varias repeticiones, versiones, o instancias de un objeto básico. Los objetos en la figura 7-19(a) modelan productos de software para los cuales hay varias versiones. Ejemplos de estos productos son Microsoft Internet Explorer o Netscape Navigator. Ejemplos de versiones son Access 2000 y Access 2002.

La representación relacional de PRODUCTO y VERSIÓN se muestra en la figura 7-19(b). Se crea una relación para PRODUCTO, y otra para VERSIÓN. La llave de VERSIÓN es la combinación de la llave de PRODUCTO y la llave local (Número de la Versión) de VERSIÓN.

► FIGURA 7-19

Representación relacional del ejemplo de objetos arquetipo-versión: (a) objeto arquetipo PRODUCTO y objeto versión VERSIÓN, y (b) representación relacional de PRODUCTO y VERSIÓN



PRODUCTO (Nombre, Descripción, VentasTotales)

VERSIÓN (Nombre, Número de la Versión, Fecha de la Versión, Ventas de la Versión)

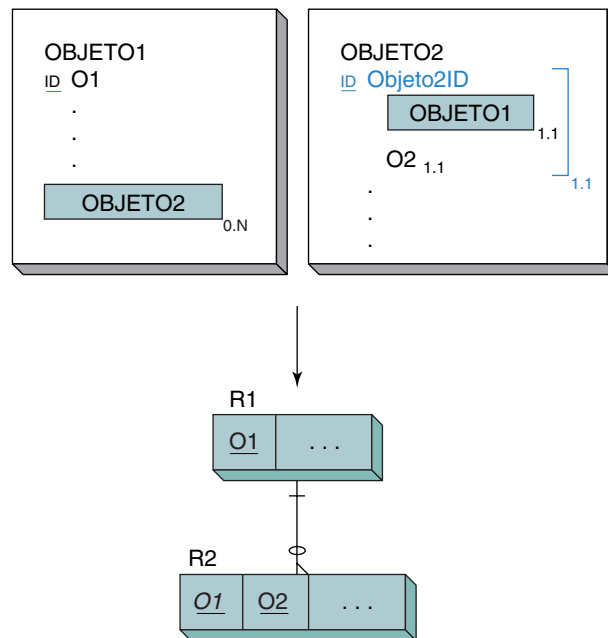
Restricción de integridad referencial:

Nombre en VERSIÓN debe existir en Nombre en PRODUCTO

(b)

► FIGURA 7-20

Transformación general de los objetos de arquetipo-versión y versión VERSIÓN



Restricción de integridad referencial:

O1 en R2 debe existir en O1 en R1

En la figura 7-20 se muestra la transformación general de los objetos arquetipo-versión. El atributo O1 de R2 es tanto una llave local como una externa, pero O2 sólo es una llave local.

► OBJETOS DE MUESTRA

Para reforzar los conceptos que hemos presentado, ahora consideraremos varios ejemplos de objetos tomados de negocios reales, y los presentaremos por orden de complejidad. Modelamos el objeto implícito y lo representamos en relaciones usando los métodos descritos en este capítulo. La especificación de restricciones de integridad relacional se deja para las preguntas 7.20, 7.21 y 7.22.

FORMA DE SUSCRIPCIÓN

La figura 7-21(a) muestra un formato de suscripción a una revista. Cuando menos dos estructuras de objeto podrían representarla. Si los editores de *Fine Woodworking* consideran a un suscriptor como un atributo de una suscripción, ésta podría ser un objeto simple representado como una relación simple, como se muestra en la figura 7-21(b).

Si esta empresa sólo cuenta con una publicación y no tiene planes para editar otras, funcionará el diseño de la figura 7-21(b); pero si tiene varias publicaciones y un cliente puede suscribirse a más de una, este diseño duplicará los datos del cliente para cada publicación. Esto no sólo sería un desperdicio de espacio en el archivo del editor, sino que lo exasperaría porque, por ejemplo, tendría que informar todos los cambios de dirección para cada publicación.

Si la editorial tiene varias publicaciones, o planea tenerlas, un diseño más apropiado sería modelar al suscriptor como un objeto por separado, como se muestra en la figura 7-21(c). CLIENTE es un objeto combinado 1:N y se representa mediante las relaciones en esta figura.

DESCRIPCIÓN DEL PRODUCTO

La figura 7-22(a) muestra la descripción de un producto popular de artículos empacados. Mientras que la figura 7-21(a) ilustra una forma genérica sin datos, la 7-22(a) ofrece un ejemplo de un reporte específico con datos acerca de un producto de cereales. Los reportes para todos los cereales de Kellogg's usan este formato.

La figura 7-22(b) muestra un objeto compuesto que podría estar implícito en este reporte. Decimos *podría* porque hay muchas formas diferentes en las que este objeto puede ser representado. También, una investigación adicional revelaría otros objetos que no son evidentes en este reporte. Por ejemplo, la ración diaria de consumo que recomienda el Ministerio de Agricultura de Estados Unidos puede ser un objeto semántico por derecho propio.

Con el fin de ilustrar lo anterior, hacemos diferentes suposiciones acerca de los grupos Nutriente y RaciónDiariaRecomendada. El objeto PRODUCTO-CEREAL supone que se requiere cada elemento del grupo de nutriente —calorías, proteínas, carbohidratos, grasa, colesterol, sodio, y potasio— en cada instancia de este objeto. No hacemos esa suposición para el grupo RaciónDiariaRecomendada porque sólo debe existir una instancia de este grupo.

El reporte de la figura 7-22(a) tiene muchas interpretaciones y se podría modelar de diferentes maneras. En un proyecto de desarrollo real sería importante obtener tantos reportes como sea posible acerca de los cereales, los ingredientes y la información nutricional. Posiblemente los otros documentos darían la estructura adicional para este objeto semántico.

La figura 7-22(c) muestra la representación relacional para el objeto PRODUCTO-CEREAL. La cardinalidad mínima de 7 se muestra colocando el número 7 junto a la marca horizontal en la línea de relación. Las llaves externas se colocaron como se describió anteriormente en el caso de los objetos compuestos.

CITATORIO DE AMONESTACIÓN VIAL

La figura 7-23(a) muestra un ejemplo de una forma de citatorio de amonestación vial que se usa en el estado de Washington. El diseñador de esta forma nos ha dado pistas importantes de los objetos implícitos. Observe que partes de la forma se distinguen porque tienen las esquinas redondeadas, lo cual indica las diferentes secciones que pertenecen a objetos diferentes. También, algunos grupos de atributos tienen nombres, lo que representa la necesidad de grupos de atributos.

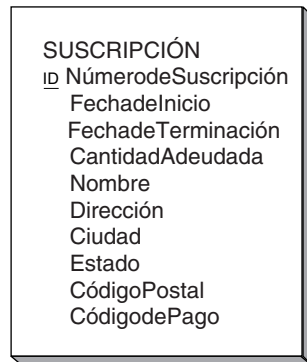
La figura 7-23(b) es una manera de ilustrar los objetos implícitos del citatorio de amonestación vial. Aunque no podemos estar seguros sólo a partir de una forma, hay ciertos indicios que nos llevan a creer que el conductor, el vehículo y el oficial son ob-

► FIGURA 7-21

Representación alternativa de suscripción:
 (a) forma de pedido de suscripción,
 (b) suscripción modelada como un objeto, y
 (c) suscripción modelada como dos objetos

Fine Wood Working		Suscripción
☐ 1 año (6 ejemplares) por sólo \$18; 20% menos que en el puesto de periódicos (\$21 por año en el extranjero) ☐ 2 años (12 ejemplares) por sólo \$34; ahorre el 24% (\$40 por dos años en el extranjero)		
Nombre _____		
Dirección _____		
Ciudad _____ Estado _____ Código postal _____		
☐ Mi pago está incluido. ☐ Pagaré al recibirla. Iniciar mi suscripción con ☐ número actual ☐ número siguiente		

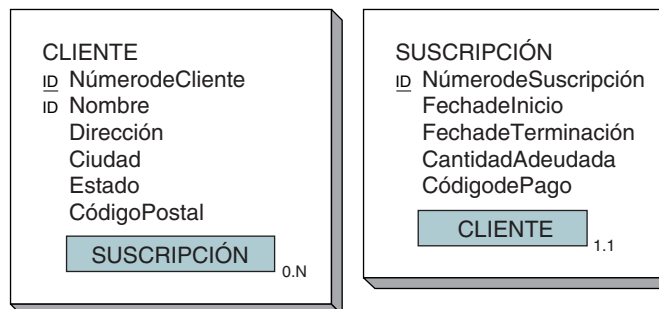
(a)



SUSCRIPCIÓN

NúmeroSuscripción	FechaInicio	FechaTerminación	CantidadAdeudada	Nombre
Dirección	Ciudad	Estado	CódigoPostal	CódigoPago

(b)



CLIENTE

NúmeroCliente	Nombre	Dirección	Ciudad	Estado	CódigoPostal
---------------	--------	-----------	--------	--------	--------------

SUSCRIPCIÓN

NúmeroSuscripción	FechaInicio	FechaTerminación	CantidadAdeudada
CódigoPago	NúmeroCliente		

(c)

FIGURA 7-22

Representación de producto cereal:
(a) reporte de producto cereal,
(b) diagrama de objetos de PRODUCTO-CEREAL, y
(c) representación relacional de PRODUCTO-CEREAL

Kellogg's®
RICE KRISPIES®

INFORMACIÓN NUTRICIONAL
TAMAÑO DE RACIÓN: 1 ONZA (28.4 g, APROXIMADAMENTE 1 TAZA)
RACIONES POR PAQUETE: 13

	CEREAL	CON 1/2 TAZA LECHE DESCREMADA VITAMINAS A Y D
CALORÍAS	110	150*
PROTEÍNAS	2 g	6 g
CARBOHIDRATOS	25 g	31 g
GRASAS	0 g	0 g*
COLESTEROL	0 mg	0 mg*
SODIO	290 mg	350 mg
POTASIO	35 mg	240 mg

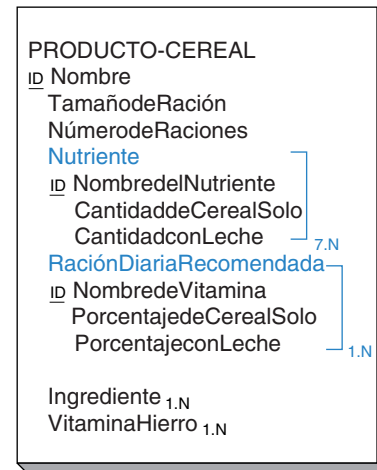
PORCENTAJE DE CANTIDADES DIARIAS QUE RECOMIENDA EL MINISTERIO DE AGRICULTURA DE ESTADOS UNIDOS DE AMÉRICA

PROTEÍNAS	2	10
VITAMINA A	25	30
VITAMINA C	25	25
TIAMINA	35	40
RIBOFLAVINA	35	45
NIACINA	35	35
CALCIO	**	15
HIERRO	10	10
VITAMINA D	10	25
VITAMINA B ₆	35	35
ÁCIDO FÓLICO	35	35
FÓSFORO	4	15
MAGNESIO	2	6
ZINC	2	6
COBRE	2	4

* LA LECHE ENTERA PROPORCIONA 30 CALORÍAS MÁS, 4 g DE GRASA Y 15 mg DE COLESTEROL.
** CONTIENE MENOS DEL 2% DE LA RACIÓN DIARIA RECOMENDADA.

INGREDIENTES: ARROZ, AZÚCAR, SAL, JARABE DE MAÍZ, SABORIZANTE DE MALTA.
VITAMINAS E HIERRO: VITAMINA C (ASCORBATO DE SODIO Y ÁCIDO ASCÓRBICO), NIACINAMIDA, HIERRO, VITAMINA B₆ (HIDROCLORHIDRATO DE PIRIDOXINA), VITAMINA A (PALMITATO), VITAMINA B₂ (RIBOFLAVINA), VITAMINA B₁ (HIDROCLORHIDRATO DE TIAMINA), ÁCIDO FÓLICO Y VITAMINA D. PARA MANTENER FRESCO ESTE CEREAL SE LE AGREGÓ BHT AL EMPAQUE.

(a)



(b)

NUTRIENTE

Nombre	Nombre del Nutriente	Cantidad de Cereal Solo	Cantidad con Leche
--------	----------------------	-------------------------	--------------------

INGREDIENTE

Nombre	Ingrediente
--------	-------------

PRODUCTO-CEREAL

Nombre	Tamaño de Ración	Raciones por Paquete
--------	------------------	----------------------

VITAMINA-HIERRO

Nombre	Vitamina Hierro
--------	-----------------

RACIÓN-DIARIA-RECOMENDADA

Nombre	Nombre de Vitamina	Porcentaje de Cereal Solo	Porcentaje con Leche
--------	--------------------	---------------------------	----------------------

(c)

jetos independientes. Primero, los datos de cada uno están por separado en una sección del formato. Pero lo más importante: cada sección tiene campos que sin duda identifican atributos de algo aparte de AVISO-INFRACCIÓN. Por ejemplo {Licencia de Conducir, Estado} únicamente identifica al conductor; Licencia de Vehículo, Estado y NIV (Número de Identificación del Vehículo) identifican los vehículos registrados, y el

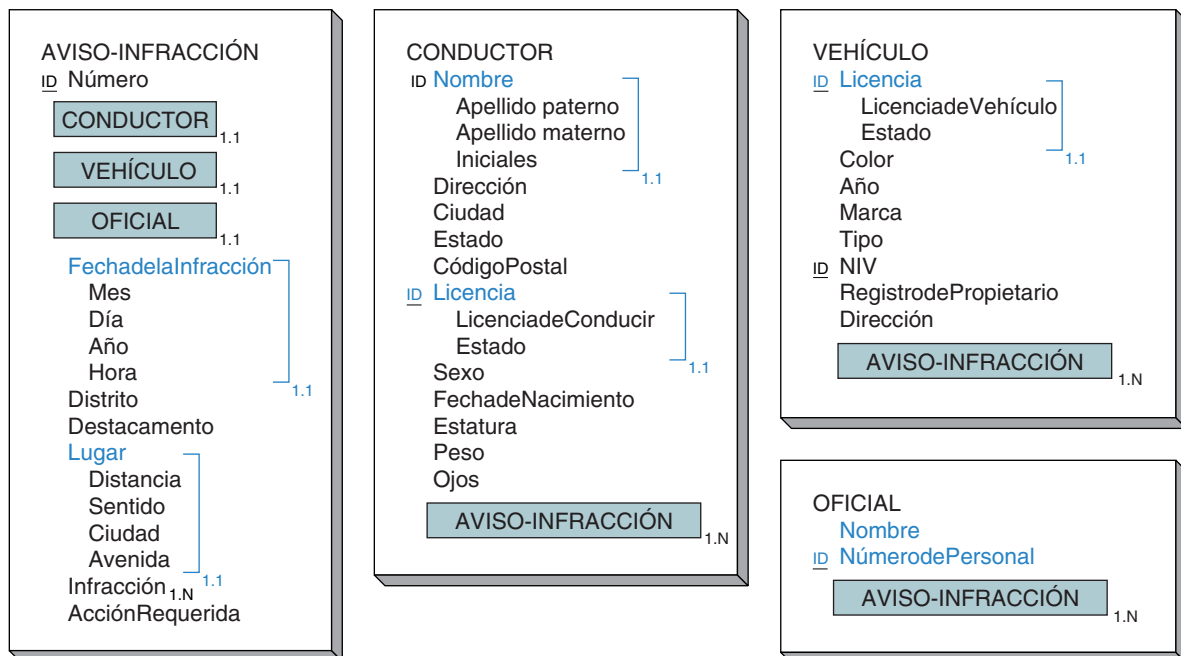
► FIGURA 7-23

Representación de un aviso de infracción: (a) forma de ejemplo, (b) diagrama de objeto de AVISO-INFRACCIÓN, y (c) Representación relacional de AVISO-INFRACCIÓN

AVISO DE INFRACCIÓN. PATRULLA DEL ESTADO DE WASHINGTON

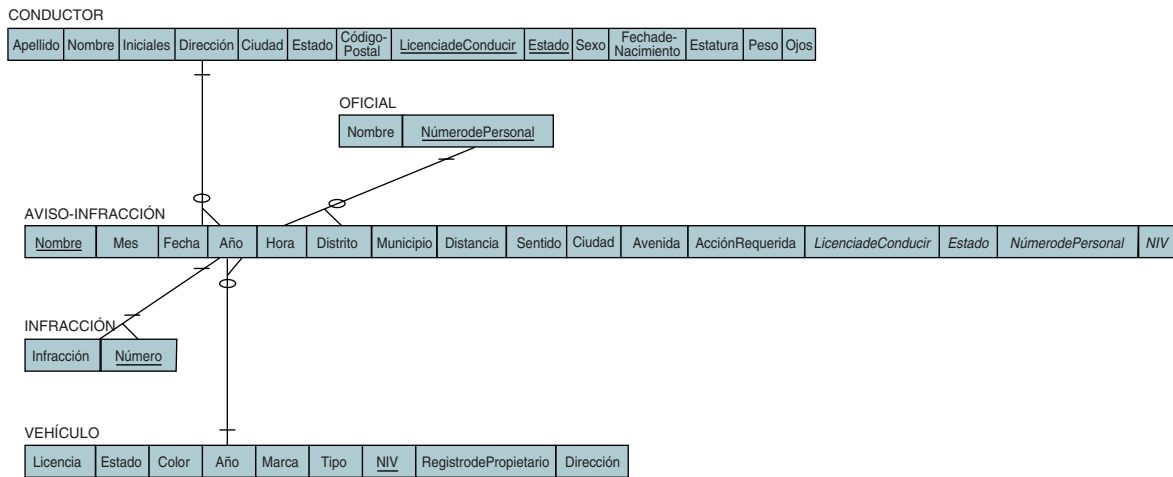
NOMBRE <i>Kroenke</i>		NOMBRE <i>David M</i>	
APELLIDO		NOMBRE	
DIRECCIÓN <i>5053 88 Ave SE</i>			
CIUDAD <i>Mecer Island</i>		ESTADO <i>Wa</i>	CÓDIGO POSTAL <i>98040</i>
LICENCIA DE CONDUCIR <i>00000</i>	ESTADO <i>Wa</i>	FECHA DE NACIMIENTO <i>2/27/46</i>	ESTATURA <i>6</i> PESO <i>165</i> OJOS <i>Cafés</i>
NÚMERO DE MOTOR <i>AAA000</i>	ESTADO <i>Wa</i>	COLOR <i>90</i>	MARCA <i>Saab</i> TIPO <i>900</i>
NIV			
REGISTRO			
PROPIETARIO			
DIRECCION			
FECHA DE LA INFRACCIÓN		DISTRITO DESTACAMENTO	
MES <i>11</i>	DÍA <i>7</i> AÑO <i>2001</i>	HORA <i>935</i>	<i>2 17</i>
LUGAR <i>17 MILLAS E DE Enumckum SOBRE SR410</i>			
VIOLACIONES <i>Escribía mientras manejaba</i>			
FIRMA DEL OFICIAL <i>S Scott</i> NÚMERO DE PERSONAL <i>850</i>			
☒ Ésta es una amonestación; no se requieren acciones posteriores. ☐ Se autoriza el traslado para reparar el vehículo. No se autoriza la reparación en esta vía. ☐ CORRIJA LA(S) VIOLACIÓN(ES) DE INMEDIATO. Regrese firmada esta tarjeta como prueba de cumplimiento dentro de 15-30 días (si se marca este recuadro).			
FIRMA DEL CONDUCTOR <i>[Firma]</i>			

(a)



► FIGURA 7-23

(Continuación)



(c)

NúmerodePersonal identifica a un oficial. Obviamente estos campos llave son determinantes, por lo tanto se definieron objetos para cada uno. En la figura 7-23(c) se ilustra la relación de estos diagramas.

► RESUMEN

La transformación de objetos semánticos en relaciones depende del tipo de objeto. Los simples se representan mediante una relación individual. Los atributos que no son objetos se manejan como atributos de la relación.

Los objetos compuestos requieren dos o más relaciones para su presentación. Una relación contiene atributos de valor único del objeto. Se construye otra relación para cada atributo simple multivaluado o atributo de grupo. La llave de las relaciones que representan los atributos multivaluados siempre es una llave compuesta que contiene la llave del objeto, más un identificador del grupo compuesto dentro de ese objeto.

Se requieren cuando menos dos relaciones para representar un objeto combinado. Cada relación tiene su propia llave distinta. Hay cuatro tipos diferentes de objetos combinados: uno a uno, uno a muchos, muchos a uno y muchos a muchos, que se representan insertando llaves externas. Para las relaciones uno a uno, la llave de cada tabla se coloca en la otra tabla, y para las relaciones uno a muchos y muchos a uno la llave del padre se coloca en la relación hijo. Por último, para las relaciones muchos a muchos se crea una tabla de intersección que contenga las llaves de ambas relaciones.

Los objetos híbridos se representan creando una tabla para el atributo del grupo multivaluado del objeto compuesto, y colocando dentro la llave de la relación que representa al objeto no compuesto. Los cuatro casos de híbridos se enumeran en la figura 7-13.

Los objetos de asociación requieren cuando menos tres relaciones para su representación, una para cada objeto involucrado. Cada relación tiene su propia llave y la relación que representa a los objetos de asociación contiene, como llaves externas, las llaves de los otros dos objetos.

Los objetos padre y subtipo se representan creando una relación para el padre y una para cada subtipo. La llave de todas las relaciones normalmente es la misma. Algunas veces se coloca un atributo identificador en el padre para indicar el tipo del objeto.

Para los objetos arquetipo-versión se crea una relación para el objeto arquetipo y una segunda para la versión. La llave de la relación de la versión siempre contiene la llave del arquetipo.

► PREGUNTAS DEL GRUPO I

- 7.1 Proporcione un ejemplo de un objeto simple, diferente al que se mencionó en este texto. Muestre cómo se representa ese objeto por medio de una relación.
- 7.2 Exponga un ejemplo de objeto compuesto, diferente al de este texto, y diga cómo se representa por medio de relaciones.
- 7.3 Mencione un ejemplo de un objeto combinado 1:1, diferente al de este texto. Muestre dos formas de representarlo por medio de relaciones.
- 7.4 Señale un ejemplo de un objeto combinado 1:N, diferente al de este texto, y muestre cómo representarlo a través de relaciones.
- 7.5 Dé un ejemplo referente a objeto combinado de M:1, diferente al que se mencionó, y muestre cómo representarlo por medio de relaciones.
- 7.6 Provea un ejemplo de un objeto combinado M:N, diferente al de este texto. Muestre cómo representarlo por medio de relaciones.
- 7.7 Proporcione un ejemplo del caso 1 (véase la figura 7-13) de objeto híbrido. Muestre cómo representarlo por medio de relaciones.
- 7.8 Mencione un ejemplo del caso 2 (véase la figura 7-13) sobre objeto híbrido. Muestre cómo representarlo por medio de relaciones.
- 7.9 Dé un ejemplo sobre objetos de asociación y objetos relacionados, diferente a los de este texto. Muestre cómo representarlo por medio de relaciones. Suponga que el objeto de asociación tiene su propio identificador.
- 7.10 Haga lo mismo que en la pregunta 7.9, pero suponga que el objeto de asociación no tiene un identificador propio.
- 7.11 Proporcione un ejemplo sobre un objeto padre, con dos subtipos excluyentes cuando menos, y muestre cómo representarlos mediante relaciones. Use un atributo indicador de tipo.
- 7.12 Mencione un ejemplo de objeto padre con dos subtipos no excluyentes, cuando menos. Muestre cómo representar ambos objetos por medio de relaciones. Use un atributo indicador de tipo.
- 7.13 Encuentre un ejemplo de un formato que se emplee en la universidad a la que usted asiste, el cual esté modelado apropiadamente con un objeto simple. Muestre cómo representarlo por medio de una relación.
- 7.14 Localice un ejemplo sobre un formato de la universidad que esté modelado apropiadamente con un objeto compuesto. Muestre cómo representarlo por medio de relaciones.
- 7.15 Ubique un ejemplo sobre un formato de la universidad que esté modelado apropiadamente con uno de los tipos de un objeto combinado. Muestre cómo representarlos por medio de relaciones.
- 7.16 Descubra un ejemplo de un formato de la universidad que esté modelado apropiadamente con un objeto híbrido. Clasifique el objeto de acuerdo con la figura 7-13 y muestre cómo se representan estos objetos por medio de relaciones.

- 7.17 Localice un ejemplo de un formato de la universidad que esté modelado apropiadamente con objetos de asociación y relacionados. Muestre cómo representar dichos objetos por medio de relaciones.
- 7.18 Encuentre un ejemplo de un formato de la universidad que esté modelado apropiadamente con objetos padre-subtipo. Muestre cómo representarlos mediante relaciones.
- 7.19 Ubique un ejemplo de un formato de la universidad que esté modelado adecuadamente con objetos arquetipo-versión. Muestre cómo representar estos objetos por medio de relaciones.
- 7.20 ¿Qué restricciones de integridad referencial, si las hay, deberían especificarse para los diseños en la figura 7-21(b) y (c)?
- 7.21 ¿Qué restricciones de integridad referencial, si las hay, deberían especificarse para el diseño de la figura 7-22(c)?
- 7.22 ¿Qué restricciones de integridad referencial, si las hay, deberían especificarse para el diseño de la figura 7-23(c)?
- 7.23 Suponga que el objeto O1 tiene una relación 1:N con el objeto O2, y que el O1 tiene una segunda relación 1:N con el objeto O3. Suponga además que se requiere O2 en O1, pero O3 es opcional en O1. ¿Existe alguna diferencia entre la restricción de integridad referencial para la relación de O1 y O2 y la restricción de integridad referencial para la relación de O1 y O3? Si es así, ¿cuál es?

► PREGUNTAS DEL GRUPO II

- 7.24 Con base en la figura 7-13, proporcione un ejemplo diferente para cada uno de los cuatro casos en la columna de la derecha. Muestre cómo cada uno estaría representado con relaciones.
- 7.25 Modifique la figura 7-22(b) y (c) para agregar los reportes que se muestran en la figura 7-24.

► FIGURA 7-24

Reportes para la pregunta 7.25

REPORTE DEL MINISTERIO DE AGRICULTURA #6272 Fecha: 06/30/2001 Emisor: Compañía Kellogg's Título del reporte: resumen de productos por ingrediente	
Maíz	Corn Flakes Krispix Nutrigrain (maíz)
Jarabe de maíz	Rice Krispies Frosted Flakes Sugar Pops
Malta	Rice Krispies Sugar Smacks
Trigo	Sugar Smacks Nutrigrain (trigo)

(a)

LISTA DE PROVEEDORES Fecha: 06/30/2001		
Ingrediente	Proveedor	Precio
Maíz	Wilson	2.80
	J. Perkins	2.72
	Pollack	2.83
	McKay	2.80
Trigo	Adams	1.19
	Kroner	1.19
	Schmidt	1.22
Cebada	Wilson	0.85
	Pollack	0.84

(b)

► FIGURA 7-25

Reporte para la pregunta 7.26

<p>Amor sin barreras Basada en una idea de Jerome Robbins</p>	
<p>Libreto de ARTHUR LAURENTS Música de LEONARD BERNSTEIN Canciones de STEPHEN SONDHEIM</p>	
<p>Dirección y coreografía de la producción original por JEROME ROBBINS</p>	
<p>Producida originalmente en Broadway por Robert E. Griffith y Harold S. Prince, con arreglos de Roger L. Stevens Orquestación de Leonard Bernstein, con Sid Ramin e Irwin Kostal</p>	
<p>PUNTOS CULMINANTES DE TODA LA GRABACIÓN</p>	
<p>María KIRI TE KANAWA Tony JOSÉ CARRERAS Anita TATIANA TROYANOS Riff KURT OLLMAN y MARILYN HORNE cantando "Somewhere"</p>	
<p>Rosalía Louise Edeiken Diesel Marty Nelson Consuela Stella Zambalis Bebé John Stephen Bogardus Fancisca Angelina Reaux A-rab Peter Thom Acción David Livingston Niño de nieve Todd Lester Bernardo. . . . Richard Harrell</p>	
1	Jet Song [3'13] (Riff, Acción, Bebé John, A-rab, Coro)
2	Something's Coming [2'33] (Tony)
3	Maria [2'56] (Tony)
4	Tonight [5'27] (María, Tony)
5	America [4'47] (Anita, Rosalía, Coro)
6	Cool [4'37] (Riff, Coro)
7	One Hand, One Heart [5'38] (Tony, María)
8	Tonight (Conjunto) [3'40] (Todo el elenco)
9	I Feel Pretty [3'22] (María, Coro)
10	Somewhere [2'34] (Una chica)
11	Gee Officer Krupke [4'18] (Acción, Niño de nieve, Diesel, A-rab, Bebé John, Coro)
12	A Boy Like That [2'05] (Anita, María)
13	I Have a Love [3'30] (María, Anita)
14	Taunting Scene [1'21] (Orquesta)
15	Finale [2'40] (María, Tony)

7.26 Usando como guía la cubierta del disco compacto que se muestra en la figura 7-25, realice las siguientes tareas:

- Dibuje los diagramas de objetos para los objetos implícitos ARTISTA, PAPEL y CANCIÓN
 - Identifique las relaciones entre esos objetos. ¿Qué tipos de objetos son? (Simple, compuesto, etcétera.)
 - Indique si cada participante en una relación es opcional u obligatorio
 - Transforme los diagramas de objetos en diagramas de relaciones
- ¿Cuál es la llave para cada relación? ¿Qué llave externa aparece en cada relación?

► PROYECTOS

A. Termine el proyecto A que está al final del capítulo 4, si aún no lo ha hecho. Transforme su modelo de objeto semántico en un conjunto de relaciones. Si cualesquiera de sus relaciones no está en DK/NF, justifique su decisión de crearlas sin normalizar.

B. Concluya el proyecto B que está al final del capítulo 4, en caso de que no lo haya hecho. Transforme su modelo de objeto semántico en un conjunto de relaciones. Si cualesquiera de sus relaciones no está en DK/NF justifique su decisión de crearlas sin normalizar.

C. Termine el proyecto C que está al final del capítulo 4, si es que no lo ha hecho. Transforme su modelo de objeto semántico en un conjunto de relaciones. Si cualesquiera de sus decisiones no está en DK/NF, justifique su decisión de crearlas sin normalizar.

► PREGUNTAS DEL PROYECTO FIREDUP

Si aún no lo ha hecho, cree objetos semánticos para las preguntas A y C en el proyecto FiredUp que se encuentra al final del capítulo 4.

A. Transforme el diseño de objeto semántico de la pregunta A al final del capítulo 4 en un conjunto de relaciones en forma normal dominio-llave. Para cada relación, especifique la llave primaria, llaves candidatas, si las hay, y llaves externas. Especifique todas las restricciones de integridad referencial. Si es necesario, desarrolle y fundamente sus supuestos considerando los implícitos semánticos de la aplicación.

B. Ajuste su respuesta a la pregunta A para permitir las relaciones sin normalizar, si considera que sólo las relaciones son apropiadas. Justifique cualquiera de las relaciones sin normalizar que tenga. Si es necesario, desarrolle y justifique los supuestos considerando los implícitos semánticos de la aplicación.

C. Transforme los diseños de los objetos semánticos de la pregunta C al final del capítulo 4 en un conjunto de relaciones, preferiblemente en forma normal dominio-llave. Si cualquiera de sus relaciones no está en forma normal dominio-llave, explique el porqué. Para cada relación especifique la llave primaria, las llaves candidatas, si las hay, y las llaves externas. Especifique todas las restricciones de integridad referencial.

D. Ajuste su respuesta a la pregunta C, y suponga que casa, fax y teléfono celular están representados por separado, con atributos de valor individual. ¿Este es un mejor diseño que el de su respuesta a la pregunta C? Explique por qué.

IMPLEMENTACIÓN DE BASES DE DATOS CON EL MODELO RELACIONAL

La Parte IV aborda la implementación de bases de datos usando el modelo relacional. El capítulo 8 inicia con un análisis sobre el manejo de datos relacionales. Primero nos enfocamos en los tipos de lenguajes de manejo de datos relacionales y después explicamos los operadores básicos del álgebra relacional, e ilustramos su uso.

El capítulo 9 describe el lenguaje de consulta estructurado, o SQL. Dicho lenguaje tiene el respaldo del American National Standards Institute (ANSI), como estándar para el manejo de bases de datos relacionales, y también es el lenguaje más importante de manejo de datos para los productos DBMS relacionales de comercio. El capítulo 10 concluye esta parte con un análisis sobre el diseño de aplicaciones de bases de datos.

Fundamentos de la implementación relacional

Este capítulo aborda la implementación de las bases de datos relacionales. Comenzaremos por definir los datos relacionales, revisando la terminología correspondiente y explicando cómo se define un diseño con el DBMS. Después regresaremos a la asignación de espacio y creación de datos de la base de datos. El resto del capítulo versará sobre el manejo de datos relacionales: primero se realiza un estudio de los cuatro tipos de lenguajes para el manejo de los datos relacionales (DML, por sus siglas en inglés); enseguida se abordan los tres modos comunes de interfaz DML con el DBMS y, por último, las operaciones básicas de álgebra relacional y consultas mediante ejemplo expresadas en términos de álgebra relacional.

► DEFINICIÓN DE DATOS RELACIONALES

Cuando se implementa una base de datos relacional hay que desarrollar varias tareas. Primero, se debe definir la estructura de una base de datos con el DBMS. Para hacerlo, los programadores usan un lenguaje de definición de datos (DDL, por sus siglas en inglés) o algún otro medio equivalente (como por ejemplo una pantalla) para mostrar la estructura. A continuación le asignan a la base de datos un medio de almacenamiento físico y lo llenan con datos. En esta sección analizaremos cada una de estas tareas, pero primero revisaremos la terminología relacional.

REVISIÓN DE TERMINOLOGÍA

Como se estableció en el capítulo 5, una **relación** es una tabla que tiene varias propiedades:

1. Las entradas en la relación tienen sólo un valor; no se permiten valores múltiples. Por lo tanto, la intersección de un renglón y una columna sólo contienen un valor.
2. Todas las entradas en cualquier columna son de la misma clase. Por ejemplo, una columna puede contener nombres de clientes y otra, de cumpleaños. Cada una tie-

► FIGURA 8-1

Ejemplo de una estructura de la relación PACIENTE

	Col 1 (o atributo 1)	Col 2	Col 3	Col 4	Col 5
	Nombre	Fecha de nacimiento	Género	Número de cuenta	Médico
Renglón 1 (o tuple 1)	Riley	01/19/1946	F	147	Lee
Renglón 2	Murphy	12/28/1981	M	289	Singh
Renglón 3	Krajewski	10/21/1973	F	533	Levy
Renglón 4	Ting	05/23/1938	F	681	Spock
Renglón 5	Dixon	04/15/1987	M	704	Levy
Renglón 6	Abel	06/19/1957	M	193	Singh

ne un nombre único y el orden de las columnas no es importante en la relación. Las columnas de la relación se llaman **atributos**. Cada atributo tiene un **dominio**, que es una descripción física y lógica de los valores permitidos.

3. En la relación no existen dos renglones que sean idénticos, y el orden de los renglones no es importante (véase la figura 8-1). Cada renglón de la relación se conoce como tuple.

La figura 8-1 es un ejemplo, o prueba. El formato generalizado PACIENTE (Nombre, Fecha de Nacimiento, Género, Número de Cuenta, Médico) corresponde a la estructura de la relación y es a lo que la mayoría de las personas se refiere cuando usa el término *relación*. (Recuerde que, como establecimos en el capítulo 5, un atributo subrayado es una llave.) Si agregamos restricciones en los valores de datos permitidos a la estructura de la relación, entonces tenemos un **esquema relacional**. Estos términos se resumen en la figura 8-2.

CONFUSIÓN CON RESPECTO AL TÉRMINO LLAVE. El término **llave** es una fuente común de confusión porque tiene diferentes significados en las etapas de diseño e implementación. Durante el diseño, el término *llave* se refiere a una o más columnas que únicamente identifican un renglón en una relación. Como explicamos en el capítulo 5, sabemos que cada relación tiene una llave porque cada renglón es único; en la restricción, la composición de cada columna en la relación es una llave. Sin embargo, con frecuencia la llave se compone de una o dos columnas.

Durante la implementación, el término *llave* se usa en forma diferente. Para la mayoría de los productos relacionales, una llave es una columna en la cual el DBMS construye un índice u otra estructura de datos. Esto se hace para tener acceso rápido a los renglones por medio de ese valor de columna. Así, las llaves no necesariamente son únicas y, de hecho, con mucha frecuencia no lo son. Éstas se construyen sólo para mejorar el desempeño (véase el apéndice A para información con respecto a estas estructuras de datos).

Por ejemplo, considere la relación PEDIDO (Número de Pedido, Fecha de Pedido, Número de Cliente, Cantidad). Desde el punto de vista del *diseño* relacional, la llave de esta relación es Número de Pedido, ya que el subrayado significa que Número de Pedido identifica en forma única los renglones de la relación. Sin embargo, desde el punto de vista de la *implementación* relacional, cualesquiera de las cuatro columnas pueden ser una llave. Por ejemplo, Fecha de Pedido se puede definir como una llave. De ser así, el DBMS creará una estructura de datos de tal forma que se pueda acceder en forma rápida a los renglones de PEDIDO mediante el valor de Fecha de Pedido. Probablemente habrá muchos renglones para un valor determinado de Fecha de Pedido. Definirlo como llave no dice nada con respecto a su unicidad.

Algunas veces los términos **llave lógica** y **llave física** se usan para distinguir entre ambos significados de llave. Una llave lógica es un identificador único, mientras que una física es una columna que tiene un índice u otra estructura de datos definida, con el fin de mejorar el rendimiento.

► FIGURA 8-2

Resumen de
terminología
relacional

<i>Término</i>	<i>Significado</i>
Relación (o tabla) (o archivo)	Tabla de dos dimensiones.
Atributo (o columna) (o campo) (o elemento de datos)	Columna de una relación.
Tuple (o renglón) (o registro)	Renglón en una relación.
Dominio	Descripción física o lógica de valores permitidos.
Estructura de la relación	Formato de la relación.
Ocurrencia	Estructura de la relación con datos.
Esquema relacional	Estructura de la relación más restricciones.
Llave	Grupo de uno o más atributos que en forma única identifican un tuple en una relación.
Llave lógica	Igual que llave.
Llave física (o índice)	Un grupo de uno o más atributos soportado por una estructura de datos que facilita la rápida recuperación, o un rápido acceso secuencial.

ÍNDICE. Puesto que una llave física por lo general es un índice, algunas personas reservan el término *llave* para una llave lógica y usan *índice* para una llave física. En este texto haremos exactamente eso: usaremos dicho término para mencionar una llave lógica, y el término *índice* para representar una llave física.

Hay tres razones para la definición de índices. Una es permitir el acceso en forma rápida a los renglones por medio del valor del atributo indizado. Otra es facilitar el ordenamiento de renglones de acuerdo con ese atributo. Por ejemplo, en PEDIDO, FechadePedido puede estar definida como llave, de tal forma que pueda generarse rápidamente un reporte que muestre los pedidos por fecha.

Una tercera razón para la construcción de un índice es imponer la unicidad. A pesar de que los índices no tienen que ser únicos, cuando el programador quiere que una columna lo sea, se crea un índice por medio del DBMS. Con la mayoría de los productos DBMS relacionales se puede imponer que una columna, o grupo de columnas, tenga carácter de única usando la palabra llave UNIQUE cuando se defina la aparición de una columna en una tabla.

IMPLEMENTACIÓN DE UNA BASE DE DATOS RELACIONAL

En este texto usamos el modelo relacional para expresar el diseño de bases de datos. Puesto que así lo hemos hecho, podemos proceder directamente a partir del diseño de la base de datos hasta implementarla. No es necesario transformar el diseño durante la etapa de implementación, simplemente definimos el diseño relacional que existe en el DBMS.

La situación es diferente cuando implementamos bases de datos usando productos DBMS basados en modelos de datos distintos al modelo relacional. Por ejemplo, cuando se implementa una base de datos para DL/I, debemos convertir el diseño relacional en uno jerárquico, y después definir el diseño convertido al producto DBMS.

DEFINICIÓN DE LA ESTRUCTURA DE BASE DE DATOS AL DBMS. Existen diferentes medios para describir la estructura de bases de datos en el DBMS, dependiendo del producto de DBMS que se use. Con algunos productos se construye un archivo de texto que describa la estructura de la base de datos. El lenguaje que se usa para describirla a veces se conoce como **lenguaje de definición de datos**, o DDL (Data Definition Language, por sus siglas en inglés). El archivo de texto DDL da nombre a las tablas en la base de datos, nombra y describe las columnas de esas tablas, define los índices y describe otras estructuras tales como restricciones y restricciones de seguridad. La figura 8-3 muestra el lenguaje de definición de datos típico que se usa para definir una base de datos relacional simple en el caso de un DBMS hipotético. En los capítulos 12 y 13 se muestran ejemplos más reales usando una norma llamada SQL.

Algunos productos DBMS no requieren que un DDL defina la base de datos en el formato de archivo de texto. Una alternativa común es proporcionar un medio gráfico para definir la estructura de la base de datos. Por ejemplo, con Access 2002 se le muestra al programador una estructura gráfica de lista y se le pide que llene la tabla y los nombres de las columnas en los lugares apropiados. En el capítulo 2 vimos un ejemplo de esto (figura 2-4).

En general, las facilidades de definición gráfica son comunes para los productos DBMS en las computadoras personales. Tanto los DDL textuales como gráficos son usuales en productos DBMS, en servidores y macrocomputadoras. Por ejemplo, se emplean tanto servidores de ORACLE como de SQL. La figura 8-4 resume los procesos de definición de bases de datos.

Independientemente de los medios por los cuales se defina la estructura de la base de datos, el programador debe nombrar cada tabla, definiendo las columnas en ésta, así como el formato físico (por ejemplo TEXTO 10) de cada una. Así mismo, dependiendo de los medios del DBMS, el programador puede especificar las restricciones que se imponen al DBMS. Por ejemplo, se pueden establecer los valores de columna como NO NULO (NOT NULL o UNIQUE). Algunos productos también permiten definir el rango y las restricciones de valor (partes menores que 10000, o color igual a uno de ['Rojo', 'Verde', 'Azul']). Por último, se pueden establecer restricciones de interrelación. Un ejemplo es que NúmerodeDepartamento en EMPLEADO debe coincidir con un valor de NúmerodeDepartamento en DEPARTAMENTO.

FIGURA 8-3

Ejemplo de un archivo de texto DDL para la definición de una base de datos

```
CREATE SCHEMA MÉDICO

CREATE TABLE PACIENTE
(Nombre                CHARACTER VARYING (35) NOT NULL,
 Fecha de nacimiento  DATE/TIME,
 Género                CHARACTER VARYING (10),
 NúmerodeCuenta        INTEGER NOT NULL,
 NombredelMédico_FK1   CHARACTER VARYING (35) NOT NULL,

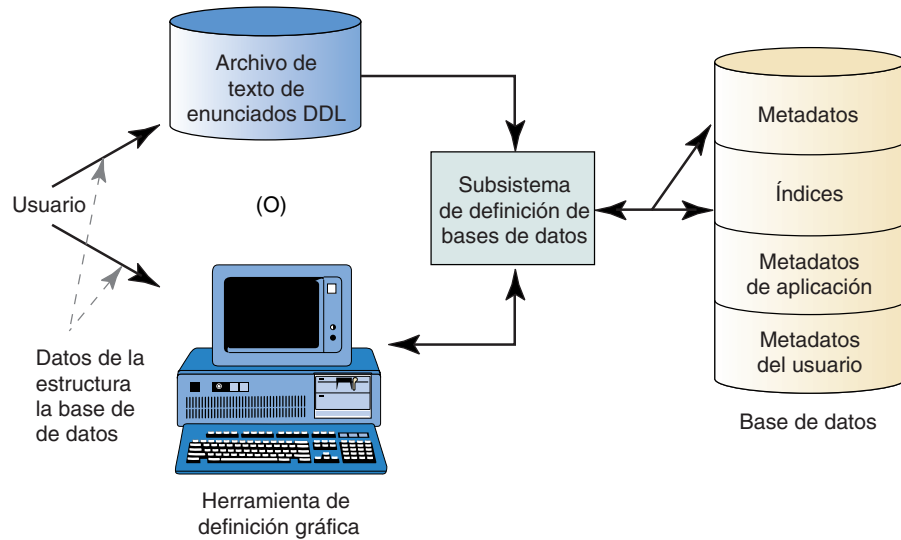
 PRIMARY KEY (NúmerodeCuenta)
 FOREIGN KEY (NombredelMédico_FK1)
 REFERENCES MÉDICO
 )

CREATE TABLE MÉDICO
(NombredelMédico        CHARACTER VARYING (35) NOT NULL,
 CódigodeÁrea           CHARACTER VARYING (3),
 NúmeroLocal            CHARACTER VARYING (8) NOT NULL,

 PRIMARY KEY (NombredelMédico)
 )
```

► FIGURA 8-4

Proceso para la
definición de
la base de datos



Con muchos productos, el programador también puede definir contraseñas (*passwords*) y otros medios de control y seguridad. Como se muestra en el capítulo 11, se pueden usar diferentes estrategias. Algunas colocan controles en las construcciones de datos (por ejemplo, contraseñas en tablas) y otras los colocan en las personas (el usuario de la contraseña X puede leer y actualizar las tablas T1 y T2).

ASIGNACIÓN DE ESPACIO. Además de definir la estructura de una base de datos, el programador debe asignar estructuras de base de datos a medios físicos. Otra vez, las tareas específicas dependerán del producto particular DBMS que se use. Para una base de datos personal todo lo que se necesita hacer es asignar la base de datos a un directorio y darle a ésta un nombre. Entonces el DBMS asigna espacio de almacenamiento en forma automática.

Otros productos DBMS, especialmente los que se emplean en servidores y macrocomputadoras, requieren más trabajo. Para mejorar el rendimiento y el control se debe planear con cuidado la distribución de los datos en la base, tanto en el disco como en los canales. Por ejemplo, dependiendo de la naturaleza del procesamiento de la aplicación, puede ser recomendable ubicar ciertas tablas en el mismo disco, o puede ser importante asegurarse de que éstas no sean colocadas en el mismo disco.

Considere, por ejemplo, un objeto de pedido que está conformado por los datos de las tablas PEDIDO, ARTÍCULO-LÍNEA y ARTÍCULO. Suponga que cuando se procesa un PEDIDO la aplicación recupera un renglón desde PEDIDO, varios de ARTÍCULO-LÍNEA y un renglón de ARTÍCULO por cada uno de ARTÍCULO-LÍNEA. Además, los renglones ARTÍCULO-LÍNEA de un pedido específico tienden a estar agrupados, pero los de ARTÍCULO no estarán todos agrupados. La figura 8-5 ilustra lo anterior.

Ahora suponga que una organización procesa en conjunto muchos pedidos y tiene un disco grande y rápido y otro pequeño y lento. El programador debe determinar cuál es el mejor lugar para colocar los datos. Una posibilidad es que el desempeño mejore si la tabla ARTÍCULO se almacena en el disco más grande y más rápido, y los datos de PEDIDO y ARTÍCULO-LÍNEA se ubiquen en el más pequeño y lento. Quizás el desempeño mejorará si los datos de PEDIDO y ARTÍCULO-LÍNEA de los pedidos de meses anteriores se colocan en el disco lento y todos los datos de los pedidos de este mes en el disco rápido.

No podemos responder aquí esta pregunta, ya que la respuesta depende de la cantidad de datos, de las características de procesamiento del DBMS y del sistema operativo, del tamaño y la velocidad de los discos y canales, y de los requisitos para el procesamiento de todas las aplicaciones que usa la base de datos. Lo importante es que se deben considerar estos factores cuando se asigne el espacio a los medios de la base de datos.

Además de especificar la ubicación y cantidad de espacio para los datos de usuario, el programador también puede necesitar declarar si el espacio de archivo se incrementa-

► FIGURA 8-5

Datos de ejemplo de tres tablas que representan un PEDIDO

Tabla PEDIDO		Tabla ARTÍCULO-LÍNEA			Tabla ARTÍCULO	
Número-PEDIDO		Número-Pedido	Número-LÍNEA	Número-ARTÍCULO	Número-ARTÍCULO	DESCRIPCIÓN-ARTÍCULO
100	└─┐	100	1	10	10	A
200		100	2	70	20	B
300		100	3	50	30	C
		200	1	50	40	D
		200	2	10	50	E
	└─┐	300	1	60	60	F
		300	2	10	70	G
		300	3	50		
		300	4	20		
		300	5	30		

Nota: Para un pedido específico, los renglones ARTÍCULO-LÍNEA están agrupados, pero los renglones ARTÍCULO no.

rá o no cuando sea necesario y, si es así, cuánto aumentará. Por lo general, esa cantidad de espacio adicional se expresa como una cantidad específica, o como un porcentaje del espacio inicial.

En la creación de la base de datos, el programador también necesitará ubicar el espacio de archivo para los registros de la base de datos. Usted aprenderá acerca de los registros en los capítulos 11 al 13; por ahora sólo tiene que darse cuenta de que el DBMS registrará cambios de datos que se pueden usar más tarde para recuperar la base de datos en caso de que sea necesario. El espacio para estos registros se define cuando se crea la base de datos.

CREACIÓN DE UN PLAN DE MANTENIMIENTO PARA LA BASE DE DATOS.

Un plan de mantenimiento es una planificación de actividades que se llevarán a cabo de manera repetitiva. Estas tareas incluyen: respaldo de la base de datos, vaciado del contenido de los registros de la base en archivos de respaldo, verificación de violaciones de integridad referencial, optimización del espacio en disco para los datos del usuario e índices, etc. Este tema lo abordaremos en el capítulo 11, por lo pronto estemos conscientes de que se debe desarrollar un plan de mantenimiento cuando se crea la base de datos, o un poco después.

CREACIÓN DE LA BASE DE DATOS. Una vez que se ha definido la base de datos y ha sido ubicada en un almacenamiento físico, se puede alimentar con datos. Los medios a través de los cuales se realiza esto dependen de los requisitos de la aplicación y de las características de los productos DBMS. En el mejor de los casos, todos los datos ya están en un formato entendible para la computadora y el DBMS tiene características y herramientas que facilitan la importación de los datos desde medios magnéticos. En el peor de los casos, todos los datos se deberán introducir manualmente mediante el uso de programas de aplicación que desarrollen los programadores. La mayoría de las conversiones de datos están entre estos dos extremos.

Una vez que se introducen los datos, se deben revisar con esmero. La verificación es una tarea intensiva y tediosa, pero muy importante. Con frecuencia, especialmente en el caso de grandes bases de datos, es bueno valorar el tiempo y el gasto del equipo de desarrollo, así como escribir programas de verificación. Dichos programas cuentan el número de registros de varias categorías, calculan los totales de control, realizan comprobaciones razonables con respecto a los valores de datos de los artículos y proporcionan otros tipos de verificación.

► MANEJO DE DATOS RELACIONALES

Hasta aquí hemos analizado el diseño de bases de datos relacionales y los medios a través de los cuales se definen en el DBMS. Siempre que nos hemos referido al procesamiento de relaciones lo hemos hecho de una manera general e intuitiva. Aunque está

bien en el caso de los diseños analizados, para implementar aplicaciones necesitamos lenguajes claros y no ambiguos con el fin de expresar la lógica del procesamiento. Estos lenguajes se llaman **lenguajes de manipulación de datos (DML)**.

CATEGORÍAS DE LENGUAJES DE MANIPULACIÓN DE DATOS RELACIONALES

A la fecha se han propuesto cuatro estrategias diferentes de manejo de datos relacionales: el **álgebra relacional**, que es la primera de estas estrategias, define operadores que trabajan en relaciones (de manera similar a los operadores +, -, etc., del álgebra de secundaria). Las relaciones se pueden manipular usando estos operadores para lograr el resultado deseado. Pero el álgebra relacional es difícil de usar, en parte porque está basada en procedimientos; esto es, cuando se usa álgebra relacional debemos saber no sólo *qué* queremos, sino *cómo* obtenerlo.

En el proceso de bases de datos comerciales no se usa el álgebra relacional. A pesar de que los productos DBMS no comerciales exitosos proporcionan facilidades de álgebra relacional, aquí analizaremos ésta porque ayuda a aclarar el manejo relacional y a establecer las bases para aprender SQL.

El **cálculo relacional** es un segundo tipo de manejo de datos relacionales. No se basa en procedimientos; es un lenguaje para decir lo que queremos sin expresar cómo obtenerlo. Recordemos la variable de integración en cálculo, cuyo rango se extiende sobre el intervalo de integración. El cálculo relacional tiene una variable similar. Para el cálculo relacional de tuples la variable se extiende sobre los tuples de la relación, y para el cálculo relacional del dominio la variable se extiende sobre los valores de un dominio. El cálculo relacional se deriva de una rama de las matemáticas llamada cálculo de predicados.

A menos que usted se vaya a convertir en un teórico de la tecnología relacional, probablemente no necesitará aprender este tipo de cálculo. Nunca se usa en el procesamiento comercial de bases de datos, y para nuestros fines no es necesario aprenderlo, así que no lo analizaremos.

Aunque el cálculo relacional es difícil de entender y usar, su característica respecto a que no está basado en procedimientos es muy conveniente. Por lo tanto, los diseñadores DBMS buscaron otras técnicas no basadas en procedimientos, lo cual condujo a la tercera y cuarta categorías de DML relacionales.

Los **lenguajes orientados a la transformación** son una clase de lenguajes no basados en procedimientos que transforman datos de entrada expresados como relaciones en resultados que se expresan como una relación independiente. Estos lenguajes proporcionan estructuras fáciles de usar para manifestar lo que es conveniente, considerando los datos proporcionados. SQUARE, SEQUEL y SQL son lenguajes orientados a la transformación. Estudiaremos SQL con detenimiento en los capítulos 9, 12 y 13.

La cuarta categoría de DML relacional es gráfica. **Query-by-Example** (consulta mediante ejemplo) y **Query-by-Form** (consulta por forma) entran en esta categoría. Los productos que se basan en esta categoría son Approach (de Lotus) y Access. Mediante una interfaz gráfica, se ofrece al usuario la materialización de una o más relaciones, la cual podría ser una forma de ingreso de datos, una hoja de cálculo, o alguna otra estructura. Los DBMS asocian la materialización a la relación implícita y construyen consultas (probablemente en SQL) en beneficio del usuario. Los usuarios entonces hacen que las instrucciones DML se ejecuten, pero no tienen conocimiento de este hecho. Las cuatro categorías de los DML relacionales se listan en la figura 8-6.

► FIGURA 8-6

Cuatro categorías de DML relacional

- Álgebra relacional
- Cálculo relacional
- Lenguajes orientados a la transformación (como SQL)
- Query-by-example/Query-by-form

► FIGURA 8-7

Ejemplo de una forma de pantalla tabular predeterminada

Nombre	Fecha de Nacimiento	Género	Número de Cuenta	Medio
Riley	1/19/46	F	147	Lee
Abel	6/19/57	M	193	Singh
Murphy	12/28/81	M	289	Singh
Krajewski	10/21/73	F	533	Levy
Ting	5/23/38	F	661	Spock
Dixon	4/15/87	M	704	Levy

INTERFACES DML CON EL DBMS

En esta sección consideramos cuatro interfaces diferentes para manejar datos de la base.

MANIPULACIÓN DE DATOS MEDIANTE FORMAS. La mayoría de los productos DBMS relacionales incluyen herramientas para construir formas. Algunas se crean automáticamente cuando se define una tabla, pero otras las debe crear el programador, quizás con ayuda inteligente como la que proporcionan los Wizards, de Access. Una forma puede ser tabular, por ejemplo una hoja de cálculo, en cuyo caso mostrará múltiples renglones a la vez, o puede mostrar cada renglón como una entidad independiente. Las figuras 8-7 y 8-8 muestran un ejemplo de cada una para la tabla PACIENTE (PATIENT) en la figura 8-1. Con la mayoría de los productos, se proporciona alguna flexibilidad en el procesamiento de formas y reportes. Por ejemplo, se pueden seleccionar renglones para el procesamiento basado en valores de columnas y también se pueden ordenar. La tabla de la figura 8-7 está ordenada por Número de Cuenta.

Muchas de las formas predeterminadas presentan datos de una sola relación a la vez. Si se requieren datos de dos o más relaciones, entonces por lo general se fabrican formas que se deben crear usando herramientas DBMS. Es posible crear ambas formas multitabla y multirenglón usando estas herramientas. Sin embargo, el uso de dichas herramientas depende del producto, así que por lo tanto no las analizaremos.

INTERFAZ DE LENGUAJE CONSULTA-ACTUALIZACIÓN. El segundo tipo de interfaz para una base de datos es a través de un **lenguaje de consulta-actualización**, o simplemente **lenguaje de consulta** (aunque la mayoría de estos lenguajes realizan tanto la consulta como la actualización, por lo general se les conoce como lenguajes de consulta). Con este tipo de lenguaje el usuario introduce órdenes de consulta que especifican acciones en la base de datos. Los DBMS decodifican las órdenes y llevan a cabo las acciones apropiadas. La figura 8-9 muestra los programas involucrados en el procesamiento de consultas.

El lenguaje de consulta más importante es SQL. Para dar una idea sobre los lenguajes de consulta, considere la siguiente instrucción de SQL que procesa la relación PACIENTE, la cual se muestra en la figura 8-1:

```
SELECT Nombre, Fecha de Nacimiento
FROM PACIENTE
WHERE Médico = 'Levy'
```

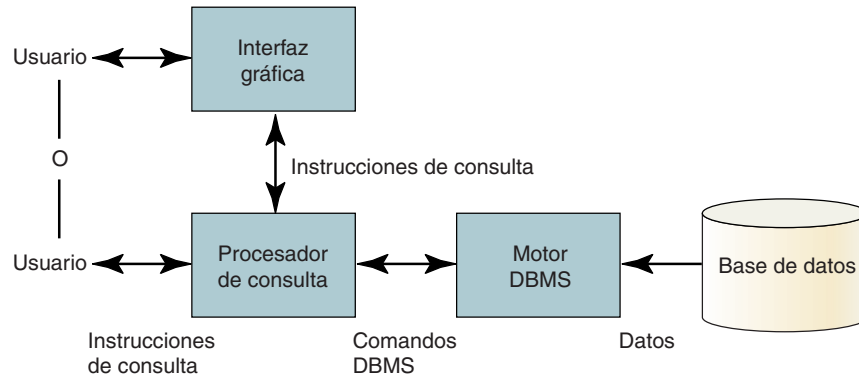
► FIGURA 8-8

Ejemplo de una forma de pantalla de un renglón individual predeterminado

The screenshot shows a form window titled 'PATIENT'. It contains several input fields with the following data: Name (Krajewski), Date of Birth (10/21/73), Gender (F), Account Number (533), and Physician (Levy). At the bottom of the form, there is a 'Record:' label followed by the number '4' and a set of navigation buttons (back, forward, etc.).

► FIGURA 8-9

Programas involucrados en el procesamiento de consultas



Esta instrucción de consulta SQL extrae todos los renglones de la relación PACIENTE en la que el médico es 'Levy'. En una segunda tabla despliega Nombre y Fecha de Nacimiento para los renglones que cumplen esta condición.

PROCEDIMIENTOS ALMACENADOS (STORED PROCEDURES). Con el tiempo, los usuarios y los programadores de bases de datos encuentran que ciertas secuencias de órdenes SQL necesitan realizarse periódicamente. El único cambio de un periodo a otro es el uso de diferentes valores en las cláusulas WHERE. Por ejemplo, factura de fin de mes implica procesar la misma instrucción SQL, pero con una fecha de cierre diferente. Para atender esta necesidad, los proveedores de DBMS desarrollaron **procedimientos almacenados**, los cuales son conjuntos de instrucciones SQL que se guardan como un archivo al que se puede llamar con una sola orden. Cuando se invoca los parámetros pueden pasar por ese procedimiento para ser llenados en las cláusulas WHERE, etc. Un ejemplo de su uso es:

DO BILLING STORED_PROCEDURE FOR BILLDATE = "9/1/2000"
 (HACER PROCEDIMIENTO DE FACTURACIÓN ALMACENADA PARA FECHA DE FACTURA = "19/1/2000")

A medida que los programadores obtienen experiencia el problema se aclara. SQL fue creado como un sublenguaje de datos y no como un conjunto de características de programación. Sin embargo, algunos de los aspectos faltantes eran necesarios para almacenar procedimientos, por lo cual los fabricantes de DBMS ampliaron a SQL para agregarlos. Uno de tales lenguajes, PL/SQL, lo desarrolló Oracle; otro, llamado TRANSACT-SQL, lo creó SQL Server. En los capítulos 12 y 13 aprenderá sobre estos lenguajes.

Un tipo especial de procedimientos almacenados, llamado **disparador** (trigger), lo invoca el DBMS cuando tiene lugar una condición específica. Por ejemplo, en una aplicación de entrada de órdenes, un programador puede crear un disparador que se active siempre que Cantidad Disponible de un artículo en un inventario esté por debajo de su asociado Cantidad ReOrdenada. Aprenderá más acerca de procedimientos almacenados y disparadores en los capítulos 12 y 13.

INTERFAZ DE PROGRAMAS DE APLICACIÓN. El cuarto tipo de interfaz para el ingreso de datos se encuentra en los programas de aplicación escritos en lenguajes de programación tales como COBOL, BASIC, Perl, Pascal y C++. Además, algunos programas de aplicación están escritos en lenguajes que proporcionan los fabricantes de DBMS, entre los cuales el lenguaje de programación de dBASE es el más conocido.

Existen dos estilos de interfaz de programas de aplicación para los DBMS. En uno, el programa de aplicación invoca rutinas de una biblioteca de funciones que forma parte del DBMS. Por ejemplo, para leer determinado renglón de una tabla, dicho programa llama a una función de lectura del DBMS y transmite parámetros que indican la tabla a la que se debe acceder, los datos que hay que recuperar, el criterio de selección de renglones, etcétera.

En algunos casos la sintaxis de la orientación de objetos es más usada que las llamadas de función. En el siguiente código Access, el objeto de referencia *db* se coloca

en la actual base de datos abierta y un segundo objeto de referencia *rs* apunta a los renglones de la tabla PATIENT (PACIENTE).

```
set db = currentdb( )
set rs = db.OpenRecordset("PATIENT")
```

Entonces se puede tener acceso a las propiedades del *record set* abierto y se pueden ejecutar métodos usando la variable de referencia. Por ejemplo, la propiedad *rs.AllowDeletions* se puede referenciar para decidir cuáles registros en PACIENTE se pueden eliminar. El método *rs.MoveFirst*, se puede usar para posicionar un cursor en el primer renglón.

El segundo y más antiguo estilo de interfaz a veces se usa en macrocomputadoras y productos de servidores DBMS. En este caso, el vendedor del DBMS define un conjunto de comandos de acceso de datos de alto nivel. Estos comandos, que son particulares del procesamiento de bases de datos y no son parte de cualquier lenguaje estándar, están implantados dentro del código de los programas de aplicación.

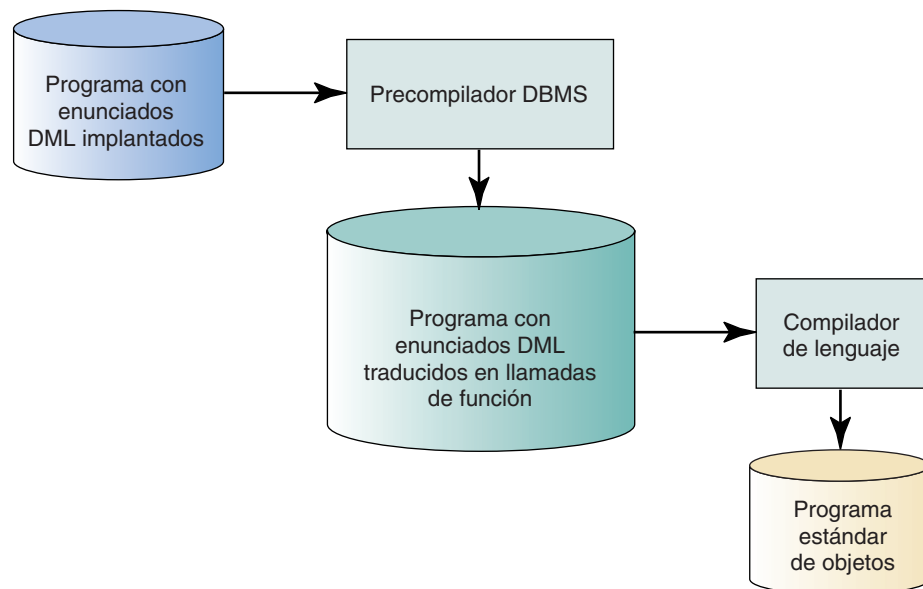
El programa de aplicación, con órdenes implantadas, es sometido entonces a un precompilador que proporciona el vendedor de DBMS. Este precompilador traduce las instrucciones de acceso de datos en llamadas de funciones válidas y definiciones de estructura de datos. En este proceso, el precompilador establece secuencias de parámetros para las llamadas y define áreas de datos que compartirán los programas de aplicación y el DBMS. El precompilador también inserta programas lógicos para mantener las áreas de datos. Así, la rutina del precompilador se somete al compilador de lenguaje. En la figura 8-10 se muestran las relaciones de los programas que involucra este proceso.

Además de su papel en el procesamiento de consultas, SQL también se usa como un lenguaje en programas de aplicación. Así, las instrucciones de SQL se implantan en los programas y se traducen en funciones de llamadas mediante un precompilador. Los costos de entrenamiento y el tiempo de aprendizaje se reducen debido a que se puede usar el mismo lenguaje para acceder tanto a la consulta como a los programas de aplicación. Sin embargo, hay un problema: SQL es un lenguaje orientado a la transformación que acepta relaciones, las maneja y arroja una relación resultante. Por lo tanto, aborda una relación a la vez. Casi todos los programas de aplicación están orientados a los renglones (registros); esto es, leen un renglón, lo procesan, leen el siguiente renglón, y así sucesivamente. Estos programas trabajan con un renglón a la vez.

Por ende, existe un desajuste entre la orientación básica de SQL y los lenguajes de programas de aplicación. Para corregir dicho desajuste, se supone que en el programa

► FIGURA 8-10

Programa de procesamiento con enunciados DML implantados



de aplicación los resultados de las instrucciones de SQL se consideran como archivos. Para ilustrar esto suponga que los siguientes enunciados SQL (los mismos de antes) están insertados en el programa de aplicación:

```
SELECT  Nombre, FechadeNacimiento
FROM    PACIENTE
WHERE   Médico = 'Levy'
```

El resultado de estos enunciados es una tabla que tiene dos columnas y N renglones. Con el fin de aceptar los resultados de esta consulta, el programa de aplicación está escrito para suponer que esos enunciados han producido un archivo con N registros. La aplicación abre la consulta, procesa el primer renglón, procesa el siguiente renglón, y así sucesivamente, hasta que el último ha sido procesado. Esta lógica es la misma que para el procesamiento de un archivo secuencial. Verá ejemplos de estos programas de aplicación en los capítulos 12, 13, 15 y 16. Por ahora, sólo tenga en mente que existe un desajuste en la orientación básica de SQL (orientado a relación) y los lenguajes de programación (orientados a renglones, o registros), y que ese desajuste se debe corregir cuando los programas acceden a una base de datos relacional a través de SQL.

► ÁLGEBRA RELACIONAL

El álgebra relacional es similar al álgebra que aprendió en la preparatoria, pero con una diferencia importante: en el álgebra de la preparatoria las variables representan números, y los operadores como $+$, $-$, $\times$, y $/$ operan sobre cantidades numéricas. Sin embargo, en el álgebra relacional las variables son relaciones y los operadores las manejan para formar nuevas relaciones. Por ejemplo, la operación de unión combina los tuples de una relación con los de otra, con lo cual producen una tercera relación. De hecho, el álgebra relacional es *cerrada*, lo cual significa que el resultado de una o más operaciones relacionales son *siempre* una relación.

Las relaciones son conjuntos. Los tuples de una relación se pueden considerar elementos de un conjunto y, por lo tanto, las operaciones que se pueden realizar en conjuntos también se pueden desarrollar en relaciones. Primero mostraremos cuatro de estos operadores de conjuntos y después analizaremos otros que son particulares del álgebra relacional. Sin embargo, antes de proceder considere las siguientes relaciones muestra que usaremos en este capítulo y en el siguiente.

OPERADORES RELACIONALES

En la figura 8-11 se muestran seis relaciones, sus atributos y definiciones de dominio. Observe que el atributo Nombre se usa en varias relaciones. Cuando nos referimos a un atributo específico, lo calificamos con el nombre de la relación. De acuerdo con lo anterior, Nombre en CLASE a veces se denota como CLASE.Nombre.

En el siguiente análisis los valores de caracteres se muestran entre comillas y los caracteres no entrecomillados representan nombres. Así “HABITACIÓN” difiere de Habitación porque “HABITACIÓN” es un valor, en tanto que Habitación es, por así decirlo, un nombre de dominio. Con respecto a datos numéricos, los números no entrecomillados se refieren a cantidades numéricas, y los entrecomillados, a cadenas de caracteres. Por lo que 123 es un número y “123” es una cadena de los caracteres “1”, “2” y “3”.

UNIÓN. La **unión** de dos relaciones está formada por la adición de tuples de una relación con los de una segunda relación que produce una tercera. El orden en el que aparecen los tuples en la tercera relación no es importante, pero se deben eliminar los que estén duplicados. La unión de relaciones A y B se denota por $A + B$.

Para que esta operación tenga sentido, las relaciones deben ser **compatibles en la unión**; esto es, cada relación debe tener el mismo número de atributos, y los atributos en las columnas correspondientes deben provenir del mismo dominio. Si, por

► FIGURA 8-11

Ejemplos de relaciones y dominios:
(a) definiciones de relaciones,
(b) atributos y dominios, y
(c) definiciones de dominios

1. JUNIOR (Enum, Nombre, Especialidad)
2. ESTUDIANTE-HONOR (Número, Nombre, Interés)
3. ESTUDIANTE (EID, Nombre, Especialidad, Grado)
4. CLASE (Nombre, Horario, Aula)
5. INSCRIPCIÓN (NúmeroEstudiante, NombreClase, NúmeroPosición)
6. FACULTAD (FID, Nombre, Departamento)

(a)

Atributo	Dominio
1. Enum JUNIOR.Nombre Especialidad	IdentificadordePersonas NombresdePersonas NombresdeMaterias
2. Número ESTUDIANTE-HONOR.Nombre Interés	IdentificadordePersonas NombresdePersonas NombresdeMaterias
3. EID ESTUDIANTE.Nombre Especialidad Grado	IdentificadordePersonas NombresdePersonas NombresdeTemas Clases
4. CLASE.Nombre Horario Aula	NombresdeClases HorariodeClases Aulas
5. NúmerodeEstudiante NombredeClase NúmeroPosición	IdentificadoresdePersonas NombresdeClases TamañodeClases
6. FID FACULTAD.Nombre Departamento	IdentificadoresdePersonas NombresdePersonas NombresdeTemas

(b)

Nombre del dominio	Formato
IdentificadoresdePersonas	Decimal (3)
NombresdePersonas	Carac (8) (no real, pero manejable en el caso de estos ejemplos)
NombresdeTemas	Carac (10)
Clases	Uno de [FR, SO, JR, SN, GR]
NombresdeClases	Carac (10)
HorariosdeClases	Carac (5) formato: DDDHH, donde D es uno de [M, T, W, R, F, o blanco], y HH es decimal entre 1 y 12
Aulas	Carac (5) formato: BBRRR, donde BB es un código construido y RRR es un número de aula
TamañodeClases	Decimal desde 0 hasta 100

(c)

► FIGURA 8-12

Relaciones y uniones de JUNIOR y ESTUDIANTE-HONOR: (a) ejemplo de Relación JUNIOR, (b) ejemplo de relación ESTUDIANTE-HONOR, y (c) relación de unión de JUNIOR y ESTUDIANTE-HONOR

Enum	Nombre	Especialidad
123	JONES	HISTORIA
158	PARKS	MATEMÁTICAS
271	SMITH	HISTORIA

(a)

Número	Nombre	Interés
105	ANDERSON	ADMINISTRACIÓN
123	JONES	HISTORIA

(b)

Enum o Número	Nombre	Especialidad o interés
123	JONES	HISTORIA
158	PARKS	MATEMÁTICAS
271	SMITH	HISTORIA
105	ANDERSON	ADMINISTRACIÓN

(c)

ejemplo, el tercer atributo de una relación proviene del dominio Aulas, el tercer atributo de la segunda relación también debe provenir del dominio Aulas.

En la figura 8-11 las relaciones JUNIOR y ESTUDIANTE-HONOR son compatibles en la unión porque ambas tienen tres atributos, los cuales provienen del mismo dominio. JUNIOR.Enum y ESTUDIANTE-HONOR.Nombre tienen el dominio IdentificadoresdePersona; JUNIOR.Nombre y ESTUDIANTE-HONOR.Nombre tienen el dominio NombresdePersonas; y JUNIOR.Especialidad y ESTUDIANTE-HONOR.Interés tienen el dominio Nombredemateria. Las relaciones JUNIOR y CLASE poseen tres atributos cada una, pero tienen **incompatibilidad de la unión** porque los tres atributos no tienen el mismo dominio.

En la figura 8-12 se muestra la unión de las relaciones JUNIOR y ESTUDIANTE-HONOR. Observe que el tuplo [123, JONES, HISTORIA], el cual ocurre en ambas relaciones, no está duplicado en la unión.

DIFERENCIA. La **diferencia** de dos relaciones es una tercera relación que contiene tuples que están presentes en la primera relación, pero no en la segunda. Las relaciones deben ser compatibles en la unión. La diferencia de JUNIOR y ESTUDIANTE-HONOR se muestra en la figura 8-13. Al igual que en aritmética, el orden de la sustracción es importante y por lo tanto $A - B$ no es lo mismo que $B - A$.

INTERSECCIÓN. La **intersección** de dos relaciones es una tercera relación que contiene los tuples que aparecen tanto en la primera como en la segunda relación. Una vez más, las relaciones deben ser compatibles en la unión. En la figura 8-14 la intersec-

► FIGURA 8-13

Relación JUNIOR menos ESTUDIANTE-HONOR

Enum	Nombre	Especialidad
158	PARKS	MATEMÁTICAS
271	SMITH	HISTORIA

► FIGURA 8-14

Relaciones de intersección de JUNIOR y ESTUDIANTE-HONOR

Enum o Número	Nombre	Especialidad o Interés
123	JONES	HISTORIA

ción de JUNIOR y ESTUDIANTE-HONOR es un tuple independiente [123, JONES, HISTORIA], el cual es el único que aparece en JUNIOR y en ESTUDIANTE-HONOR.

PRODUCTO. El **producto** de dos relaciones (a veces llamado **producto cartesiano**), es la concatenación de cada tuple de una relación con cada tuple de una segunda relación. El producto de relación A (con m tuples) y relación B (con n tuples) tiene m veces n tuples. El producto se denota como $A \times B$, o A VECES B. En la figura 8-15 la relación ESTUDIANTE tiene cuatro tuples y la relación INSCRIPCIÓN tiene tres. Por lo tanto, ESTUDIANTE VECES INSCRITO tiene doce tuples que se muestran en la figura 8-16. (La relación resultante en la figura 8-16 contiene algunos tuples sin importancia. Necesitarían llevarse a cabo otras operaciones que mostraremos más adelante para extraer cualquier información importante de esta relación. Esto es simplemente una ilustración del operador producto.)

PROYECCIÓN. **Proyección** es un operador que selecciona atributos específicos de una relación. El resultado de la proyección es una nueva relación con los atributos seleccionados; en otras palabras, una proyección escoge columnas de una relación. Por ejemplo, considere los datos de la relación ESTUDIANTE en la figura 8-15(a), de los cuales la proyección de ESTUDIANTE en los atributos Nombre y Especialidad, denotados con corchetes como ESTUDIANTE [Nombre, Especialidad], se muestran en la figura 8-17(a). La proyección de ESTUDIANTE en Especialidad y Grado, que se denota como ESTUDIANTE [Especialidad, Grado], aparece en la figura 8-17(b).

Observe que aunque ESTUDIANTE tiene cuatro tuples para empezar, la proyección ESTUDIANTE [Especialidad, Grado] tiene sólo tres. Se eliminó un tuple porque después que la proyección fue terminada, el tuple [HISTORIA, JUNIOR] aparece dos veces. Debido a que el resultado de proyección es una relación, y a que las relaciones no pueden contener tuples duplicados, se elimina el tuple redundante.

La proyección también se puede usar para cambiar el orden de los atributos en una relación. Por ejemplo, la proyección ESTUDIANTE [Grado, Especialidad, Nombre, EID] invierte el orden de atributos de ESTUDIANTE (véase la figura 8-11 para conocer el orden original). Esta característica se puede usar algunas veces para hacer dos relaciones compatibles en la unión.

► FIGURA 8-15

Ejemplo de relaciones:
(a) ESTUDIANTE,
y (b) INSCRIPCIÓN

EID	Nombre	Especialidad	Grado
123	JONES	HISTORIA	JR
158	PARKS	HISTORIA	GR
105	ANDERSON	ADMINISTRACIÓN	SN
271	SMITH	HISTORIA	JR

(a)

Númerode- Estudiante	Nombrede- Clase	Númerode- Posición
123	H350	1
105	BA490	3
123	BA490	7

(b)

► FIGURA 8-16

Producto de las relaciones ESTUDIANTE e INSCRIPCIÓN en la figura 8-15

EID	Nombre	Especialidad	Grado	Númerode- Estudiante	Nombred- deClase	Número- dePosición
123	JONES	HISTORIA	JR	123	H350	1
123	JONES	HISTORIA	JR	105	BA490	3
123	JONES	HISTORIA	JR	123	BA490	7
158	PARKS	MATEMÁTICAS	GR	123	H350	1
158	PARKS	MATEMÁTICAS	GR	105	BA490	3
158	PARKS	MATEMÁTICAS	GR	123	BA490	7
105	ANDERSON	ADMINISTRACIÓN	SN	123	H350	1
105	ANDERSON	ADMINISTRACIÓN	SN	105	BA490	3
105	ANDERSON	ADMINISTRACIÓN	SN	123	BA490	7
271	SMITH	HISTORIA	JR	123	H350	1
271	SMITH	HISTORIA	JR	105	BA490	3
271	SMITH	HISTORIA	JR	123	BA490	7

SELECCIÓN. Así como el operador de proyección toma un subconjunto vertical (columnas) de una relación, el operador **selección** toma un subconjunto horizontal (renglón). Proyección identifica los *atributos* que serán incluidos en la nueva relación, y selección identifica los *tuples* que serán incluidos en la nueva relación. Selección se denota especificando el nombre de la relación, seguido por la palabra llave WHERE, y después por una condición que involucra atributos. La figura 8-18(a) muestra la selección de la relación ESTUDIANTE WHERE Especialidad = 'Matemáticas', y la figura 8-18(b) muestra la selección de ESTUDIANTE WHERE Grado = 'JR'.

JOIN. La operación **join** es una combinación del producto, selección y (posiblemente) operaciones de proyección. La asociación de dos relaciones, es decir, A y B, opera de la siguiente manera: primero forma el producto de A veces B. Después lleva a cabo una selección para eliminar algunos tuples (los criterios de selección se especifican como parte de la unión). Después (optativamente), se eliminan algunos atributos por medio de proyección.

Considere las relaciones ESTUDIANTE e INSCRIPCIÓN que se muestran en la figura 8-15. Suponga que queremos conocer Nombre y Número de Posición de cada estudiante. Para encontrarlos, necesitamos juntar los tuples de ESTUDIANTE acoplando con los de INSCRIPCIÓN basados en el EID. Denotamos a este join como ESTUDIANTE JOIN (EID = NúmerodeEstudiante) INSCRIPCIÓN. El significado de esta expresión es "Juntar UN tuple de ESTUDIANTE con un tuple de INSCRIPCIÓN si el EID de ESTUDIANTE es igual al NúmerodeEstudiante de INSCRIPCIÓN".

Para formar este join, primero encontramos el producto de ESTUDIANTE e INSCRIPCIÓN, una operación que se ha mostrado en la figura 8-16. A continuación SELECCIONamos los tuples a partir del producto donde EID de ESTUDIANTE es igual a NúmerodeEstudiante de INSCRIPCIÓN (sólo hay tres). Esta operación conduce a la re-

► FIGURA 8-17

Proyección de las relaciones ESTUDIANTE:
(a) ESTUDIANTE [Nombre, Especialidad], y (b) ESTUDIANTE [Especialidad, Grado]

Nombre	Especialidad
JONES	HISTORIA
PARKS	MATEMÁTICAS
ANDERSON	ADMINISTRACIÓN
SMITH	HISTORIA

(a)

Especialidad	Grado
HISTORIA	JR
MATEMÁTICAS	GR
ADMINISTRACIÓN	SN

(b)

► FIGURA 8-18

Ejemplos de selección relacional: (a) ESTUDIANTE WHERE Especialidad = 'MATEMÁTICAS' y (b) ESTUDIANTE WHERE Grado = 'JR'

EID	Nombre	Especialidad	Grado
158	PARKS	MATEMÁTICAS	GR

(a)

EID	Nombre	Especialidad	Grado
123	JONES	HISTORIA	JR
271	SMITH	HISTORIA	JR

(b)

lación de la figura 8-19(a). Observe que dos atributos son idénticos: EID y Número de Estudiante. Uno es redundante, así que lo eliminamos (en este caso escogemos Número de Estudiante). El resultado es el join de la figura 8-19(b). El join en la figura 8-19(a) se llama el **equijoin de igualdad**, y el de la figura 8-19(b), **join natural**. A menos que se especifique lo contrario, cuando alguien dice join se refiere al join natural.

Debido a que formar el producto de dos grandes relaciones lleva tiempo, el algoritmo que usa un DBMS para join de dos relaciones es diferente al que se ha descrito aquí. Sin embargo, el resultado será idéntico.

Hacer un join en condiciones distintas a la igualdad también es posible. Por ejemplo, ESTUDIANTE JOIN (EID not = Número de Estudiante) INSCRIPCIÓN, o ESTUDIANTE JOIN (EID < FID) FACULTAD. El último join dará como resultado tuples en los que los números de estudiante son menores que los números de facultad. Este join tendría significado si los Identificadores de Personas estuvieran asignados en orden cronológico. Este join representaría pares de estudiantes y maestros, de tal forma que los estudiantes parecerían tener más tiempo en la institución que el maestro.

Existe una restricción importante en las condiciones de un join: los atributos en la condición deben provenir de un dominio común, de tal forma que ESTUDIANTE JOIN

► FIGURA 8-19

Ejemplos de relaciones de un join entre ESTUDIANTE e INSCRIPCIÓN: (a) equijoin, (b) join natural, y (c) left outer join

EID	Nombre	Especialidad	Grado	Número de Estudiante	Nombre de Clase	Número de Posición
123	JONES	HISTORIA	JR	123	H350	1
123	JONES	HISTORIA	JR	123	BA490	7
105	ANDERSON	ADMINISTRACIÓN	SN	105	BA490	3

(a)

EID	Nombre	Especialidad	Grado	Nombre de Clase	Número de Posición
123	JONES	HISTORIA	JR	H350	1
123	JONES	HISTORIA	JR	BA490	7
105	ANDERSON	ADMINISTRACIÓN	SN	BA490	3

(b)

EID	Nombre	Especialidad	Grado	Número de Estudiante	Nombre de Clase	Número de Posición
123	JONES	HISTORIA	JR	123	H350	1
123	JONES	HISTORIA	JR	123	BA490	7
158	PARKS	MATEMÁTICAS	GR	nulo	nulo	nulo
105	ANDERSON	ADMINISTRACIÓN	SN	105	BA490	3
271	SMITH	HISTORIA	JR	nulo	nulo	nulo

(c)

(Especialidad = CLASE.Nombre) CLASE es ilógico. Aun cuando el valor de Especialidad y CLASE.Nombre son Carac (10), no se desprenden del mismo dominio. Semánticamente, este tipo de join no tiene sentido (por desgracia muchos productos relacionales DBMS permiten este join).

OUTER JOIN. La operación join producirá una relación de los estudiantes y las clases que están tomando. Sin embargo, los estudiantes que no toman ninguna clase serán omitidos del resultado. Si queremos incluir a todos podemos usar un **outer join**. Así, ESTUDIANTE LEFT OUTER JOIN (EID = NúmeroEstudiante) INSCRIPCIÓN incluirá cada renglón de estudiante. El resultado se muestra en la figura 8-19(c). El estudiante Smith quedó incluido aunque no está inscrito en ninguna clase. La palabra clave IZQUIERDA (LEFT) especifica que todos los renglones en la tabla del lado izquierdo de la expresión (ESTUDIANTE) aparecerán en el resultado. ESTUDIANTE RIGHT OUTER JOIN (EID = NúmeroEstudiante), INSCRIPCIÓN especifica que todos los renglones en INSCRIPCIÓN están incluidos en el resultado. Los outer joins son útiles cuando funcionan en una relación en la que la mínima cardinalidad es cero en uno o ambos lados. Cuando puede surgir ambigüedad entre los dos tipos de join, a veces se usa el término INNER JOIN en lugar de JOIN.

EXPRESIÓN DE CONSULTAS EN ÁLGEBRA RELACIONAL

La figura 8-20 resume las operaciones básicas relacionales recién analizadas. El conjunto de operaciones estándar incluye +, −, intersección y producto. Selección escoge tuples específicos (renglones) de una relación, de acuerdo con las condiciones de valores de atributo. Proyección escoge atributos específicos (columnas) de una relación, mediante el significado del nombre de atributo. Por último, join concatena los tuples de dos relaciones de acuerdo con una condición sobre los valores de los atributos.

Ahora veamos cómo se pueden usar los operadores relacionales para expresar consultas usando las relaciones ESTUDIANTE, CLASE e INSCRIPCIÓN de la figura 8-11; la figura 8-21 incluye datos muestra. Nuestro propósito es demostrar el uso de relaciones. Aunque probablemente nunca usará el álgebra relacional en un ambiente comercial, estos ejemplos le ayudarán a entender cómo se pueden manejar las relaciones.

► FIGURA 8-20

Resumen de operaciones de álgebra relacional

Tipo	Formato	Ejemplo
Conjunto de operaciones	+, −, intersección, producto	ESTUDIANTE [Nombre] − JUNIOR [Nombre]
Selección	Relación WHERE condición	CLASE WHERE Nombre = 'A'
Proyección	relación [lista de atributos]	ESTUDIANTE [Nombre, Especialidad]
Join	relación 1 JOIN (condición) relación 2	ESTUDIANTE JOIN (EID = NúmeroEstudiante) INSCRIPCIÓN
Inner Join	Sinónimo de join	
Outer Join	relación 1 LEFT OUTER JOIN (condición) relación 2	ESTUDIANTE LEFT OUTER JOIN (EID = NúmeroEstudiante) INSCRIPCIÓN
	o relación 1 RIGHT OUTER JOIN (condición) relación 2	ESTUDIANTE RIGHT OUTER JOIN (EID = NúmeroEstudiante) INSCRIPCIÓN

► FIGURA 8-21

Datos de ejemplo para las relaciones definidas en la figura 8-11: (a) relación ESTUDIANTE, (b) relación INSCRIPCIÓN, y (c) relación CLASE

EID	Nombre	Especialidad	Grado
100	JONES	HISTORIA	GR
150	PARKS	CONTABILIDAD	SO
200	BAKER	MATEMÁTICAS	GR
250	GLASS	HISTORIA	SN
300	BAKER	CONTABILIDAD	SN
350	RUSSELL	MATEMÁTICAS	JR
400	RYE	CONTABILIDAD	FR
450	JONES	HISTORIA	SN

(a)

Númerode- Estudiante	Nombre- deClase	Númerode- Posición
100	BD445	1
150	BA200	1
200	BD445	2
200	CS250	1
300	CS150	1
400	BA200	2
400	BF410	1
400	CS250	2
450	BA200	3

(b)

Nombre	Tiempo	Aula
BA200	M-F9	SC110
BD445	MWF3	SC213
BF410	MWF8	SC213
CS150	MWF3	EA304
CS250	MWF12	EB210

(c)

1. ¿Cuáles son los nombres de todos los estudiantes?

ESTUDIANTE [Nombre]

Ésta es sólo la proyección del atributo Nombre de la relación ESTUDIANTE y el resultado es

JONES
PARKS
BAKER
GLASS
RUSSELL
RYE

Los nombres duplicados han sido omitidos. Aunque los nombres JONES y BAKER realmente aparecen dos veces en la relación ESTUDIANTE, se omitieron las repeticiones porque el resultado de una proyección es una relación, y las relaciones no pueden tener tuples duplicados.

2. ¿Cuáles son los números de todos los estudiantes inscritos en una clase?

INSCRIPCIÓN [NúmeroEstudiante]

Ésta es similar a la primera consulta, pero la proyección ocurre en la relación INSCRIPCIÓN. El resultado es

100
150
200
300
400
450

Otra vez se omitieron los tuples duplicados.

3. ¿Cuáles son los números de estudiante de todos los que no se han inscrito en una clase?

ESTUDIANTE[EID] - INSCRIPCIÓN [NúmeroEstudiante]

Esta expresión encuentra la diferencia en la proyección de las dos relaciones: ESTUDIANTE [EID] tiene los números de todos los estudiantes, e INSCRIPCIÓN [NúmeroEstudiante] tiene los números de todos que están inscritos en una clase. La diferencia es el número de los que no están inscritos. El resultado es

250
350

4. ¿Cuáles son los números de estudiantes inscritos en la clase 'BD445'?

INSCRIPCIÓN WHERE NombreClase = 'BD445' [NúmeroEstudiante]

Esta expresión selecciona los tuples apropiados y luego los proyecta en el atributo NúmeroEstudiante. El resultado es

250
350

5. ¿Cuáles son los nombres de los estudiantes inscritos en la clase 'BD445'?

ESTUDIANTE JOIN (EID = NúmeroEstudiante) INSCRIPCIÓN WHERE NombreClase = 'BD445' [ESTUDIANTE.Nombre]

Para responder a esta consulta, se necesitan tanto los datos de ESTUDIANTE como los de INSCRIPCIÓN. Específicamente, los nombres de estudiantes deben provenir de ESTUDIANTE, mientras que la condición "inscripción en BD445" se debe verificar en INSCRIPCIÓN. Debido a que ambas relaciones son necesarias se deben juntar. Después de haber juntado ESTUDIANTE e INSCRIPCIÓN, se aplica la selección y posteriormente una proyección de los nombres de estudiante. El resultado es

JONES
BAKER

Como ya lo establecimos, cuando se consideran dos o más relaciones se pueden duplicar los nombres de los atributos. Por lo tanto, para aclarar, el nombre de la relación se puede anteponer al nombre del atributo. Así, en nuestro ejemplo, la proyección es sobre [ESTUDIANTE.Nombre]. En este ejemplo se añadió este prefijo sólo

lo por claridad, ya que los nombres de los atributos son diferentes; pero cuando son idénticos (un join que involucra ESTUDIANTE y CLASE produce dos atributos, ambos llamados Nombre) se requiere el prefijo. Considere la siguiente consulta:

6. ¿Cuáles son los nombres y los horarios de las clases del estudiante 'PARKS'?

Para responder esta pregunta debemos juntar los datos en las tres relaciones. Necesitamos los datos ESTUDIANTE para encontrar el número de PARKS; los datos de INSCRIPCIÓN para saber en qué clases está inscrito, y los datos de CLASE para determinar los horarios de las clases a las que asiste.

ESTUDIANTE WHERE Nombre = 'PARKS' JOIN (EID = NúmerodeEstudiante)
INSCRIPCIÓN JOIN (NombredeClase = Nombre) CLASE [CLASE.Nombre, Horario]

Esta expresión primero selecciona el tuple PARKS y lo junta con los tuples correspondientes de INSCRIPCIÓN. Después el resultado se junta con los tuples correspondiente de CLASE. Por último, se toma la proyección para imprimir clases y horarios. El resultado es

BA200	M-F9
-------	------

Debemos especificar CLASE.Nombre; el hecho de sólo especificar Nombre es ambiguo porque tanto ESTUDIANTE como CLASE tienen un atributo llamado Nombre.

Existen otras formas equivalentes de responder a esta consulta. Una es

JOIN ESTUDIANTE (EID = NúmerodeEstudiante) INSCRIPCIÓN JOIN (NombredeClase = Nombre) CLASE WHERE ESTUDIANTE.Nombre = 'PARKS' [CLASE.Nombre, Horario]

Esta expresión difiere de la primera porque la selección de PARKS no se efectúa hasta después que se realizaron todas las juntas. Suponga que la computadora lleva a cabo las operaciones como se establecen; esta expresión será más lenta que la primera porque se juntan muchos tuples.

Estas diferencias son la mayor desventaja del álgebra relacional. Para el usuario, dos consultas equivalentes podrían tomar el mismo tiempo (y por tanto el mismo costo). Imagine la frustración si una de las formas de consulta costara \$1.17 y la otra \$4356. Para un usuario incauto y nada sofisticado la diferencia en el costo parecerá un capricho.

7. ¿Cuáles son los grados y las aulas de todos los estudiantes, incluyendo a los que no están inscritos en una clase?

Debido a que todos deberán ser incluidos, esta consulta requiere el uso de un OUTER JOIN. La sintaxis es directa:

ESTUDIANTE LEFT OUTER JOIN (EID = NúmerodeEstudiante) JOIN INSCRIPCIÓN (NombredeClase = Nombre) CLASE [Grado, Aula]

El resultado incluye el Grado de Glass y de Russell, quienes no están inscritos en ninguna clase.

GR	SC213
SO	SC110
GR	EB210
SN	Nulo
SN	EA304
JR	Nulo
FR	SC110
FR	SC213
FR	EB210
SN	SC110

► RESUMEN

Cuando se implementa una base de datos relacional hay que llevar a cabo varias tareas. Primero, se debe definir la estructura de la base de datos en el DBMS. Después, ubicar el espacio de archivo, y por último, llenar la base con los datos.

El modelo relacional representa y procesa datos en forma de tablas llamadas relaciones. Las columnas de la tabla se llaman atributos y los renglones tuples. Los valores de los atributos surgen de dominios. Los términos *tabla*, *columna* y *renglón*, y *archivo*, *campo* y *registro* se usan como sinónimos de los términos *relación*, *atributo* y *tuple*, respectivamente.

El uso del término *llave* puede ser confuso porque se emplea de manera diferente en las etapas de diseño e implementación. Durante el diseño, el término significa una llave lógica, la cual es uno o más atributos que únicamente definen un renglón. Durante la implementación, el término significa una llave física, que es una estructura de datos usada para mejorar el desempeño. Una llave lógica puede ser o no una llave física, y una llave física puede ser o no una llave lógica. En este texto usaremos llave para significar una llave lógica, e índice para referirnos a una llave física.

Debido a que estamos usando el modelo relacional para expresar diseños de bases de datos, no es necesario transformar el diseño durante la etapa de implementación. Sólo definimos el diseño relacional en el DBMS. Dos formas de definirlo son: expresarlo en un archivo de texto DDL y usar una herramienta gráfica para la definición de datos. En cualquier caso, las tablas, columnas, índices, restricciones, contraseñas y otros controles están definidos en el DBMS.

Además de establecer la estructura de la base de datos, los programadores deben asignar espacio medio para la base de datos. Con sistemas multiusuario, esta asignación puede ser importante para el desempeño eficaz del DBMS. Por último, se llena la base con datos usando las herramientas que proporcionan los fabricantes del DBMS, los programas que éstos han desarrollado, o ambos.

Las cuatro categorías de lenguajes de manejo de datos relacionales son el álgebra relacional, el cálculo relacional, los lenguajes orientados a la transformación, y el Query-by-Example. El álgebra relacional consta de un grupo de operadores relacionales que se pueden usar para manejar relaciones y así obtener el resultado deseado. El álgebra relacional está basada en procedimientos. Los lenguajes orientados a la transformación ofrecen un medio que no está basado en procedimientos para transformar un conjunto de relaciones en un resultado deseado. SQL es el ejemplo más común.

Existen tres medios para acceder a una base de datos relacional: usar las formas y reportes que proporciona el DBMS; usar un lenguaje de consulta-actualización, del cual el SQL es el más común, y finalmente a través de programas de aplicación.

Las interfaces de los programas de aplicación pueden ser mediante llamadas de función, métodos de objeto o comandos de base de datos de propósito especial que se traducen por medio de un precompilador. La orientación al procesamiento del modelo relacional es una relación a la vez, pero la orientación de la mayoría de los lenguajes de programación es un renglón a la vez. Se deben desarrollar algunos medios para corregir este desajuste.

El álgebra relacional se usa para manejar relaciones con el fin de obtener el resultado deseado. Los operadores son la unión, la diferencia, la intersección, el producto, la proyección, la selección, el join (interno) y el join externo.

► PREGUNTAS DEL GRUPO I

- 8.1 Nombre y describa las tres tareas necesarias para implementar una base de datos relacional.
- 8.2 Defina *relación*, *atributo*, *tuple* y *dominio*.

- 8.3 Explique el uso de los términos *tabla*, *columna*, *renglón*, *archivo*, *campo* y *registro*.
 - 8.4 Explique la diferencia entre un esquema relacional y una relación.
 - 8.5 Defina *llave*, *índice*, *llave lógica* y *llave física*.
 - 8.6 Mencione tres razones para usar índices.
 - 8.7 ¿Bajo qué condiciones es necesario transformar el diseño de datos durante la etapa de implementación?
 - 8.8 Explique el término *lenguaje de definición de datos*. ¿Para qué sirve?
 - 8.9 ¿Cómo puede definirse una estructura de base de datos en forma distinta de la de un archivo de texto?
 - 8.10 ¿Qué aspectos del diseño de base de datos se necesita definir en el DBMS?
 - 8.11 Dé un ejemplo, diferente al de este texto, en el cual la asignación de la base de datos a medios físicos sea importante.
 - 8.12 Describa los extremos mejor y peor para cargar una base de datos.
 - 8.13 Nombre y explique brevemente las cuatro categorías de DML relacional.
 - 8.14 Describa cómo se pueden manejar los datos relacionales por medio de formas.
 - 8.15 Explique el papel que desempeñan los lenguajes de consulta en el manejo de los datos relacionales. ¿Cómo difieren las consultas almacenadas de los programas de aplicación? ¿Por qué se usan?
 - 8.16 Describa los dos estilos de interfaz de programas de aplicación a la base de datos. En su respuesta explique el papel del precompilador.
 - 8.17 Describa el desajuste entre la orientación del SQL y la orientación de la mayoría de los lenguajes de programación. ¿Cómo se corrige ese desajuste?
 - 8.18 ¿Cuál es la diferencia entre el álgebra relacional y el álgebra de la escuela preparatoria?
 - 8.19 ¿Por qué el álgebra relacional es *cerrada*?
 - 8.20 Defina *compatibilidad de unión*. Dé un ejemplo sobre dos relaciones que estén en compatibilidad de unión y dos que estén en incompatibilidad de unión.
- Las preguntas 8.21 a la 8.23 se refieren a las siguientes relaciones:

COMPañÍA (Nombre, NúmerodeEmpleados, Ventas)

FABRICANTES (Nombre, TotaldePersonas, Ingreso)

- 8.21 Exponga un ejemplo sobre la unión de estas dos relaciones.
- 8.22 Proporcione un ejemplo que muestre la diferencia de esas dos relaciones.
- 8.23 Mencione un ejemplo de la intersección de esas dos relaciones.

Las preguntas 8.24 a 8.28 se refieren a las tres relaciones siguientes:

VENDEDOR (Nombre, Salario)

PEDIDO (Número, Nombre del Cliente, Nombre del Vendedor, Cantidad)

CLIENTE (Nombre, Ciudad, Tipo de Industria)

Una prueba de estas relaciones se muestra en la figura 8-22. Use los datos de estas tablas para los siguientes problemas:

- 8.24 Mencione un ejemplo del producto de VENDEDOR y PEDIDO.
- 8.25 Muestre un ejemplo de:

VENDEDOR [Nombre, Salario]

VENDEDOR [Salario]

¿Bajo qué condiciones el VENDEDOR [Salario] tendrá menos renglones que VENDEDOR?

- 8.26 Muestre un ejemplo de una selección en VENDEDOR Nombre, VENDEDOR Salario y en ambas, VENDEDOR Nombre y Salario.

► FIGURA 8-22

*Datos de muestra
para las preguntas
8.24 a 8.28*

Nombre	Salario
Abel	120,000
Baker	42,000
Jones	36,000
Murphy	50,000
Zenith	118,000
Kobad	34,000

VENDEDOR

Número	NombredelCliente	NombredelVendedor	Cantidad
100	Abernathy Construction	Zenith	560
200	Abernathy Construction	Jones	1800
300	Manchester Lumber	Abel	480
400	Amalgamated Housing	Abel	2500
500	Abernathy Construction	Murphy	6000
600	Tri-City Builders	Abel	700
700	Manchester Lumber	Jones	150

PEDIDO

Nombre	Ciudad	TipodelIndustria
Abernathy Construction	Willow	B
Manchester Lumber	Manchester	F
Tri-City Builders	Memphis	B
Amalgamated Housing	Memphis	B

CLIENTE

- 8.27 Muestre un ejemplo de un equijoin y un join natural de VENDEDOR y PEDIDO en el cual Nombre del VENDEDOR sea igual a NombredelVendedor en PEDIDO.
- 8.28 Muestre las expresiones del álgebra relacional para
- Los nombres de todos los vendedores
 - Los nombres de todos los vendedores que tienen un renglón PEDIDO
 - Los nombres de los vendedores que no tienen un renglón PEDIDO
 - Los nombres de los vendedores que tienen un PEDIDO con Abernathy Construction
 - Los salarios de los vendedores que tienen un PEDIDO con Abernathy Construction
 - La ciudad de todos los CLIENTES que tienen un PEDIDO con el vendedor Jones
 - Los nombres de todos los vendedores con los nombres de los clientes que les han hecho pedidos. Incluya a los vendedores que no tienen pedidos

Lenguaje de consulta estructurado

El lenguaje de consulta estructurado, o SQL (Structured Query Language), es el lenguaje de manejo de datos relacionales actual más importante. Ha recibido el respaldo del American National Standards Institute (ANSI) como el lenguaje seleccionado para el manejo de bases de datos relacionales, y es el lenguaje de acceso a datos que usan muchos productos DBMS comerciales, incluyendo DB2, SQL/DS, Oracle, INGRES, SYBASE, SQL Server, dBase para Windows, Paradox, Microsoft Access y muchos otros. Debido a su popularidad, SQL ha sido el lenguaje estándar para el intercambio de información entre computadoras. Puesto que hay una versión SQL que puede funcionar en casi todas las computadoras y sistemas operativos, los sistemas de cómputo pueden intercambiar datos, consultas y respuestas.

El desarrollo del SQL empezó en las instalaciones de investigación de IBM San José, a mediados de la década de 1970 con el nombre de SEQUEL. Salieron varias versiones de SEQUEL y en 1980 el producto se denominó SQL. Desde entonces, IBM se ha unido a muchos otros proveedores en el desarrollo de productos SQL. El American National Standards Institute se ha dado a la tarea de mantener SQL y periódicamente publica versiones actualizadas de la norma SQL. Este capítulo analiza el núcleo de SQL como se describe en la norma ANSI de 1992, que con frecuencia se conoce como SQL92¹. La versión más reciente, SQL3, se refiere a las extensiones del lenguaje de programación orientada a objetos. Esa versión es analizada en el capítulo 18.

Las construcciones y expresiones en una implementación particular de SQL (por ejemplo, en Oracle o en el SQL Server) pueden diferir un poco de la norma ANSI, en parte porque muchos de los productos DBMS se desarrollaron antes de que hubiera un acuerdo sobre la norma, y también debido a que los proveedores agregaron capacidades a sus productos para que fueran más competitivos. Desde una perspectiva de mercadotecnia, a la norma ANSI a veces se le juzgó poco interesante.

Las órdenes de SQL se pueden usar interactivamente como un lenguaje de consulta, o se pueden insertar en programas de aplicación. De esta manera, SQL no es un lenguaje de programación (como COBOL), sino un *sublenguaje de datos*, o un *lenguaje de acceso de datos* que se implanta en otros lenguajes.

¹ Revista de la Organización Internacional de Normas ISO/IEC 9075: 1992, *Database Language SQL*.

En este capítulo presentamos enunciados interactivos SQL que necesitan ser ajustados y modificados cuando se implantan en programas, como se muestra en los capítulos 12 y 13. El presente capítulo se refiere sólo a los enunciados para el manejo de datos; en los capítulos 12 y 13 se analizan los enunciados de definición de datos.

SQL es un lenguaje orientado a la transformación que acepta como entrada una o más relaciones y produce una sola relación de salida. El resultado de cada consulta SQL es una relación; incluso si el resultado es un número independiente, ese número se considera como una relación con un solo renglón y una sola columna. Por lo tanto, SQL es como el álgebra relacional: *cerrado*.

► CONSULTA DE UNA SOLA TABLA

En esta sección consideramos las facilidades SQL para consultar una tabla independiente. Más adelante analizaremos tablas múltiples y enunciados de actualización. Por costumbre, las palabras reservadas de SQL tales como SELECT y FROM están escritas con letras mayúsculas. También, los enunciados SQL por lo general se escriben en líneas múltiples, como se muestra en este capítulo. Sin embargo, los compiladores de lenguaje SQL no requieren ni letras mayúsculas ni líneas múltiples. Estas convenciones se usan sólo para proporcionar una claridad especial a quienes leen los enunciados SQL.

Usamos el mismo conjunto de seis relaciones con las que ilustramos el álgebra relacional en el capítulo 8. La estructura de estas relaciones se ejemplifica en la figura 9-1 y los datos de muestra para tres de ellas aparecen en la figura 9-2.

► FIGURA 9-1

Relaciones usadas para los ejemplos SQL

- 1. JUNIOR (Enum, Nombre, Especialidad)
- 2. ESTUDIANTE-HONOR (Número, Nombre, Interés)
- 3. ESTUDIANTE (EID, Nombre, Especialidad, Grado)
- 4. CLASE (Nombre, Horario, Aula)
- 5. INSCRIPCIÓN (NúmerodeEstudiante, NombredeClase, NúmerodePosición)
- 6. FACULTAD (FID, Nombre, Departamento)

Atributo	Dominio
1. Enum	IdentificadoresdePersonas
Nombre.JUNIOR	NombresdePersonas
Especialidad	NombresdeTemas
2. Número	IdentificadoresdePersonas
Nombre.ESTUDIANTE-HONOR	NombresdePersonas
Interés	NombresdeTemas
3. EID	IdentificadoresdePersonas
ESTUDIANTE.Nombre	NombresdePersonas
Especialidad	NombresdeTemas
Grado	Clases
4. CLASE.Nombre	NombresdeClases
Tiempo	HorariodeClases
Aula	Aulas
5. NúmerodeEstudiante	IdentificadoresdePersonas
NombredeClase	NombresdeClases
NúmerodePosición	TamañosdeClases
6. FID	IdentificadoresdePersonas
FACULTAD.Nombre	NombresdePersonas
Departamento	NombresdeTemas

► FIGURA 9-2

Datos de muestra
usados en los
ejemplos SQL:
(a) relación ESTU-
DIANTE, (b) relación
INSCRIPCIÓN, y
(c) relación CLASE

EID	Nombre	Especialidad	Grado
100	JONES	HISTORIA	GR
150	PARKS	CONTABILIDAD	SO
200	BAKER	MATEMÁTICAS	GR
250	GLASS	HISTORIA	SN
300	BAKER	CONTABILIDAD	SN
350	RUSSELL	MATEMÁTICAS	JR
400	RYE	CONTABILIDAD	FR
450	JONES	HISTORIA	SN

(a)

Número de Estudiante	Nombre de Clase	Número de Posición
100	BD445	1
150	BA200	1
200	BD445	2
200	CS250	1
300	CS150	1
400	BA200	2
400	BF410	1
400	CS250	2
450	BA200	3

(b)

Nombre	Horario	Aula
BA200	M-F9	SC110
BD445	MWF3	SC213
BF410	MWF8	SC213
CS150	MWF3	EA304
CS250	MWF12	EB210

(c)

PROYECCIONES UTILIZANDO SQL

Para formar una proyección con SQL nombramos la relación a proyectarse y listamos las columnas que van a ser mostradas. Utilizando la sintaxis estándar SQL, la proyección ESTUDIANTE [EID, Nombre, Especialidad] se especifica como

```
SELECT      EID, Nombre, Especialidad
FROM        ESTUDIANTE
```

Las palabras reservadas SELECT y FROM siempre se requieren; las columnas a obtener se listan después de la palabra reservada SELECT, y la tabla que se va a usar se enumera después de la palabra reservada FROM. El resultado de esta proyección para los datos de la figura 9-2 es

100	JONES	HISTORIA
150	PARKS	CONTABILIDAD
200	BAKER	MATEMÁTICAS
250	GLASS	HISTORIA
300	BAKER	CONTABILIDAD
350	RUSSELL	MATEMÁTICAS
400	RYE	CONTABILIDAD
450	JONES	HISTORIA

No confunda la palabra reservada SELECT con el operador de selección del álgebra relacional. SELECT es un verbo SQL que se puede usar para realizar una proyección de álgebra relacional, seleccionar, y especificar otras acciones. Por otra parte, Selección difiere de SELECT porque es la operación de álgebra relacional para obtener un subconjunto de renglones de una tabla.

Considere otro ejemplo:

```
SELECT    Especialidad
FROM      ESTUDIANTE
```

El resultado de esta operación es el siguiente:

HISTORIA
CONTABILIDAD
MATEMÁTICAS
HISTORIA
CONTABILIDAD
MATEMÁTICAS
CONTABILIDAD
HISTORIA

Como puede ver, esta tabla contiene renglones duplicados y, en consecuencia, en un sentido estricto esta tabla no es una relación. De hecho, SQL no elimina automáticamente los duplicados porque puede tomar mucho tiempo y en muchos casos no es deseable ni necesario.

Si se deben eliminar los renglones duplicados el calificador DISTINCT debe ser especificado así:

```
SELECT    DISTINCT Especialidad
FROM      ESTUDIANTE
```

El resultado de esta operación es la relación:

HISTORIA
CONTABILIDAD
MATEMÁTICAS

SELECCIONES USANDO SQL

El operador de selección de álgebra relacional también se lleva a cabo con la orden SQL SELECT. Un ejemplo de esto es:

```
SELECT    EID, Nombre, Especialidad, Grado
FROM      ESTUDIANTE
WHERE     Especialidad = 'MATEMÁTICAS'
```

Esta expresión SELECT especifica los nombres de todas las columnas de las tablas. FROM especifica la tabla que hay que usar, y la nueva frase, WHERE, proporciona la(s) condición(es) para la selección. El formato SELECT-FROM-WHERE es la estructura fundamental de los enunciados SQL. La siguiente es una forma equivalente de la consulta anterior:

```
SELECT    *
FROM      ESTUDIANTE
WHERE     Especialidad = 'MATEMÁTICAS'
```

El asterisco (*) significa que deben obtenerse todas las columnas de la tabla. El resultado de ambas consultas es:

200	BAKER	MATEMÁTICAS	GR
350	RUSSELL	MATEMÁTICAS	JR

Podemos combinar la selección y la proyección como sigue:

```
SELECT    Nombre, Grado
FROM      ESTUDIANTE
WHERE     Especialidad = 'MATEMÁTICAS'
```

El resultado es:

BAKER	GR
RUSSELL	JR

Se pueden expresar varias condiciones en la cláusula WHERE. Por ejemplo, con la expresión:

```
SELECT    Nombre, Grado
FROM      ESTUDIANTE
WHERE     Especialidad = 'MATEMÁTICAS' y Grado = 'GR'
```

se obtiene lo siguiente:

BAKER	GR
-------	----

Las condiciones en las cláusulas WHERE se pueden referir a un conjunto de valores. Para hacer esto, se pueden usar las palabras reservadas IN o NOT IN. Considere:

```
SELECT    Nombre
FROM      ESTUDIANTE
WHERE     Especialidad IN ['MATEMÁTICAS', 'CONTABILIDAD']
```

Observe que dentro de los corchetes se pueden colocar valores múltiples. Esta expresión significa: “Desplegar los nombres de los estudiantes que tienen una especialidad, ya sea en matemáticas o en contabilidad”. El resultado es:



PARKS
BAKER
BAKER
RUSSELL
RYE

La expresión:

```
SELECT    Nombre
FROM      ESTUDIANTE
WHERE     Especialidad NOT IN ['MATEMÁTICAS', 'CONTABILIDAD']
```

hace que aparezcan los nombres de estudiantes que no tienen especialidad en matemáticas o contabilidad. El resultado es:

JONES
GLASS
JONES

La expresión ESPECIALIDAD IN significa que el valor de la columna Especialidad puede ser igual a *cualquiera* de las especialidades listadas. Esto es equivalente al operador lógico OR. La expresión ESPECIALIDAD NOT IN significa que el valor debe ser diferente a *todas* las especialidades listadas.

Las cláusulas WHERE también se pueden referir a rangos y valores parciales. La palabra reservada BETWEEN se usa para los rangos. Por ejemplo, con el enunciado:

```
SELECT    Nombre, Especialidad
FROM      ESTUDIANTE
WHERE     EID BETWEEN 200 AND 300
```

se obtendrá el siguiente resultado:

BAKER	MATEMÁTICAS
GLASS	HISTORIA
BAKER	CONTABILIDAD

Esta expresión equivale a:

```
SELECT    Nombre, Especialidad
FROM      ESTUDIANTE
WHERE     EID >= 200 AND EID <= 300
```

Por lo tanto, los valores finales de BETWEEN (200 y 300 en este caso) están incluidos en el rango seleccionado.

La palabra reservada LIKE se usa en expresiones SQL para seleccionar valores parciales. El símbolo_ (guión bajo) representa un carácter independiente no especificado; el símbolo % representa una serie de uno o más caracteres no especificados. Así, el resultado de la expresión:

```
SELECT    Nombre, Grado
FROM      ESTUDIANTE
WHERE     Grado LIKE '_R'
```

es una relación que tiene las columnas Nombre y Grado, donde Grado consta de dos caracteres, el segundo de los cuales es R:

JONES	GR
BAKER	GR
RUSSELL	JR
RYE	FR

De manera similar, la siguiente expresión encontrará a los estudiantes cuyos apellidos terminen con S:

```
SELECT    Nombre
FROM      ESTUDIANTE
WHERE     Nombre LIKE '%S'
```

el resultado es:

JONES
PARKS
GLASS
JONES

Cabe señalar que Microsoft Access usa un conjunto de símbolos comodines diferentes a la norma ANSI. Un “?” se usa en lugar del guión bajo, y un “*” se emplea en lugar de “%”.

Por último, las palabras reservadas IS NULL se usan para buscar valores nulos (o faltantes). Con la expresión:

```
SELECT    Nombre
FROM      ESTUDIANTE
WHERE     Grado IS NULL
```

se obtendrán los nombres de los estudiantes que no tienen un valor registrado de Grado. Para los datos de la figura 9-2 todos los estudiantes tienen un Grado y esta expresión regresará una relación sin renglones.

ORDENAMIENTO

Los renglones del resultado de la relación se pueden ordenar por los valores de una o más columnas. Considere el siguiente ejemplo:

```
SELECT    Nombre, Especialidad, Grado
FROM      ESTUDIANTE
WHERE     Especialidad = 'CONTABILIDAD'
ORDER BY  Nombre
```

Esta consulta listará las especialidades de contabilidad en secuencia ascendente por el valor del nombre. El resultado es:

BAKER	CONTABILIDAD	SN
PARKS	CONTABILIDAD	SO
RYE	CONTABILIDAD	FR

Se puede elegir más de una columna para el ordenamiento. Si es así, la primera columna listada será el campo ordenado de la especialidad; la segunda, el siguiente campo ordenado de la especialidad, y así sucesivamente. Las columnas también pueden declararse como ascendentes (ASC) o descendentes (DESC), como se muestra en el siguiente enunciado:

```
SELECT    Nombre, Especialidad, Grado
FROM      ESTUDIANTE
WHERE     Grado IN ['FR', 'SO', 'SN']
ORDER BY  Especialidad ASC, Grado DESC
```

El resultado es:

PARKS	CONTABILIDAD	SO
BAKER	CONTABILIDAD	SN
RYE	CONTABILIDAD	FR
GLASS	HISTORIA	SN
JONES	HISTORIA	SN

ORDER BY se puede combinar con cualquiera de los enunciados SELECT.

FUNCIONES SQL INTERCONSTRUIDAS

SQL proporciona cinco funciones interconstruidas: COUNT (contar), SUM (sumar), AVG (promediar), MAX (obtener el valor máximo) y MIN² (obtener el valor mínimo). Aunque COUNT y SUM suenan similares, son realmente diferentes. COUNT calcula el número de renglones en una tabla, mientras que SUM totaliza las columnas numéricas, AVG, MAX y MIN también operan en columnas numéricas: AVG calcula el valor promedio, y MAX y MIN obtienen los valores máximos y mínimos de una columna en una tabla.

La expresión de consulta:

```
SELECT    COUNT(*)
FROM      ESTUDIANTE
```

cuenta el número de renglones ESTUDIANTE y despliega ese total en una tabla con un renglón independiente y una sola columna:

8

Considere las expresiones:

```
SELECT    COUNT(Especialidad)
FROM      ESTUDIANTE
```

y

```
SELECT    COUNT (DISTINCT Especialidad)
FROM      ESTUDIANTE
```

La primera expresión cuenta todas las especialidades, incluyendo duplicados, y la segunda cuenta sólo las especialidades únicas. Los resultados son:

8

y

3

respectivamente.

Con excepción de GROUP BY (que analizaremos a continuación), las funciones interconstruidas no se pueden mezclar con los nombres de las columnas en el enunciado SELECT. Así,

```
SELECT    Nombre, COUNT (*)
```

no está permitido.

Las funciones interconstruidas se pueden usar para solicitar un resultado, como en los ejemplos anteriores. En la mayoría de las implementaciones de SQL y en la norma ANSI de SQL, las funciones interconstruidas *no se pueden* usar como parte de una cláusula WHERE.

FUNCIONES INTERCONSTRUIDAS Y DE AGRUPAMIENTO

Para aumentar su utilidad, se pueden aplicar funciones interconstruidas a grupos de renglones dentro de una tabla. Estos grupos se forman uniando esos renglones (en forma lógica, no física) que tengan el mismo valor de una columna específica. Por ejemplo, se

²A veces las funciones construidas se conocen como funciones agregadas para distinguirlas de las funciones interconstruidas de lenguajes de programación tales como SUBSTRING.

pueden agrupar los estudiantes por especialidad, lo que significa que un grupo estará formado por cada valor de ESPECIALIDAD. Para los datos de la figura 9-2, hay un grupo de estudiantes de HISTORIA, otro de CONTABILIDAD, y otro más de MATEMÁTICAS.

La palabra reservada de SQL: GROUP BY, instruye al DBMS a que agrupe esos renglones que tengan el mismo valor de una columna. Considere:

```
SELECT    Especialidad, COUNT (*)
FROM      ESTUDIANTE
GROUP BY  Especialidad
```

El resultado de esta expresión es:

HISTORIA	3
CONTABILIDAD	3
MATEMÁTICAS	2

Los renglones de la tabla ESTUDIANTE se han agrupado en forma lógica por el valor de ESPECIALIDAD, y la función COUNT suma el número de renglones en cada grupo. El resultado es una tabla con dos columnas, el nombre de la especialidad y la suma. Para los subgrupos se pueden especificar las columnas y las funciones interconstruidas en el enunciado SELECT.

En algunos casos no queremos considerar todos los grupos. Por ejemplo, podríamos formar grupos de estudiantes con la misma especialidad y considerar sólo aquellos grupos que tengan más de dos estudiantes. En este caso, usaríamos la cláusula SQL HAVING para identificar el subconjunto de grupos que deseamos considerar.

Los siguientes enunciados SQL pueden listar las especialidades que tengan más de dos estudiantes y también la cuenta de estudiantes en cada una de las especialidades.

```
SELECT    Especialidad, COUNT (*)
FROM      ESTUDIANTE
GROUP BY  Especialidad
HAVING    COUNT (*) > 2
```

Aquí se forman grupos de estudiantes con la misma especialidad y a continuación se seleccionan los grupos que tienen más de dos estudiantes (los demás grupos se ignoran). La especialidad y la cuenta de estudiantes se produce en estos grupos seleccionados. El resultado es:

HISTORIA	3
CONTABILIDAD	3

Para una generalidad aun mayor se pueden agregar las cláusulas WHERE. Sin embargo, al hacerlo se pueden generar ambigüedades. Por ejemplo,

```
SELECT    Especialidad, MAX (EID)
FROM      ESTUDIANTE
WHERE     Grado = 'SN'
GROUP BY  Especialidad
HAVING    COUNT (*) > 1
```

El resultado de esta expresión diferirá dependiendo de si la condición WHERE se aplica o no antes o después de la condición HAVING. Para eliminar esta incertidumbre, la norma SQL especifica que las cláusulas WHERE deberán ser aplicadas primero. En concordancia, en los enunciados anteriores las operaciones son: seleccionar a los estudiantes graduados; formar los grupos; seleccionar los grupos que cumplan la condición HAVING; desplegar los resultados. En este caso, el resultado es:

HISTORIA	450
----------	-----

(No hay que perder de vista que esta consulta no es válida para todas las implementaciones SQL. Para algunas, los únicos atributos que pueden aparecer en la frase SELECT de una consulta con GROUP BY son los atributos que pueden aparecer en la frase GROUP BY y las funciones interconstruidas de esos atributos. Así, en esta consulta sólo serían permitidas ESPECIALIDAD y las funciones interconstruidas de ESPECIALIDAD.)

► CONSULTAS DE TABLAS MÚLTIPLES

En esta sección ampliamos nuestro análisis de SQL para incluir operaciones en dos o más tablas. Los datos de la figura 9-2 ESTUDIANTE, CLASE, e INSCRIPCIÓN se utilizan para ilustrar estos comandos SQL.

RECUPERACIÓN USANDO UNA SUBCONSULTA

Suponga que necesitamos saber los nombres de aquellos estudiantes inscritos en la clase BD445. Si sabemos que los estudiantes con EID de 100 y 200 están inscritos en esa clase, lo siguiente producirá los nombres correctos:

```
SELECT    Nombre
FROM      ESTUDIANTE
WHERE     EID IN [100, 200]
```

En general no conocemos los EID de los estudiantes en una clase, pero tenemos los medios para encontrarlos. Examine la expresión:

```
SELECT    NúmerodeEstudiante
FROM      INSCRIPCIÓN
WHERE     NombredeClase = 'BD445'
```

El resultado de esta operación es:

100
200

Éstos son los números de estudiantes que necesitamos. Combinando las últimas dos consultas, obtenemos lo siguiente:

```
SELECT    Nombre
FROM      ESTUDIANTE
WHERE     EID IN
          (SELECT    NúmerodeEstudiante
           FROM      INSCRIPCIÓN
           WHERE     NombredeClase = 'BD445')
```

El segundo SELECT, denominado **subconsulta**, está entre paréntesis.

Puede ser más fácil comprender estos enunciados si trabaja desde la parte inferior y lee hacia arriba. Los tres últimos enunciados producen los nombres de los dos estudiantes inscritos en BD445, y los primeros tres enunciados arrojan los nombres de los dos estudiantes seleccionados. El resultado de esta consulta es:

JONES
BAKER

Para que esta operación sea semánticamente correcta, EID y NúmerodeEstudiante deben provenir del mismo dominio.

Las subconsultas pueden constar de tres o más tablas. Por ejemplo, supongamos que queremos saber los nombres de los estudiantes inscritos en la clase de los lunes, miércoles y viernes a las 3:00 p.m. (denotado como LMV3 en nuestros datos). Primero, necesitamos los nombres de las clases que cumplen este horario:

```
SELECT    CLASE.Nombre
FROM      CLASE
WHERE     Horario = 'LMV3'
```

(Puesto que estamos tratando con tres tablas diferentes, calificamos los nombres de las columnas con los nombres de las tablas para evitar confusión y ambigüedad. Así, CLASE.Nombre se refiere a la columna Nombre en la relación CLASE.)

Ahora obtenemos los números de identificación de los estudiantes en estas clases usando la siguiente expresión:

```
SELECT    INSCRIPCIÓN.NúmeroEstudiante
FROM      INSCRIPCIÓN
WHERE     INSCRIPCIÓN.ClaseNombre IN
          (SELECT    CLASE.Nombre
           FROM      CLASE
           WHERE     HORARIO = 'LMV3')
```

Esto da como resultado:

100
200
300

los cuales son los números de los estudiantes en la clase LMV3. Para obtener los nombres de aquellos estudiantes, especificamos:

```
SELECT    ESTUDIANTE.Nombre
FROM      ESTUDIANTE
WHERE     ESTUDIANTE.EID IN
          (SELECT INSCRIPCIÓN.NúmeroEstudiante
           FROM INSCRIPCIÓN
           WHERE INSCRIPCIÓN.NombredeClase IN
              (SELECT CLASE.Nombre IN
               FROM    CLASE
               WHERE   CLASE.Horario = 'LMV3'))
```

El resultado es:

JONES
BAKER
BAKER

Esta estrategia funciona bien siempre que los atributos en la respuesta provengan de una sola tabla. Sin embargo, si el resultado proviene de dos o más tablas tendremos un problema. Por ejemplo, suponga que queremos saber los nombres de los estudiantes y los de sus clases. Digamos que necesitamos EID, NombredeEstudiante, y Nom-

bredeClase. En este caso, los resultados provienen de dos tablas diferentes (ESTUDIANTE e INSCRIPCIÓN) y así la estrategia de la subconsulta no funcionará.

JOIN CON SQL

Para producir el EID, Nombre y NombredeClase para todos los estudiantes, debemos hacer el join de la tabla ESTUDIANTE con la tabla INSCRIPCIÓN. Los siguientes enunciados harán esto:

```
SELECT      ESTUDIANTE.EID, ESTUDIANTE.Nombre, INSCRIPCIÓN.NombredeClase
FROM        ESTUDIANTE, INSCRIPCIÓN
WHERE       ESTUDIANTE.EID = INSCRIPCIÓN.NúmerodeEstudiante
```

Recuerde que un join es la combinación de la operación de un producto seguido por una selección, y (generalmente) por una proyección. Así, el enunciado FROM expresa el producto de ESTUDIANTE e INSCRIPCIÓN, y el enunciado WHERE, la selección. El significado es: “Seleccionar del producto ESTUDIANTE e INSCRIPCIÓN aquellos renglones en las que el EID de ESTUDIANTE sea igual a NúmerodeEstudiante en INSCRIPCIÓN”. Por último, después de seleccionar, se toma la proyección del número de estudiante, su nombre y el de la clase. El resultado es:

100	JONES	BD445
150	PARKS	BA200
200	BAKER	BD445
200	BAKER	CS250
300	BAKER	CS125
400	RYE	BA200
400	RYE	BF410
400	RYE	CS250
450	JONES	BA200

La cláusula WHERE puede contener calificadores además de los que se necesitan para la junta. Por ejemplo,

```
SELECT      ESTUDIANTE.EID INSCRIPCIÓN.NombredeClase
FROM        ESTUDIANTE, INSCRIPCIÓN
WHERE       ESTUDIANTE.EID = INSCRIPCIÓN.NúmerodeEstudiante
AND         ESTUDIANTE.Nombre = 'RYE'
AND         INSCRIPCIÓN.NúmerodePosición = 1
```

Los calificadores adicionales aquí son ESTUDIANTE.Nombre = 'RYE' e INSCRIPCIÓN.NúmerodePosición = 1. Esta operación listará el número de estudiante y el nombre de la clase de todos los estudiantes llamados RYE que se inscribieron primero en la clase. El resultado es:

400	BF410
-----	-------

Cuando se necesiten datos de más de dos tablas, podemos usar una estrategia similar. En el siguiente ejemplo se juntan tres tablas:

```
SELECT      ESTUDIANTE.EID, CLASE.Nombre, CLASE.Horario,
            INSCRIPCIÓN.NúmerodePosición
FROM        ESTUDIANTE, INSCRIPCIÓN, CLASE
WHERE       ESTUDIANTE.EID = INSCRIPCIÓN.NúmerodeEstudiante
AND         INSCRIPCIÓN.NombredeClase = CLASE.Nombre
AND         ESTUDIANTE.Nombre = 'BAKER'
```

El resultado de esta operación es:

200	BD445	LMV3	2
200	CS250	LMV12	1
300	CS150	LMV3	1

COMPARACIÓN DEL JOIN Y DE LA SUBCONSULTA SQL

Se puede usar un join como forma alternativa de expresión para muchas subconsultas. Por ejemplo, usamos una subconsulta para encontrar a los estudiantes inscritos en la clase BD445. También podemos usar un join para expresar esta consulta:

```
SELECT    ESTUDIANTE.Nombre
FROM      ESTUDIANTE, INSCRIPCIÓN
WHERE     ESTUDIANTE.EID = INSCRIPCIÓN.NúmerodeEstudiante
AND       INSCRIPCIÓN.NombredeClase = 'BD445'
```

De manera similar, la consulta: “¿Cuáles son los nombres de los estudiantes en las clases LMV a las 3?” se puede expresar como:

```
SELECT    ESTUDIANTE.NOMBRE
FROM      ESTUDIANTE, INSCRIPCIÓN, CLASE
WHERE     ESTUDIANTE.EID = INSCRIPCIÓN.NúmerodeEstudiante
AND       INSCRIPCIÓN.NombredeClase = CLASE.Nombre
AND       CLASE.Horario = 'LMV3'
```

Aunque las expresiones del join pueden sustituir a muchas expresiones de subconsultas, no pueden reemplazarlas a todas. Por ejemplo, las subconsultas que implican EXISTS y NOT EXISTS (las cuales analizaremos en la siguiente sección) no se pueden representar por medio de un join.

De igual forma, las subconsultas no se pueden sustituir por todos los join. Cuando usamos una unión, las columnas desplegadas pueden provenir de cualquiera de las tablas juntadas, pero cuando usamos una subconsulta las columnas desplegadas pueden provenir sólo de la tabla indicada en la expresión FROM en el primer SELECT. Por ejemplo, supongamos que queremos saber los nombres de las clases que toman los no graduados. Podemos expresar esto como una subconsulta:

```
SELECT    DISTINCT NombredeClase
FROM      INSCRIPCIÓN
WHERE     NúmerodeEstudiante IN
          (SELECT EID
           FROM ESTUDIANTE
           WHERE Grado NOT = 'GR')
```

o como un join:

```
SELECT    DISTINCT INSCRIPCIÓN.NombredeClase
FROM      INSCRIPCIÓN, ESTUDIANTE
WHERE     INSCRIPCIÓN.NúmerodeEstudiante = ESTUDIANTE.EID
AND       ESTUDIANTE.Grado NOT = 'GR'
```

Pero si queremos conocer los nombres de las clases y los niveles de grado de los estudiantes que aún no terminan su posgrado, debemos usar un join. Una subconsulta no será suficiente porque los resultados deseados provienen de dos tablas diferentes. Esto es, los nombres de las clases en INSCRIPCIÓN y los nombres de los estudiantes se almacenan en ESTUDIANTE. Lo siguiente arroja la respuesta correcta:


```

SELECT      DISTINCT INSCRIPCIÓN.NombredeClase, ESTUDIANTE.Grado
FROM        INSCRIPCIÓN, ESTUDIANTE
WHERE       INSCRIPCIÓN.NúmerodeEstudiante = ESTUDIANTE.EID
AND         ESTUDIANTE.Grado NOT = 'GR'

```

El resultado es:

BA200	SO
CS150	SN
BA200	FR
BF410	FR
CS250	FR
BA200	SN

OUTER JOIN

La norma SQL de la ANSI no avala los outer join. Sin embargo, muchos productos DBMS los manejan. Aquí ilustraremos el uso de uno de ellos.

Suponga que queremos una lista de todos los estudiantes y los nombres de las clases que toman, y que además deseamos incluir a todos los estudiantes, incluso aquellos que no están tomando clases. La siguiente expresión SQL dará este resultado utilizando Microsoft Access:

```

SELECT      Nombre, NombredeClase
FROM        ESTUDIANTE LEFT JOIN INSCRIPCIÓN
           ON EID = NúmerodeEstudiante;

```

El resultado es:

JONES	BD445
PARKS	BA200
BAKER	BD445
BAKER	CS250
GLASS	Nulo
BAKER	CS150
RUSSELL	Nulo
RYE	BA200
RYE	BF410
RYE	CS250
JONES	BA200

Observe las diferencias en Access SQL y la norma de notación ANSI. Las condiciones de la junta se especifican usando la palabra llave ON. También, todas las expresiones SQL terminan con punto y coma.

► EXISTS Y NOT EXISTS

EXISTS y NOT EXISTS son operadores lógicos cuyo valor puede ser verdadero o falso, dependiendo de la presencia o ausencia de renglones que cumplan las condiciones calificativas. Por ejemplo, suponga que deseamos saber los números de estudiantes inscritos en más de una clase.

```

SELECT      DISTINCT NúmerodeEstudiante
FROM        INSCRIPCIÓN A
WHERE       EXISTS
            (SELECT      *
             FROM        INSCRIPCIÓN B
             WHERE       A.NúmerodeEstudiante = B.NúmerodeEstudiante
             AND         A.NombredeClase NOT = B.NombredeClase)

```

En este ejemplo, tanto la consulta como la subconsulta se refieren a la tabla INSCRIPCIÓN. Para evitar la ambigüedad, a estos dos usos de INSCRIPCIÓN se les ha asignado un nombre diferente. En el primer enunciado FROM, a INSCRIPCIÓN se le ha asignado, temporal y arbitrariamente, el nombre A, y en el segundo enunciado FROM, el nombre B.

El significado de la expresión subconsulta es: Encontrar dos renglones en INSCRIPCIÓN que tengan el mismo número de estudiante, pero diferentes nombres de clases (lo cual significa que el estudiante está tomando más de una clase). Si existen dos de estos renglones, entonces el valor lógico de EXISTS es verdadero. En este caso, presentamos el número de estudiante en la respuesta. De otra manera, el valor lógico de EXISTS es falso, así que no presentamos ese EID en la respuesta.

Otra forma de ver esta consulta es imaginar dos copias separadas e idénticas de la tabla INSCRIPCIÓN. Nombre a una copia Tabla A y a la otra Tabla B. Compare cada renglón de la tabla A con cada renglón de la B. Primero busque en el primer renglón en A y en el primer renglón de B. En este caso, puesto que ambos son idénticos, tanto NúmerodeEstudiantes como NombresdeClases son iguales, así que no desplegamos el EID.

Ahora observe el primer renglón en A y el segundo renglón en B. Si NúmerodeEstudiantes son los mismos y NombresdeClases son diferentes, desplegamos NúmerodeEstudiante. Esencialmente, estamos comparando el primer renglón de INSCRIPCIÓN con el segundo renglón de INSCRIPCIÓN. Para los datos de la figura 9-2, ni NúmerodeEstudiantes ni NombresdeClases son iguales.

Continuamos comparando el primer renglón de A con cada renglón de B. Si las condiciones se cumplen imprimimos el NúmerodeEstudiante. Cuando se hayan examinado todos los renglones en B, nos movemos al segundo renglón de A y lo comparamos con todos los renglones en B (realmente, si estamos considerando los n renglones en A, sólo se necesita considerar en B los renglones mayores que n).

El resultado de esta consulta es:

200
400

Para ilustrar la aplicación NOT EXISTS suponga que queremos conocer los nombres de los estudiantes que asisten a todas las clases. Otra forma de establecer lo anterior es que queremos los nombres de los estudiantes donde no haya clases que éstos no tomen. Lo siguiente expresa esto:

```

SELECT      ESTUDIANTE.Nombre
FROM        ESTUDIANTE
WHERE       NOT EXISTS
            (SELECT      *
             FROM        INSCRIPCIÓN
             WHERE       NOT EXISTS
                     (SELECT      *
                      FROM        CLASE
                      WHERE       CLASE.Nombre = INSCRIPCIÓN.NombredeClase
                      AND         INSCRIPCIÓN.NúmerodeEstudiante = ESTUDIANTE.EID))

```

Esta consulta tiene tres partes. En la inferior, se encuentran las clases que el estudiante ha tomado. La parte media determina si se encontraron algunas clases que el estudiante no haya tomado. De lo contrario, significa que el estudiante está tomando todas las clases y su nombre será desplegado.

Esta consulta puede ser difícil de comprender. Si tiene problemas use los datos de la figura 9-2 y siga las instrucciones. Para estos datos la respuesta es que ningún estudiante está tomando todas sus clases. Puede tratar de cambiar los datos, de tal forma que un estudiante tome todas las clases. Otra forma de analizar esta consulta es tratar de resolverla de otra manera usando NOT EXISTS. Los problemas que encuentre le ayudarán a comprender por qué NOT EXISTS es necesario.

► CAMBIO DE DATOS

SQL tiene medios para cambiar datos en tablas insertando renglones nuevos, eliminando renglones, y modificando los valores de los renglones existentes. SQL también puede cambiar la estructura de los datos, pero esto lo abordaremos hasta los capítulos 12 y 13.

INSERCIÓN DE DATOS

En una tabla se pueden insertar renglones uno por uno o en grupos. Para insertar uno sólo establecemos:

```
INSERT      INTO INSCRIPCIÓN
           VALUES (400, 'BD445', 44)
```

Si no conocemos todos los datos —por ejemplo, si no conocemos el NúmerodePosición— podríamos decir:

```
INSERT      INTO INSCRIPCIÓN
           (NúmerodeEstudiante, NombredeClase)
           VALUES (400, 'BD445')
```

El NúmerodePosición se puede agregar después. Como podrá observar, esto ocasiona que el valor de NúmerodePosición tenga un valor nulo en el nuevo renglón.

También podemos copiar renglones en bloque de una tabla a otra. Por ejemplo, suponga que queremos llenar la tabla JUNIOR que se muestra en la figura 9-1.

```
INSERT      INTO JUNIOR
           VALUES
           (SELECT EID, Nombre, Especialidad
            FROM      ESTUDIANTE
            WHERE     Grado = 'JR')
```

Se pueden usar las expresiones contenidas en SELECT y todas las SELECT de SQL desarrolladas en las dos secciones anteriores para identificar los renglones que serán copiados. Esta característica ofrece capacidades bastante poderosas.

ELIMINACIÓN DE DATOS

Al igual que con la inserción, los renglones se pueden eliminar uno por uno o en grupos. El siguiente ejemplo suprime el renglón para Estudiante 100:

```
DELETE      FROM ESTUDIANTE
           WHERE ESTUDIANTE.EID = 100
```

Observe que si Estudiante 100 está inscrito en las clases, esta eliminación causará un problema de integridad: como los renglones de INSCRIPCIÓN tienen NúmerodeEstudiante = 100 no tendrán correspondencia con el renglón ESTUDIANTE.

Los grupos de renglones se pueden borrar como se muestra en los dos ejemplos siguientes, lo cual elimina todas las inscripciones para la especialidad de contabilidad, así como también a todos los estudiantes de esa especialidad.

```
DELETE FROM INSCRIPCIÓN
WHERE INSCRIPCIÓN.NúmeroEstudiante IN
      (SELECT ESTUDIANTE.EID
       FROM ESTUDIANTE
       WHERE ESTUDIANTE.Especialidad = 'Contabilidad')
DELETE FROM ESTUDIANTE
WHERE ESTUDIANTE.Especialidad = 'Contabilidad'
```

El orden de estas dos operaciones es importante, porque si se invirtiera ninguno de los renglones INSCRIPCIÓN se eliminarían debido a que los renglones coincidentes de ESTUDIANTE ya habrían sido eliminados.

MODIFICACIÓN DE DATOS

Los renglones también se pueden modificar uno a la vez o en grupos. La palabra reservada SET se usa para cambiar el valor de una columna. Después de SET, se cambia el nombre de la columna y se especifica el nuevo valor o la forma de calcularlo. Considere dos ejemplos:

```
UPDATE INSCRIPCIÓN
SET NúmeroPosición = 44
WHERE EID = 400
```

y

```
UPDATE INSCRIPCIÓN
SET NúmeroPosición = MAX (NúmeroPosición) + 1
WHERE EID = 400
```

En el segundo enunciado UPDATE, el valor de la columna se calcula usando la función MAX. Sin embargo, algunas implementaciones de SQL no permiten usar la función interconstruida como argumento en el comando SET.

Para ilustrar las actualizaciones en masa, suponga que el nombre de un curso ha cambiado de BD445 a BD564. En este caso, para evitar problemas de integridad se deben cambiar las tablas INSCRIPCIÓN y CLASE.

```
UPDATE INSCRIPCIÓN
SET NombreClase = 'BD564'
WHERE NombreClase = 'BD445'
UPDATE CLASE
SET NombreClase = 'BD564'
WHERE NombreClase = 'BD445'
```

Recuerde que las actualizaciones en bloque pueden ser bastante peligrosas. Se le da un gran poder al usuario —poder que cuando se usa en forma correcta ayuda a realizar rápidamente la tarea, pero cuando se usa erróneamente puede ocasionar serios problemas.

► RESUMEN

SQL es el lenguaje de manejo de datos relacionales más importante que hay. Se ha convertido en el estándar para el intercambio de información entre computadoras, y su popularidad continúa en aumento. Los enunciados SQL que operan en una sola tabla incluyen SELECT, SELECT con WHERE, SELECT con GROUP BY y SELECT con GROUP BY y HAVING. SQL también contiene las funciones interconstruidas de COUNT, SUM, AVG, MAX y MIN.

Las operaciones en dos o más tablas se pueden hacer usando subconsultas, join, EXISTS y NOT EXISTS. Las subconsultas y los join realizan muchas de las mismas operaciones, pero no las sustituyen completamente. Las subconsultas requieren que los atributos recuperados provengan de una relación independiente, pero los join, no. Por otro lado, algunas consultas son posibles con subconsultas y EXISTS y NOT EXISTS, las cuales son imposibles con los join.

Los enunciados SQL para la modificación de datos incluyen las órdenes INSERT, DELETE y UPDATE, que se usan para agregar, remover y cambiar valores de datos.

En este capítulo presentamos los comandos básicos SQL de manera genérica, y en los capítulos 13, 14, y 16 los usaremos para procesar una base de datos empleando productos comerciales DBMS.

► PREGUNTAS DEL GRUPO I

Las preguntas en este grupo se refieren a las tres relaciones siguientes:

VENDEDOR (Nombre, PorcentajeCuota, Salario)

PEDIDO (Número, NombreCliente, NombreVendedor, Cantidad)

CLIENTE (Nombre, Ciudad, TipoIndustria)

Una instancia de estas relaciones se muestra en la figura 9-3. Use los datos en esas tablas y muestre los enunciados SQL para desplegar o modificar datos como se indica en las siguientes preguntas:

- 9.1 Muestre los salarios de todos los vendedores.
- 9.2 Muestre los salarios de todos los vendedores, pero omita duplicados.
- 9.3 Muestre los nombres de todos los vendedores que están por abajo del 30% de la cuota.
- 9.4 Muestre los nombres de todos los vendedores que tengan un pedido con Abernathy Construction.
- 9.5 Muestre los nombres de todos los vendedores que ganan más de \$49999 y menos de \$100000.
- 9.6 Muestre los nombres de todos los vendedores con un PorcentajeCuota mayor a 49 y menor de 60. Use la palabra reservada BETWEEN.
- 9.7 Muestre los nombres de todos los vendedores con un PorcentajeCuota de más de 49 y menos de 60. Use la palabra reservada LIKE.
- 9.8 Muestre los nombres de los clientes que se localicen en una Ciudad que termine con S.
- 9.9 Muestre los nombres y los salarios de todos los vendedores que no tengan un pedido con Abernathy Construction, en orden ascendente de salario.
- 9.10 Calcule el número de pedidos.
- 9.11 Calcule el número de diferentes clientes que tienen un pedido.
- 9.12 Calcule el porcentaje promedio de cuota para los vendedores.

- 9.13 Muestre el nombre del vendedor con el porcentaje de cuota más alto.
- 9.14 Calcule el número de pedidos de cada vendedor.
- 9.15 Calcule el número de pedidos de cada vendedor, considerando sólo los pedidos que excedan de 500.
- 9.16 Muestre los nombres y porcentajes de los vendedores que tienen un pedido con ABERNATHY CONSTRUCTION, en orden descendente de porcentaje de cuota (use una subconsulta).
- 9.17 Muestre los nombres y porcentajes de cuota de los vendedores que tengan un pedido con ABERNATHY CONSTRUCTION, en orden descendente de porcentaje de cuota (use un join).
- 9.18 Muestre los porcentajes de cuota de los vendedores que tengan un pedido con un cliente en MEMPHIS (use una subconsulta).
- 9.19 Muestre los porcentajes de cuota de los vendedores que tengan un pedido con un cliente en MEMPHIS (use un join).
- 9.20 Muestre el tipo de industria y los nombres de los vendedores de todos los pedidos para las compañías en MEMPHIS.
- 9.21 Muestre los nombres de los vendedores junto con los nombres de los clientes que les hayan hecho un pedido. Incluya a los vendedores que no tengan pedidos. Use la notación de Microsoft Access.
- 9.22 Muestre los nombres de los vendedores que tengan dos o más pedidos.

► FIGURA 9-3

*Datos de muestra
para las preguntas
del grupo 1*

Nombre	Porcentaje- deCuota	Salario
Abel	63	120,000
Baker	38	42,000
Jones	26	36,000
Murphy	42	50,000
Zenith	59	118,000
Kobad	27	36,000

VENDEDOR

Número	NombredelCliente	NombredelVendedor	Cantidad
100	Abernathy Construction	Zenith	560
200	Abernathy Construction	Jones	1800
300	Manchester Lumber	Abel	480
400	Amalgamated Housing	Abel	2500
500	Abernathy Construction	Murphy	6000
600	Tri-City Builders	Abel	700
700	Manchester Lumber	Jones	150

PEDIDO

Nombre	Ciudad	TipodelIndustria
Abernathy Construction	Willow	B
Manchester Lumber	Manchester	F
Tri-City Builders	Memphis	B
Amalgamated Housing	Memphis	B

CLIENTE

- 9.23 Muestre los nombres y porcentajes de cuota de los vendedores que tengan dos o más pedidos.
- 9.24 Muestre los nombres y edades de los vendedores que tengan un pedido con todos los clientes.
- 9.25 Muestre un enunciado SQL para insertar un nuevo renglón en CLIENTE.
- 9.26 Muestre un enunciado SQL para insertar un nombre nuevo y edad en VENDEDOR; suponga que el salario no está determinado.
- 9.27 Muestre un enunciado SQL para insertar renglones en una tabla nueva, ALTORENDIMIENTO (Nombre, salario), el cual incluya que un vendedor debe tener un salario de cuando menos \$100000.
- 9.28 Muestre un enunciado SQL para borrar un cliente de ABERNATHY CONSTRUCTION.
- 9.29 Muestre un enunciado SQL para eliminar todos los pedidos de ABERNATHY CONSTRUCTION.
- 9.30 Muestre un enunciado SQL para cambiar el salario del vendedor JAIMES a \$45,000.
- 9.31 Muestre un enunciado SQL para dar a todos los vendedores un aumento de 10 por ciento.
- 9.32 Suponga que el vendedor JAIMES cambia su apellido a PARKER. Muestre el enunciado SQL que crea los cambios apropiados.

► PREGUNTAS DEL GRUPO II

- 9.33 Instale Access 2002 y abra la base de datos Northwind. Use la herramienta Query-by-design/SQL View; escriba los enunciados SQL para las siguientes preguntas e imprímalas.
 - a. Liste todas las columnas de proveedores
 - b. Liste NombredelaCompañía de los proveedores con NombredeCompañía empezando con “Nuevo”
 - c. Liste todas las columnas de los productos que abastecen los proveedores con NombredeCompañía empezando con “Nuevo”. Muestre las respuestas usando un join y una subconsulta
 - d. Liste NiveldeReordenamiento y cuente todos los productos
 - e. Liste NiveldeReordenamiento y cuente todos los NivelesdeReordenamiento que tengan más de un elemento
 - f. Liste NiveldeReordenamiento y cuente todos los NivelesdeReordenamiento que tengan más de un elemento para los productos de los proveedores cuyos nombres empiecen con “Nuevo”

► PREGUNTAS DEL PROYECTO FIREDUP

Suponga que FiredUp ha creado una base de datos con las siguientes tablas:

CLIENTE (ClienteSK, Nombre, Teléfono, CorreoElectrónico)

ESTUFA (NúmerodeSerie, Tipo, Versión, FechadeFabricación)

REGISTRO (ClienteSK, NúmerodeSerie, Fecha)

REPARACIÓN_ESTUFA (NúmerodeFacturadeReparación, NúmerodeSerie, Fecha, Descripción, Costo, ClienteSK)

Codifique SQL para lo siguiente: suponga que todas las fechas están en el formato *mm-ddaaaa*.

- A. Muestre todos los datos en cada una de las cuatro tablas FiredUp.
- B. Liste las versiones de todas las estufas.
- C. Liste las versiones de todas las estufas del tipo "FiredNow".
- D. Liste NúmerodeSerie y Fecha de todos los registros en el año 2000.
- E. Liste NúmerodeSerie y Fecha de todos los registros en Febrero. Use el comodín guión bajo (_).
- F. Liste NúmerodeSerie y Fecha de todos los registros en Febrero. Use el comodín (%).
- G. Liste los nombres y direcciones de CorreoElectrónico de los clientes que lo tengan.
- H. Liste los nombres de todos los clientes que no tengan CorreoElectrónico; presente los resultados en orden descendente de Nombre.
- I. Determine el costo máximo de una reparación de estufa.
- J. Determine el promedio del costo de una reparación de estufa.
- K. Cuento todas las estufas.
- L. Cuento las estufas de cada tipo y despliegue el Tipo y el total.
- M. Liste los nombres y direcciones de correo electrónico de todos los clientes que han tenido una reparación de una estufa superior a \$50. Use una subconsulta.
- N. Liste los nombres y las direcciones de correo electrónico de todos los clientes que tienen registrado un tipo de estufa "FiredNow". Use una subconsulta.
- O. Liste los nombres y las direcciones de correo electrónico de todos los clientes que tengan una reparación que haya costado más de \$50. Use un join.
- P. Liste los nombres y las direcciones de correo electrónico de todos los clientes que tengan registrada un tipo de estufa "FiredNow". Use un join.
- Q. Liste los nombres, direcciones de correo electrónico y fecha de registro de todos los registros de los clientes.
- R. Muestre los nombres y direcciones de correo electrónico de todos los clientes que tengan registrada una estufa, pero que no hayan solicitado alguna reparación.
- S. Muestre los nombres y las direcciones de correo electrónico de todos los clientes que tengan registrada una estufa, pero que no hayan tenido que repararla.

Diseño de aplicaciones de bases de datos

Este capítulo introduce los conceptos fundamentales para el diseño de aplicaciones de bases de datos. Iniciamos con la enumeración y el análisis de las funciones de la aplicación de una base de datos. Cada función se describe detalladamente usando como ejemplo la aplicación de una galería de arte. Las ideas que se presentan en este capítulo son relativas a las aplicaciones de bases de datos que se desarrollan para usarlas en ambientes tradicionales tales como Windows. En los capítulos 14 al 16 se ampliarán estos conceptos para el diseño de las aplicaciones que usan la tecnología Internet.

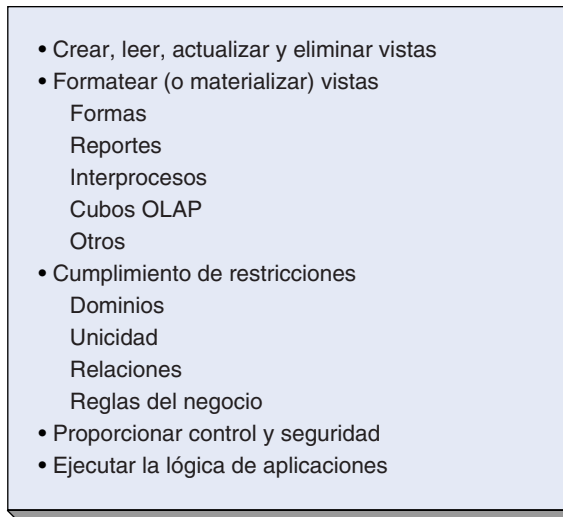
Antes de comenzar conviene una observación acerca de la terminología. En las primeras etapas del procesamiento de bases de datos era muy fácil encontrar un nexo entre el DBMS y la aplicación —las aplicaciones eran programas independientes que invocaban al DBMS—. Hoy, especialmente con los productos de escritorio del DBMS tales como Microsoft Access, esa conexión se ha vuelto confusa. Para evitar malos entendidos en este capítulo, supondremos que todas las formas, reportes, menús y cualquier código de programa que esté contenido en cualesquiera de ellos, son parte de la aplicación de la base de datos. Además, los programas que llaman al DBMS son también parte de la aplicación. Cualquier estructura, regla o restricción que sea parte de la tabla, así como las definiciones de la relación, las maneja el DBMS y son parte de la base de datos. Así, una regla que se coloca en determinada columna de una tabla durante la definición de ésta se vuelve obligatoria mediante el DBMS. La misma regla, puesta en una forma de control, se considera obligatoria mediante la aplicación de la base de datos. A medida que avancemos podremos ver la importancia de esta diferencia.

► FUNCIONES DE UNA APLICACIÓN DE BASE DE DATOS

La figura 10-1 lista las funciones de una aplicación de base de datos. La primera es procesar vistas de datos. Hay cuatro funciones de procesamiento básico: crear, leer, actualizar y eliminar. En inglés a veces a estas funciones se les denomina mediante el acrónimo **CRUD**. Así, la primera tarea de una aplicación de base de datos es para vistas CRUD.

► FIGURA 10-1

Funciones de una aplicación de base de datos



Una vista de aplicación es más que un renglón de una tabla y más que el resultado de un enunciado SQL. Con frecuencia la construcción de una vista de aplicación requiere dos o más enunciados SQL, como se verá en la siguiente sección.

Una segunda tarea de la aplicación es **formatear** o **materializar** las vistas que están siendo procesadas. Observe que existe una diferencia entre los datos contenidos (la vista) y la apariencia de ese contenido (el formato o materialización). Una vista determinada por lo general tiene diferentes formatos. La distinción entre contenido y forma es especialmente importante en las aplicaciones de bases de datos que usan la tecnología Internet.

Hay varios tipos de materialización de vistas. Las formas y reportes son los dos más comunes; sin embargo, otros también son importantes. Una materialización del interproceso se usa para enviar una vista de un servidor a otro, o de una aplicación a otra. El formato de tales materializaciones se determina mediante una interfaz estándar como Microsoft COM, o un protocolo como XML, que presentaremos en el capítulo 14. El intercambio electrónico de datos es un buen ejemplo de su uso. Otros tipos de materialización son más especializadas. En el capítulo 17 verá el papel que desempeñan los cubos OLAP. El lenguaje natural es otro tipo de ejemplo.

El cumplimiento de restricciones es una tercera función. Éstas pueden ser estructurales, por ejemplo que los datos de valores requeridos se ajusten a las especificaciones de dominio, con lo que se asegura la unicidad y se imponen restricciones a la relación. Otras restricciones implican reglas del negocio, tales como: “Ningún vendedor puede tratar con ningún cliente cuya dirección asentada en el contrato esté fuera de la región que tiene asignada.”

La cuarta función de una aplicación de bases de datos es proporcionar mecanismos de seguridad y control. La aplicación funcionará en conjunto con el sistema operativo y el DBMS para aumentar la seguridad que proporcionan los nombres de los usuarios y las contraseñas (passwords). Los menús y estructuras similares determinan qué y cuándo pueden llevar a cabo determinadas acciones los usuarios.

La última tarea de una aplicación de base de datos es ejecutar la lógica del negocio. Por ejemplo, en una aplicación de captura de pedidos, cuando un cliente ordena cinco ejemplares de un libro, la aplicación necesita reducir automáticamente la cantidad de libros en el almacén. Si no hay ejemplares suficientes en inventario, o si la cantidad disponible es menor al punto de reorden, también se necesita llevar a cabo otra acción.

Las últimas dos funciones se abordan detalladamente en las clases de desarrollo de sistemas, por lo tanto casi no las abordaremos. Sin embargo, las primeras tres funciones son específicas de las aplicaciones de base de datos, y por ende centraremos nuestra atención en ellas.

Antes de aprender mayores detalles acerca de dichas funciones considere las necesidades para una base de datos y la aplicación con respecto a la galería View Ridge.

► CASO DE APLICACIÓN: GALERÍA VIEW RIDGE

La galería View Ridge es una pequeña empresa que vende arte contemporáneo, incluyendo litografías, pinturas originales y fotografías. Todas las litografías y fotografías están firmadas y numeradas y la mayoría de las piezas cuestan entre \$1000 y \$25000. View Ridge ha estado en el negocio durante 27 años; tiene un solo propietario que le dedica tiempo completo, así como tres vendedores y dos trabajadores que elaboran marcos, cuelgan los cuadros en la galería y preparan las obras de arte para su envío.

View Ridge realiza exposiciones y otros eventos con el fin de atraer clientes. Las obras de arte también se exhiben en compañías locales, restaurantes y otros lugares públicos. View Ridge es dueña de todo el material que vende; no tiene obras de arte a consignación.

REQUERIMIENTOS PARA LA APLICACIÓN

Los requerimientos para la aplicación View Ridge se resumen en la figura 10-2. En primer lugar, tanto el dueño como los vendedores desean dar seguimiento a sus clientes, así como a los intereses de éstos en cuanto a la compra de arte. Los vendedores necesitan saber a quién contactar cuando llega nuevo material, y tener toda esa información para crear cartas personalizadas y comunicarse verbalmente con sus clientes.

Además, la base de datos debe registrar las compras de arte de los clientes para que los vendedores dediquen más tiempo a los compradores más activos. En ocasiones también necesitan usar los registros de adquisiciones para ubicar las obras de arte, porque a veces la galería compra nuevamente algunas obras que son difíciles de encontrar y las revende. La aplicación de la base de datos debe tener un formato para agregar las nuevas obras que compra la galería.

View Ridge quiere que su aplicación de base de datos proporcione una lista de los artistas y los trabajos que se han exhibido en la galería. Al dueño también le gustaría poder determinar con qué rapidez se vende el trabajo de un artista y cuáles son los márgenes de utilidad, y la aplicación de base de datos debe mostrar un inventario actual en una página Web a la que los clientes puedan tener acceso a través de Internet. Por último, la galería View Ridge quiere que la base de datos produzca reportes que aminoren la carga de trabajo del contador de tiempo parcial que está contratado.

DISEÑO DE LA BASE DE DATOS

La figura 10-3(a) muestra la estructura de los objetos que se requieren para manejar la base de datos de la galería. CLIENTE (CUSTOMER) y ARTISTA (ARTIST) son objetos compuestos y TRABAJO (WORK) es un objeto híbrido cuyo único identificador es el grupo {ARTISTA, Título y Copia} {ARTIST, Title y Copy}. El grupo multivaluado Transacción (Transaction) representa las adquisiciones y las ventas. Considerando que determinada obra puede pasar por la galería varias veces (debido a las adquisiciones y a

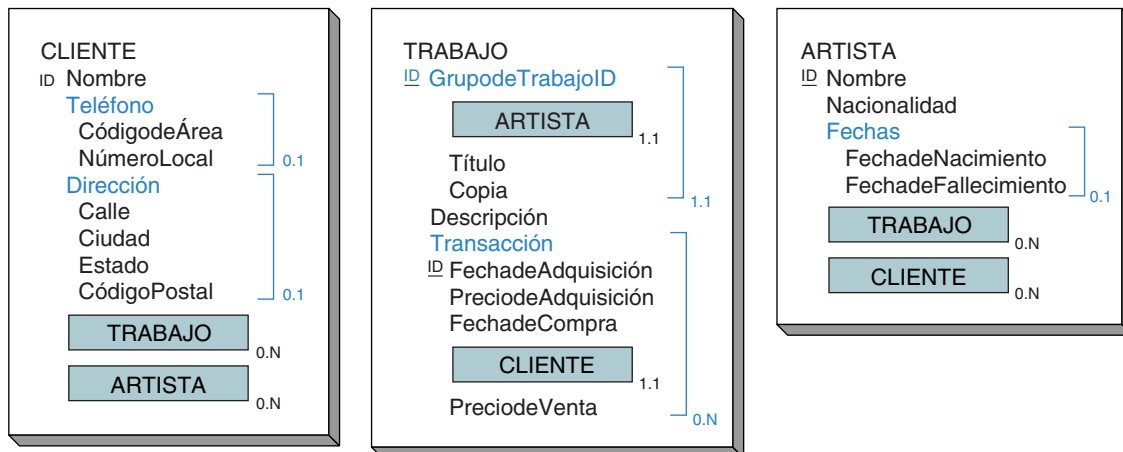
► FIGURA 10-2

*Resumen de los
requerimientos
para las aplicaciones
de la base de
datos de la galería
View Ridge*

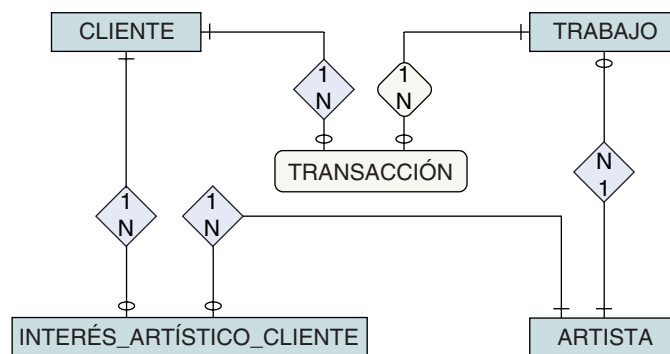
- Dar seguimiento a los clientes y a sus intereses de compra.
- Registrar las compras de material artístico de los clientes.
- Registrar las compras.
- Listar los artistas y las obras que se han exhibido en la galería.
- Reportar con qué rapidez se vende el trabajo de un artista y con qué margen de utilidad.
- Mostrar el inventario actual en una página Web.
- Listar reportes que empleará el contador de la galería.

► FIGURA 10-3

Diseño de la base de datos de la galería View Ridge: (a) objetos semánticos, (b) diagrama E-R, (c) diseño relacional, (d) diseño relacional con llaves sustitutas, (e) diagrama de relación de Access



(a)



(b)

CLIENTE (NúmerodeCliente, Nombre, CódigodeÁrea, NúmeroLocal, Calle, Ciudad, Estado, CódigoPostal)

TRABAJO (NombredelArtista, Título, Copia, Descripción)

TRANSACCIÓN (NombredelArtista, Título, Copia, FechadeAdquisición, PreciodeAdquisición, FechadeCompra, NúmerodeCliente, PreciodeVenta)

ARTISTA (NombredelArtista, Nacionalidad, FechadeNacimiento, FechadeFallecimiento)

INTERÉS_ARTÍSTICO_CLIENTE (NúmerodeCliente, NombredelArtista)

(c)

los intercambios), Transacción tiene valores múltiples. TRABAJO es un objeto híbrido porque contiene el atributo objeto CLIENTE en el grupo Transacción multivaluado (o multivalor).

View Ridge quiere registrar los intereses de sus clientes. En particular, desea conocer los artistas en los cuales un cliente en particular está interesado, y aquellos a los que les interesa un artista en especial. Estos requisitos se logran colocando los atributos multivaluados ARTISTA en CLIENTE, y CLIENTE en ARTISTA. En la figura 10-3(b) se muestra un diagrama E-R.

► FIGURA 10-3

(Continuación)

CLIENTE (ClienteID, Nombre, CódigodeÁrea, NúmeroLocal, Calle, Ciudad, Estado, CódigoPostal)
 CUSTOMER (CustomerID, Name, AreaCode, LocalNumber, Street, City, State, Zip)
 TRABAJO (TrabajoID, ArtistaID, Título, Copia, Descripción)
 WORK (WorkID, ArtistID, Title, Copy, Description)
 TRANSACCIÓN (TransacciónID, TrabajoID, FechadeAdquisición, PreciodeAdquisición,
 FechadeCompra, ClienteID, PreciodeVenta)
 TRANSACTION (TransactionID, WorkID, DateAcquired, AcquisitionPrice,
 PurchaseDate, CustomerID, SalesPrice)
 ARTISTA (ArtistaID, NombredelArtista, Nacionalidad, FechadeNacimiento, FechadeFallecimiento)
 ARTIST (ArtistID, ArtistName, Nationality, Birthdate, DeceasedDate)
 INTERÉS_ARTÍSTICO_CLIENTE (ClienteID, ArtistaID)
 CUSTOMER_ARTIST_INT (CustomerID, ArtistID)

Restricciones de integridad referencial:

ArtistaID en TRABAJO debe existir en ArtistaID en ARTISTA

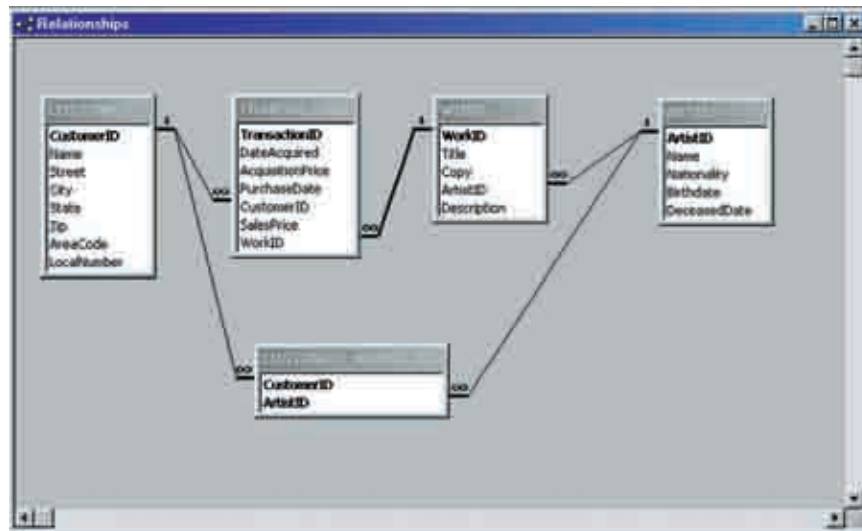
TrabajoID en TRANSACCIÓN debe existir en TrabajoID en TRABAJO

ClienteID en TRANSACCIÓN debe existir en ClienteID en CLIENTE

ClienteID en INTERÉS_ARTÍSTICO_CLIENTE debe existir en ClienteID en CLIENTE

ArtistaID en INTERÉS_ARTÍSTICO_CLIENTE debe existir en ArtistaID en ARTISTA

(d)



(e)

La representación relacional de los objetos de View Ridge se muestra en la figura 10-3(c). Puesto que el objeto CLIENTE no tiene un identificador único, se debe crear uno para usarlo como llave. Aquí hemos agregado un número de identificación: Número de Cliente. La cardinalidad máxima de los grupos Teléfono y Dirección es 1, así que los atributos en esos grupos se pueden colocar en la tabla CLIENTE; de hecho, no aparecen en la tabla como grupos, sino que la información del grupo se usa después para construir formas.

La llave de TRABAJO consiste en la llave de ARTISTA más Título y Copia. El único atributo sin la llave de TRABAJO es Descripción. Puesto que el atributo Transacción tie-

ne valores múltiples, debe creársele una tabla. Su llave es la del objeto en el que está contenida (NombredelArtista, Título, Copia) más la llave del grupo, la cual es Fecha-deAdquisición. Observe también que NúmerodeCliente se toma como llave externa en TRANSACCIÓN.

El objeto ARTISTA se representa mediante una tabla independiente, y NombredelArtista se puede usar como la llave de la tabla porque los nombres de los artistas están modelados para ser únicos. La tabla de intersección INTERÉS_ARTÍSTICO_CLIENTE se debe crear para desarrollar la relación M:N entre CLIENTE y ARTISTA.

Debido al número de llaves de texto, y en particular a la gran llave compuesta de TRANSACCIÓN, este diseño puede mejorar sustancialmente si se reemplazan llaves de datos con llaves sustitutas. Esto se ha hecho en el diseño que se muestra en la figura 10-3(d). Los nombres de las llaves sustitutas se construyen anexando las letras ID al nombre de la tabla.

El diseño de la figura 10-3(d) es mejor porque hay menos datos duplicados. No es necesario copiar las columnas NombredelArtista, Título y Copia en la tabla Transacción. Debido a que puede haber muchas transacciones para un trabajo determinado, estos ahorros son valiosos. La figura 10-3(e) muestra un diagrama de relación para el diseño de la llave sustituta.

► CREAR, LEER, ACTUALIZAR Y BORRAR INSTANCIAS DE VISTAS

Una **vista** es una lista estructurada de elementos de datos de entidades u objetos semánticos definidos en el modelo. Un ejemplo de una vista es aquél que contiene datos para una entidad u objeto semántico. La figura 10-4(a) muestra un ejemplo de vista para la base de datos de la galería en la figura 10-3. Esta vista muestra datos de un cliente, sus transacciones e intereses artísticos. En esta vista existen potencialmente muchas TRANSACCIONES para cada CLIENTE, y cada TRANSACCIÓN tiene un TRABAJO. También, pueden existir muchos nombres de ARTISTA por cada CLIENTE (los artistas en los cuales el cliente está interesado).

Observe que ARTISTA.Nombre está repetido en esta vista. La primera vez es el nombre de un artista cuyo trabajo ha comprado el CLIENTE. La segunda, es el nombre del artista que le interesa al CLIENTE. Por ejemplo, en la figura 10-4(b) un cliente ha comprado trabajos de Tobey y de Miró y está interesado no sólo en el trabajo de estos dos artistas, sino también en el de Dennis Frings. En este caso, Tobey y Miró aparecen dos veces en la vista, una como un valor de TRABAJO.ARTISTA.Nombre y otra como un valor de ARTISTA.Nombre.

De nuevo, una vista es una lista estructurada de atributos. Debido a que está estructurada, los atributos pueden encontrarse más de una vez. También observe que una vista es sólo una lista de valores de datos. Puede ser formateada o materializada de muchas formas diferentes —como forma, reporte, u otro tipo de materialización.

Ahora considere las acciones CRUD que se pueden ejecutar en una vista. Comenzamos con Leer.

LECTURA DE EJEMPLOS DE VISTAS

Para leer una vista la aplicación debe ejecutar uno o más enunciados SQL para obtener valores de datos y luego poner los valores resultantes en la estructura de la vista. La vista de Cliente en la figura 10-4 contiene datos en dos caminos de acceso: uno a través de la tabla TRANSACCIÓN, y el otro mediante la tabla INTERÉS_ARTÍSTICO_CLIENTE. La estructura de los enunciados SQL es tal que sólo se puede seguir un camino a través del esquema con un enunciado SQL independiente. Así, esta vista requerirá un enunciado SQL para cada camino. Para el primero el siguiente enunciado SQL obtendrá los datos necesarios para el nombre del cliente “Jackson, Mary”:

```
SELECT  CLIENTE.ClienteID, CLIENTE.Nombre,
        CLIENTE.CódigodeÁrea, CLIENTE.NúmeroLocal,
```

► FIGURA 10-4

Vista de Cliente:
 (a) lista estructurada
 de atributos, y
 (b) datos de muestra

CLIENTE.Nombre
 CLIENTE.Código de Área
 CLIENTE.Número Local
 TRANSACCIÓN.Fecha de Compra
 TRANSACCIÓN.Precio de Venta . . .
 TRABAJO.ARTISTA.Nombre
 TRABAJO.Título
 TRABAJO.Copia
 ARTISTA.Nombre . . .

(a) Lista estructurada de atributos

Jackson, Elizabeth		
206		
284-6783		
	7/15/94	
	4,300	
		Mark Tobey
		Lithograph One
		10/75
	2/5/99	
	2,500	
		Juan Miró
		Poster 14
		5/250
	11/22/97	
	17,850	
		Juan Miró
		Awakening Child
		1/1
	Juan Miró	
	Mark Tobey	
	Dennis Frings	

(b) Datos de muestra

```

FROM  ARTISTA.Nombre, TRABAJO.Título, TRABAJO.Copia,
      TRANSACCIÓN.Fecha de Compra, TRANSACCIÓN.Precio de Venta
WHERE CLIENTE, TRANSACCIÓN, TRABAJO, ARTISTA
      CLIENTE.ClienteID = TRANSACCIÓN.ClienteID
      AND TRABAJO.TrabajoID = TRANSACCIÓN.TrabajoID
      AND ARTISTA.ArtistaID = TRABAJO.ArtistaID
      AND CLIENTE.Nombre = 'Jackson, Mary'
```

Repasemos este enunciado SQL mientras revisamos el diagrama de relación en la figura 10-3(e). Los tres joins son necesarios para obtener datos a través de las tres relaciones en la parte superior del diagrama.

En el contexto del desarrollo de la aplicación, al resultado de un enunciado SQL a veces se le llama **recordset**. Para Microsoft, este término significa una relación con un empaquetador de programación de objetos. Un recordset tiene tanto métodos como

propiedades. *Open* es un ejemplo de método del recordset, *Cursor Type* es un ejemplo de una propiedad del recordset. Aprenderá más acerca de esto en el capítulo 14.

Para obtener los intereses de un cliente en determinado artista, se requiere un enunciado SQL para seguir el camino a través de INTERÉS_ARTÍSTICO_CLIENTE. El enunciado SQL para este camino es:

```
SELECT  CLIENTE.ClienteID, ARTISTA.Nombre
FROM    CLIENTE, INTERÉS_ARTÍSTICO_CLIENTE, ARTISTA
WHERE   CLIENTE.ClienteID = INTERÉS_ARTÍSTICO_CLIENTE.ClienteID
        AND INTERÉS_ARTÍSTICO_CLIENTE.ArtistaID = ARTISTA.ArtistaID
        AND CLIENTE.Nombre = 'Jackson, Mary'
```

Debido a que CLIENTE.Nombre no es único, posiblemente los dos recordset de estos enunciados recuperarán datos de más de un cliente. Por lo tanto, la aplicación necesita tener lógica para examinar los valores de ClienteID en el recordset y asociar los renglones correctos.

Después de ejecutar estos dos enunciados, la aplicación tiene todos los datos necesarios para construir uno o más ejemplos de la vista que se presenta en la figura 10-3. Cómo se hace esto depende del lenguaje que se utilice. En COBOL la información se coloca en estructuras definidas en el Data Division. En Visual Basic la información se puede colocar en una estructura de datos o una serie de arreglos. En C++ y Java la información se coloca en los objetos. En realidad esos puntos no nos preocupan; lo que importa es que usted logre tener una idea completa de cómo debe ejecutar la aplicación uno o más enunciados SQL para llenar la estructura de la vista de datos. Véase el capítulo 16 para tener un ejemplo de Java.

CREACIÓN DE INSTANCIAS DE UNA VISTA

Para crear una instancia de una vista, la aplicación primero debe obtener los nuevos valores de datos y relaciones. Esto se hace mejor mediante un formato de entrada de información, pero también recibe datos de otros programas, y en otras maneras. Una vez que la aplicación tiene valores de datos, entonces ejecuta los enunciados SQL para almacenar la información en la base de datos.

Considere la vista Nuevo Cliente en la figura 10-5; se usa cuando un nuevo cliente compra una pintura. Contiene información acerca de él, de la transacción de compra y de sus múltiples intereses en obras de arte. Esta vista difiere de la de la figura 10-4 porque tiene más datos de clientes y también permite sólo una transacción. Sin embargo,

► FIGURA 10-5

Nueva vista de
Cliente

```
CLIENTE.Nombre
CLIENTE.CódigodeÁrea
CLIENTE.NúmeroLocal
CLIENTE.Dirección
CLIENTE.Ciudad
CLIENTE.Estado
CLIENTE.CódigoPostal
TRANSACCIÓN.Fecha de Adquisición
TRANSACCIÓN.Precio de Adquisición
TRANSACCIÓN.Fecha de Compra
TRANSACCIÓN.Precio de Venta
TRABAJO.ARTISTA.Nombre
TRABAJO.Título
TRABAJO.Copia
ARTISTA.Nombre. . .
```

puede haber valores múltiples ARTISTA.Nombre que registren los nuevos intereses del cliente.

Suponga que los valores de los datos para esta vista se localizan en una estructura de programa llamada NuevoCliente; además suponga que podemos acceder a los valores en la estructura anexando los caracteres NuevoCliente a los nombres en la estructura. Así, NuevoCliente.CLIENTE.Nombre tiene acceso a Nombre del CLIENTE en la estructura NuevoCliente.

Para crear esta vista en la base de datos debemos almacenar los nuevos datos del cliente en CLIENTE, los nuevos datos de transacción en TRANSACCIÓN y crear un renglón en la tabla de intersección INTERÉS_ARTÍSTICO_CLIENTE para cada uno de los artistas que le interesan al cliente.

El siguiente enunciado SQL almacenará los nuevos datos del cliente:

```
INSERT INTO CLIENTE
(CLIENTE.Nombre,
CLIENTE.Código de Área,
CLIENTE.Número Local,
CLIENTE.Dirección,
CLIENTE.Ciudad,
CLIENTE.Estado,
CLIENTE.Código Postal)
VALUES (NuevoCliente.CLIENTE.Nombre,
NuevoCliente.CLIENTE.Código de Área,
NuevoCliente.CLIENTE.Número Local,
NuevoCliente.CLIENTE.Dirección,
NuevoCliente.CLIENTE.Ciudad,
NuevoCliente.CLIENTE.Estado,
NuevoCliente.CLIENTE.Código Postal)
```

Suponga que cuando se crea el nuevo renglón el DBMS asigna el valor de la llave sustituta CLIENTE.ClienteID. Necesitaremos los valores de esta llave para terminar la creación de la nueva instancia, así que la aplicación necesitará obtenerla. Una manera de hacerlo es ejecutar el siguiente SELECT de SQL:

```
SELECT CLIENTE.ClienteID, CLIENTE.Código de Área,
CLIENTE.Número Local
FROM CLIENTE
WHERE CLIENTE.Nombre = NuevoCliente.CLIENTE.Nombre
```

Debido a que CLIENTE.Nombre no es único, puede aparecer más de un renglón en el recordset. En este caso, lo correcto sería identificarlo examinando la información de número de teléfono. Suponga que esto se ha realizado si es necesario y que el valor correcto está colocado en la estructura del programa como NuevoCliente.CLIENTE.ClienteID.

Un enunciado INSERT también será utilizado para almacenar el nuevo registro TRANSACCIÓN. Sin embargo, en este caso los valores de las llaves externas TRANSACCIÓN.TrabajoID y TRANSACCIÓN.ClienteID tendrán que ser proporcionados. Ya hemos mostrado cómo obtener el valor de ClienteID, así que lo único que falta es obtener el valor de TrabajoID. El siguiente SQL hará esto:

```
SELECT TRABAJO.TrabajoID
FROM TRABAJO,ARTISTA
WHERE TRABAJO.ArtistaID = ARTISTA.ArtistaID
AND ARTISTA.Nombre = NuevoCliente.TRABAJO.ARTISTA.Nombre
AND TRABAJO.Título = NuevoCliente.TRABAJO.Título
AND TRABAJO.Copia = NuevoCliente.TRABAJO.Copia
```

Suponga que el valor de la llave sustituta que se ha regresado se almacena como NuevoCliente.TRABAJO.TrabajoID.

Se puede ejecutar el siguiente SQL para agregar el nuevo renglón TRANSACCIÓN:

```
INSERT INTO TRANSACCIÓN
    (TRANSACCIÓN.TrabajoID,
    TRANSACCIÓN.FechadeAdquisición,
    TRANSACCIÓN.PreciodeAdquisición,
    TRANSACCIÓN.FechadeCompra,
    TRANSACCIÓN.ClienteID,
    TRANSACCIÓN.PreciodeVenta)
VALUES
    (NuevoCliente.TRABAJO.TrabajoID,
    NuevoCliente.TRANSACCIÓN.FechadeAdquisición,
    NuevoCliente.TRANSACCIÓN.PreciodeAquisición,
    NuevoCliente.TRANSACCIÓN.FechadeCompra,
    NuevoCliente.CLIENTE.ClienteID,
    NuevoCliente.TRANSACCIÓN.PreciodeVenta)
```

Ahora todo lo que falta es crear renglones para la tabla de intersección INTERÉS_ARTÍSTICO_CLIENTE. Para hacerlo, necesitamos obtener el ArtistaID de cada artista que le interesa al cliente, y luego crear un nuevo renglón en la tabla de intersección. El siguiente pseudocódigo ilustra la lógica:

Para cada NuevoCliente.ARTISTA.Nombre:

```
SELECT ARTISTA.ArtistaID
FROM ARTISTA
WHERE ARTISTA.Nombre = NuevoCliente.ARTISTA.Nombre
INSERT INTO INTERÉS_ARTÍSTICO_CLIENTE
    (ClienteID, ArtistaID)
VALUES (NuevoCliente.CLIENTE.ClienteID,
    ARTISTA.ArtistaID)
```

El siguiente NuevoCliente.ARTISTA.Nombre

En este punto, la vista de Nuevo Cliente se ha almacenado en la base de datos. Por supuesto, una aplicación completa incluye la lógica para captar los errores que ha regresado el DBMS y procesarlos. Por ejemplo, la aplicación debe manejar los casos de TRABAJO que no existen en la base de datos, y los de un ARTISTA que tampoco tiene registrado.

ACTUALIZACIÓN DE INSTANCIAS DE VISTAS

La tercera acción fundamental para realizar una vista es la actualización. Cuando se actualiza una vista hay tres tipos de cambio que son posibles: uno es un cambio simple de valores, como por ejemplo un cliente que cambia de número telefónico. Otra es un cambio a una relación; por ejemplo cuando un cliente ya no tiene interés en determinado artista. Un tercer tipo de actualización requeriría agregar uno o más nuevos renglones; eso ocurriría en nuestro ejemplo cuando un cliente hiciera una nueva compra.

El primer tipo de actualización se puede lograr con enunciados UPDATE SQL. Por ejemplo, suponga que un programa tiene una estructura llamada ActualizaciónCliente

(UpdateCust), y que ésta a su vez tiene ClienteID, CódigodeÁrea y NúmeroLocal. El siguiente SQL actualizará los nuevos valores:

```
UPDATE CLIENTE
SET CLIENTE.CódigodeÁrea = ActualizaciónCliente.CódigodeÁrea
  CLIENTE.NúmeroLocal = ActualizaciónCliente.NúmeroLocal
WHERE CLIENTE.ClienteID = ActualizaciónCliente.ClienteID
```

Los cambios a relaciones también son directos. Si la relación es uno a muchos, entonces sólo se necesita actualizar el valor de la llave externa conforme al nuevo valor. Por ejemplo, suponga que la relación de DEPARTAMENTO con EMPLEADO es 1:N. Entonces Depto# (u otra llave) será almacenado como una llave externa en EMPLEADO. Para mover un EMPLEADO a un nuevo DEPTO, la aplicación sólo necesita cambiar el valor Depto# por uno nuevo.

Si la relación es muchos a muchos, entonces la llave externa en la tabla de intersección necesitará ser modificada. Por ejemplo, si en la galería un cliente ya no se interesa por Mark Tobey sino por Dennis Frings, entonces el renglón de intersección que representa la conexión con Mark Tobey necesita ser cambiado para que apunte a Dennis Frings.

Suponga que ActualizaciónCliente.ClienteID tiene el ID del cliente, ActualizaciónCliente.ViejoArtistaID tiene el ID del renglón de Mark Tobey en ARTISTA, y ActualizaciónCliente.NuevoArtistaID tiene el ID del renglón de Dennis Frings en ARTISTA. El siguiente SQL hará el cambio necesario:

```
UPDATE INTERÉS_ARTÍSTICO_CLIENTE
SET INTERÉS_ARTÍSTICO_CLIENTE.ArtistaID = ActualizaciónCliente.NuevoArtistaID
WHERE INTERÉS_ARTÍSTICO_CLIENTE.ClienteID = ActualizaciónCliente.ClienteID
  AND INTERÉS_ARTÍSTICO_CLIENTE.ArtistaID = ActualizaciónCliente.ViejoArtistaID
```

En una relación muchos a muchos también es posible eliminar una conexión sin reemplazarla, y crear una nueva sin desaparecer una vieja. Para borrar una conexión sólo eliminaríamos el renglón apropiado en la tabla de intersección. Para crear una agregaríamos un nuevo renglón en la tabla de intersección.

El tercer tipo de actualización requiere aumentar un nuevo renglón en una o más tablas. Si, por ejemplo, un cliente hace una nueva compra, entonces se necesitará crear otro renglón en la tabla de TRANSACCIÓN. Esto se puede realizar de la misma manera en que un nuevo renglón de TRANSACCIÓN se agrega cuando se crea una nueva instancia de vista.

BORRADO DE INSTANCIAS DE VISTA

Borrar una vista implica eliminar renglones de las tablas que la integran. La clave está en saber cuánto borrar. Por ejemplo, suponga que la galería quiere eliminar información de un cliente cuyo nombre es “Jones, Mary”. Obviamente, es necesario eliminar el renglón “Jones, Mary” en CLIENTE. Todo lo de los renglones de intersección en INTERÉS_ARTÍSTICO_CLIENTE que corresponden a esa cliente también deberá ser eliminado. ¿Pero qué hay acerca de los renglones en la tabla de TRANSACCIÓN? Esta tabla contiene ClienteID y si su renglón es eliminado todos los renglones en TRANSACCIÓN que tienen su valor de ClienteID tendrán datos inválidos.

Las respuestas a preguntas como esta surgen a partir del modelo de datos. En el caso del modelo E-R, todas las entidades débiles son eliminadas si la entidad de la que dependen también lo es. De otra manera, ningún renglón adicional de la tabla es suprimido. En el caso del modelo de objeto semántico todos los datos dentro de un objeto

son eliminados (puede haber muchas tablas en un objeto si éste tiene atributos multivaluados), pero no se suprime ningún dato en un objeto diferente. En adición a estas reglas, no se debe permitir ninguna eliminación si puede causar una violación a las cardinalidades de la relación. Tocaremos este tema ahora, y más adelante lo trataremos con más profundidad.

Considere el modelo en la figura 10-3. Cuando se borra un cliente, el renglón en CLIENTE es eliminado junto con todos sus renglones en INTERÉS_ARTÍSTICO_CLIENTE correspondientes. Los datos de TRANSACCIÓN no serán eliminados porque residen en un objeto diferente llamado TRABAJO. Sin embargo, observe que la cardinalidad mínima de CLIENTE en TRANSACCIÓN es 1. Por lo tanto, si un CLIENTE específico está ligado a una TRANSACCIÓN, entonces éste se requiere y su eliminación no debe ser permitida.

También podemos concluir de la figura 10-3 que si una vista de TRABAJO es eliminada, entonces el renglón TRABAJO y todos los renglones de TRANSACCIÓN relacionados se borrarán.

Por último, si se suprime una vista de ARTISTA, entonces los renglones ARTISTA e INTERÉS_ARTÍSTICO_CLIENTE se borrarán. Además, si hay algún objeto TRABAJO que esté ligado a ese ARTISTA, entonces la eliminación no podrá ser permitida.

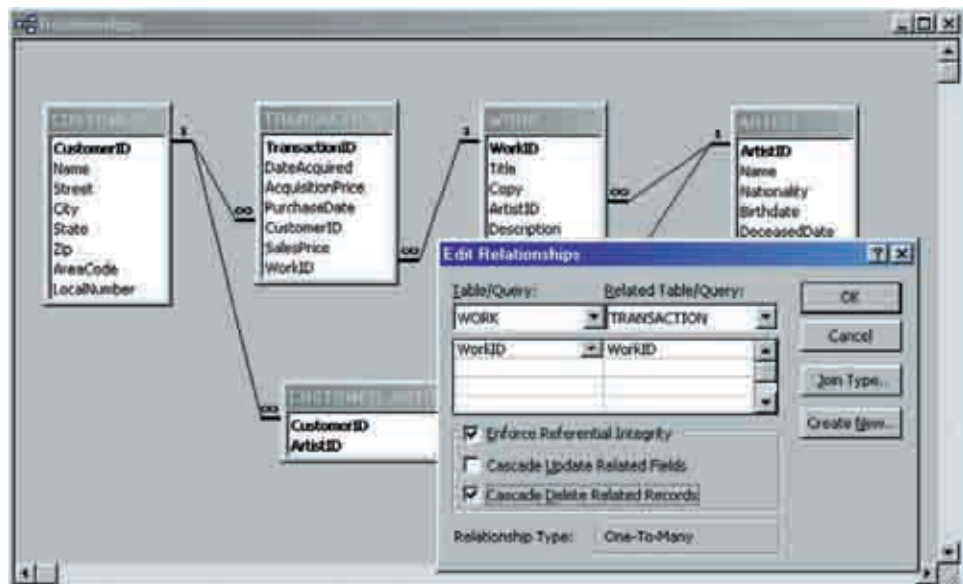
Por lo general algunos productos DBMS soportan la eliminación de los renglones dependientes bajo el término **eliminaciones en cascada**. La figura 10-6 muestra la caja de diálogo de la relación para Access. Observe que ha sido verificada la caja de registros de la relación de eliminación en cascada. Esto significa que cuando un renglón WORK (TRABAJO) es suprimido, Access automáticamente elimina cualquier renglón relacionado con TRANSACTION (TRANSACCIÓN).

Como se estableció, la cardinalidad de la relación desempeña un papel importante en la decisión de eliminar o no los renglones en la vista. Regresaremos a este tema más adelante cuando abordemos cómo forzar o hacer cumplir restricciones.

En esta sección estudiamos las acciones que se deben seguir cuando se crean, leen, actualizan y eliminan instancias de vistas. Algunas de las acciones aquí descritas se pueden realizar automáticamente mediante los productos DBMS. Por ejemplo, Access tiene asistentes (wizards) que generarán un formato con Cliente y un camino autónomo —ya sea para los intereses relacionados con Transacciones o con Artista—. Los usuarios de este formato pueden crear, leer, actualizar y eliminar instancias de un Cliente y datos en uno de los caminos. Sin embargo, los asistentes de Access no generarán un formato que apoye ambos caminos en la vista de Cliente en la figura 10-3. Para que lo anterior sea posible, un programador necesitará escribir el código de un programa.

► FIGURA 10-6

Especificación de eliminaciones en cascada



► DISEÑO DE FORMAS

Como se mostró en la figura 10-1, la segunda función principal de una aplicación de la base de datos es generar materializaciones de vistas. En este capítulo analizaremos materializaciones de formatos y reportes. En el capítulo 14 abordaremos materializaciones de interproceso usando XML, y en el capítulo 17 estudiaremos materializaciones de cubos OLAP.

Una **forma** es un desplegado en pantalla que se usa para introducir y editar datos. Las formas de sólo lectura también se pueden usar para reportar datos, pero en la mayoría de los casos, cuando los programadores hablan de formas, se refieren a las que se usan para el ingreso y la edición de datos.

Algunas formas son fáciles de usar y dan como resultado pocos errores en el ingreso de datos. Otras parecen difíciles de manejar, artificiales, y es difícil usarlas sin que se creen errores. En esta sección analizamos e ilustramos varios principios sobre el diseño correcto de formas.

LA ESTRUCTURA DE LA FORMA DEBE REFLEJAR LA ESTRUCTURA DE LA VISTA

Primero, para que parezca natural y sea fácil de usar, *la estructura de una forma debe reflejar la estructura de la vista que materializa*. Observe la forma de la figura 10-7, la cual es una materialización de la vista Cliente en la figura 10-4(a).

La estructura de esta forma refleja la de la vista Cliente (Customer). Una sección de la forma tiene los datos básicos del cliente tales como Nombre, Teléfono y Dirección (Name, Phone and Address). La segunda sección muestra los intereses del cliente en Artista (Artist interests). Por último, la tercera sección lista las transacciones de compra del cliente. Los usuarios encontrarán esta forma fácil de usar debido a que los atributos están agrupados de manera tal que reflejan su comprensión sobre la estructura de los datos del cliente.

En general, cuando se diseña una forma toda la información de una relación independiente se debe colocar en una sección contigua de la forma. Suponiendo que la base de datos está en DK/NF, cada relación debe pertenecer a un tema específico, y el

► FIGURA 10-7

Materialización de una forma de vista de Cliente en la figura 10-4(a)

Artist	Title	Copy	Purchase Date	Sales Price
Matt Tobey	Lithograph One	10/75	7/15/94	\$4,300
Juan Mao	Portrait 14	5/250	2/5/95	\$2,500
Juan Mao	Awakening Child	1/1	11/22/97	\$17,950

usuario esperaría encontrar en un lugar los datos de ese tema. Por lo tanto, en la forma hay clientes, interés del artista y secciones de transacción.

Hay una excepción a esta regla: los atributos en la relación base de la vista (aquí, la relación CLIENTE) a veces no están colocados en forma contigua. Suponga, por ejemplo, que CLIENTE tiene un atributo independiente llamado TotaldeCompras. Si siguiéramos esta regla pondríamos TotaldeCompras en la primera sección de la forma. Pero para los usuarios sería más claro poner ese atributo al final, después de que se hayan listado todas las compras.

La forma en la figura 10-7 no es la única aceptable de esta vista. Por ejemplo, TRANSACTION se podría colocar antes de Artist Interests. Los datos básicos de CUSTOMER podrían ser reacomodados para que tuvieran una apariencia más horizontal. CustomerName, AreaCode y LocalNumber se podrían colocar en una columna, y Street, City, State y Zip en otra. Todas estas alternativas permiten la estructura de la forma para que refleje la estructura de los objetos implícitos.

LA SEMÁNTICA DE LA INFORMACIÓN DEBE SER GRÁFICAMENTE OBVIA

Otra característica de una forma bien diseñada es que la semántica de los datos es gráficamente obvia. Considere la sección de datos básicos del cliente de la forma CUSTOMER en la figura 10-7. Observe que existen rectángulos alrededor de AreaCode y LocalNumber y alrededor de Street, City, State y Zip.

Para entender por qué se hace esto, remitámonos nuevamente a la figura 10-3(a). El objeto semántico CUSTOMER (Cliente) tiene un atributo de grupo llamado Teléfono y otro Dirección. Puesto que ambos tienen una cardinalidad máxima de 1, en realidad no se requieren —desde el punto de vista del diseño relacional—. De hecho, no aparecen en las relaciones en la figura 10-3(c) o (d). El único propósito de estos atributos de grupo es asociar semánticamente CódigodeÁrea con NúmeroLocal, y Calle, Ciudad, Estado y CódigoPostal con algún otro.

El propósito de los rectángulos en la forma Customer Purchase Form es aclarar gráficamente estas asociaciones. La mayoría de los usuarios están de acuerdo con este arreglo. Debido a que saben que el número de teléfono consta de CódigodeÁrea y NúmeroLocal, les parece que la asociación gráfica de ambos es muy apropiada.

LA ESTRUCTURA DE LA FORMA DEBE FOMENTAR LA ACCIÓN APROPIADA

La estructura de las formas debe facilitar la ejecución de acciones apropiadas y dificultar la ejecución de acciones erróneas. Por ejemplo, el campo para ingresar State (Estado) en la forma de la figura 10-7 es pequeño. Obviamente, se supone que el usuario introduce sólo una abreviación de dos dígitos; un diseño más apropiado sólo permitiría ingresar dos dígitos; pero el mejor diseño presentaría los nombres de los estados en una lista desplegable y permitiría que el usuario seleccionara.

En la figura 10-7 algunos campos son blancos y otros grises. Las formas en esta aplicación han sido diseñadas para que el usuario sólo pueda ingresar datos en los espacios en blanco; los elementos de los datos en gris no pueden ser modificados. Así, el usuario no puede escribir en los campos Title o Copy en la sección TRANSACTION. En lugar de eso, para seleccionar o cambiar una obra, el usuario hace “clic” en la caja que está abajo de Artist. Cuando hace clic en la flecha, se despliega una lista de obras, como se muestra en la figura 10-8. Si el usuario quiere actualizar las obras o los datos de un artista, o agregar o eliminar datos de esa obra o dicho artista, debe utilizar una forma diferente.

La razón de este diseño es que cuando los usuarios están ingresando nuevas transacciones para los clientes, sólo deberían agregar los datos de la transacción; a saber, PurchaseDate (Fecha de Compra) y SalesPrice (Precio de Venta). Permitirles cambiar datos en ARTIST o WORK abriría la puerta a cambios accidentales o erróneos, que obstaculizaría la venta de una obra.

► FIGURA 10-8

Uso de la caja de lista desplegable

Artist	Title	Copy	Purchase Date	Sales Price
Mark Tobey	Lithograph One	110/75	7/15/94	\$4,300
Juan Miró	Poster 14	5/250	2/5/95	\$2,500
Juan Miró	Awakening Child	1/1	11/22/97	\$17,850
Jackson Pollock	Late Night City	2/25		
Juan Miró	Awakening Child	1/1		
Juan Miró	Poster 14	5/250		
Mark Tobey	Deepening Sky	1/1		
Mark Tobey	Lithograph One	14/75		
Mark Tobey	Lithograph One	10/75		
Pablo Picasso	Blue Nudes	7/100		
Salvador Dalí	Time Compression	5/1000		

Un diseño similar se usa para Artist Interests. Los usuarios sólo pueden elegir de una lista. Sin embargo, el caso no es tan inflexible. Los vendedores podrían querer registrar que un cliente se interesa en un artista que no forma parte de la base de datos de la galería. Si es así, la forma cambiaría para permitir que los usuarios agreguen otros nombres a la lista. Como justificación a este diseño de la figura 10-7, el dueño de la galería quizá quiera registrar únicamente a los artistas de interés con los que trabaja o a los cuales representará. Probablemente quiera tener control de qué nombres pueden aparecer.

FORMAS EN UN AMBIENTE GUI

Hay varias características peculiares de los sistemas GUI que pueden facilitar enormemente el uso de las aplicaciones de la base de datos.

CAJAS DE LISTAS DESPLEGABLES. Una **caja de lista desplegable** es un control GUI que presenta una lista de elementos entre los cuales puede elegir el usuario. Una propiedad del control determina si la lista es fija o si los usuarios pueden agregarle algo. La figura 10-9 muestra una versión diferente de Customer Purchase Form (Forma de Compra del Cliente) en la cual los usuarios pueden agregar Nombres de Artistas que no formen parte de la lista. Tras bambalinas, la aplicación se almacena en un nuevo renglón de ARTIST en la base de datos. Por supuesto, para que esto funcione sólo ArtistName (Nombre del Artista) puede ser un atributo requerido. Si hubiera otros campos requeridos, el DBMS rechazaría la inserción del nuevo renglón.

Las cajas de listas tienen muchas ventajas sobre las cajas de entrada de datos. Primero, para la gente es más fácil *reconocer* que *recopilar*. Por ejemplo, en la forma en la figura 10-8, es más fácil elegir de la lista el nombre de un artista, que recordar a todos los que forman parte de la base de datos. También es más fácil reconocer un nombre que escribirlo en forma correcta. Finalmente, si la caja de lista está preparada para desplegar sólo los valores presentes en la base de datos, el usuario no puede introducir errores de mecanografía. Por desgracia, el DBMS considerará como dos artistas diferentes a Juan Miró (con un espacio entre la *n* y la *M*) y a Juan Miró (con dos espacios entre la *n* y la *M*). Las cajas de listas fijas desplegables evitan estos errores.

► FIGURA 10-9

Caja de lista que permite nuevos valores

The screenshot shows a web form titled "View Ridge Gallery Customer Purchase Form". The form contains several input fields for customer information: "Customer Name" (Jackson, Elizabeth), "Phone" (Area Code: 206, Local Number: 294-6783), "Address" (Street: 123 - 4th Avenue, City: Seattle, State: WA, Zip: 98119-0977), and "Artist Interests". The "Artist Interests" field is a dropdown menu that is currently open, showing a list of artists: Juan Miro, Mark Tobey, Pablo Picasso, and Dennis Fings. Below the form, there is a table titled "TRANSACTION" with columns: Artist, Title, Copy, PurchaseDate, and SalesPrice. The table contains three rows of transaction data. At the bottom, there are record navigation controls showing "Record: 1 of 2" and "Form View".

Artist	Title	Copy	PurchaseDate	SalesPrice
Mark Tobey	Lithograph One	110/75	7/15/94	\$4,300
Juan Miro	Poster 14	5/250	2/5/98	\$2,500
Juan Miro	Awakening Child	1/1	11/22/97	\$17,850

BOTONES DE OPCIONES Y GRUPOS. Un **botón de opción**, o un **botón de radio**, es un dispositivo de desplegado que permite a los usuarios seleccionar una condición alternativa o estado a partir de una lista de posibilidades. Por ejemplo, suponga que la galería decide que necesita registrar el estado fiscal de sus clientes. Suponga que éste tiene dos posibilidades: Taxable and Non-taxable (gravable y no gravable), y que ambas son mutuamente excluyentes. Se podría usar un grupo de opción como el que se muestra en la figura 10-10 para recopilar esos datos. Debido a la forma en la que el grupo de opción funciona, si el usuario selecciona Taxable, entonces Non-taxable automáticamente

► FIGURA 10-10

Uso de botones de grupo y de opción

The screenshot shows the same web form as Figure 10-9, but with the "Tax Status" field added. It contains two radio buttons: "Taxable" and "Non-taxable". The "Taxable" radio button is selected. The "Artist Interests" dropdown menu is still open, showing the same list of artists. The "TRANSACTION" table and record navigation controls are also present at the bottom.

Artist	Title	Copy	PurchaseDate	SalesPrice
Mark Tobey	Lithograph One	110/75	7/15/94	\$4,300
Juan Miro	Poster 14	5/250	2/5/98	\$2,500
Juan Miro	Awakening Child	1/1	11/22/97	\$17,850

mente deja de ser una selección. Si el usuario opta por Nogravable, entonces Graversable se elimina automáticamente.

Detrás de la forma, el programa de aplicación debe almacenar información en una columna de la tabla que representa el botón de radio seleccionado. Para este ejemplo, la columna se llama TaxStatus (EstadoFiscal). Se usa una de las dos maneras posibles para almacenar los datos del botón de opción. Una es almacenar un número entero del 1 al número de botones. En este ejemplo, uno de los valores 1 o 2 sería almacenado. La segunda opción es almacenar un valor de texto que describa la opción elegida. Las posibilidades son Sí o No.

CAJAS DE VERIFICACIÓN (CHECK BOXES). Las **cajas de verificación** son similares a los botones de opción, excepto que las alternativas en un grupo de cajas de verificación no son mutuamente excluyentes —se puede elegir más de una alternativa—. Suponga, por ejemplo, que la galería quiere registrar el tipo, o los tipos, de material artístico que les interesan a los clientes. Los posibles tipos son Lithographs, Oils, Pastels, and Photographs (Litografías, Óleos, Pasteles y Fotografías), y se muestran en una serie de cajas de verificación en la versión de la forma CUSTOMER en la figura 10-11. El usuario selecciona o marca las cajas apropiadas.

Hay varias formas de representar cajas de verificación en las relaciones. Una manera común y sencilla es definir una columna de valores booleanos para cada elemento en el grupo de cajas de verificación. El valor de cada una de estas columnas es binario; esto es, puede ser 1 o 0 para representar Sí o No. Hay otras posibilidades tales como codificar bits en un byte, aunque éstos no son importantes para nuestro análisis.

En la figura 10-11 observe los cuadros grises en Pastels y Ceramics. Ese gris significa que los valores de esas opciones son nulos. Los usuarios pueden interpretarlos como que este cliente está interesado en Lithographs y Photography y no en Oils. Su interés en Pastels y Ceramics es ambiguo (véase el análisis de nulos en el capítulo 6). Si es importante registrar el estado nulo, entonces el diseño de la base de datos debe permitir tres valores para cada caja de verificación: verificado, no verificado y nulo. Aunque el análisis también corresponde a los grupos con botones de opción.

FORMAS WEB. Las formas usadas en el explorador son GUI y todos los comentarios anteriores también se aplican. Sin embargo, las formas Web tienen una característica que las de Windows no tienen: hiperconexiones, las cuales son vínculos especial-

► FIGURA 10-11

Uso de cajas de verificación

The screenshot shows a software window titled "View Ridge Gallery Customer Purchase Form". It contains several input fields for customer data: "Customer Name" (Jackson, Elizabeth), "Phone", "AreaCode" (706), "LocalNumber" (254-5783), "Address" (123 - 4th Avenue), "City" (Seattle), "State" (WA), and "Zip" (98119-9877). There are also "Interest Area(s)" check boxes for Lithographs, Oils, Pastels, Photography, and Ceramics. A "Artist Interests" list includes Mail Tobey, Juan Miro, Pablo Picasso, and Dennis Fong. Below this is a "TRANSACTION" table with columns for Artist, Title, Copy, Purchase Date, and Sales Price. The table lists three transactions: Mail Tobey (Lithograph One, 11/79, 7/15/94, \$4,500), Juan Miro (Pastel 14, 5/250, 2/5/98, \$2,500), and Juan Miro (Smiling Child, 1/1, 11/22/97, \$17,850). The form also includes record navigation controls at the bottom.

Artist	Title	Copy	Purchase Date	Sales Price
Mail Tobey	Lithograph One	11/79	7/15/94	\$4,500
Juan Miro	Pastel 14	5/250	2/5/98	\$2,500
Juan Miro	Smiling Child	1/1	11/22/97	\$17,850

mente útiles para materializar conexiones con otros objetos semánticos o entidades fuertes. Por ejemplo, en la forma de la figura 10-7, las conexiones con artistas de interés se pueden reemplazar por hiperconexiones para que cuando el usuario haga clic en éstas sea llevado a una forma ARTIST con los datos apropiados. Se puede hacer lo mismo con los enlaces a WORKs.

MOVIMIENTO DEL CURSOR Y TECLAS DE USO EXCLUSIVO

Otro aspecto sobre el diseño de formas es la acción del cursor, el cual se debe mover a través de la forma de manera fácil y natural. Esto generalmente significa que el cursor sigue el patrón de procesamiento del usuario conforme éste lee los documentos de ingreso de los datos fuente. Si se usan formas para introducir datos por teléfono, el cursor debe controlar el ritmo de la conversación. En este caso, este movimiento progresará de tal manera que el cliente lo encuentre natural y conveniente.

El movimiento del cursor es especialmente importante durante y después de una condición de excepción. Suponga que usando la forma en la figura 10-7 se comete un error —quizás al introducir un código inválido de estado—. La forma se debe procesar para que el cursor se mueva a una posición lógica. Por ejemplo, la aplicación puede desplegar una caja de lista de valores de estado disponibles y colocar al cursor en una posición lógica en la lista —quizás en el primer estado que empieza con la primera letra, la cual introdujo el usuario—. Cuando se selecciona el State (estado), el cursor debe regresar a Zip (CódigoPostal), el siguiente espacio adecuado de la forma.

Las acciones de las teclas de propósito especial, tales como ESC y las llaves de función, deberían ser *consistentes* y *exclusivas* en su uso. Si se usa ESC para salir de formas, se le debe usar únicamente para ese propósito (excepto cuando se trate de acciones que sean lógicamente equivalentes para salir de las formas). Las acciones de las teclas deben ser consistentes en toda la aplicación. Esto es, si se usa ESC para salir de una forma, también se debe usar para salir de todas las formas. Si se usa Ctrl-D para borrar datos en una forma, se debe usar para borrar datos en todas las formas. De otro modo, los hábitos que se adquieren en una parte de la aplicación se deben olvidar o aprender nuevamente en otras partes de la aplicación. Esto, además de ser un desperdicio de tiempo, es frustrante, exasperante y ocasiona errores. Aunque estos comentarios pueden parecer obvios, el cuidado con este tipo de detalles es lo que hace a una forma fácil y conveniente de usar.

► DISEÑO DE REPORTE

El tema del diseño de reportes, más que el diseño de formas, se ha analizado ampliamente en textos de desarrollo de aplicaciones. No duplicaremos, ni siquiera trataremos de resumir esos análisis aquí, sino más bien observaremos varios conceptos directamente relacionados con la noción de un reporte, tal como una materialización de la vista de una base de datos.

ESTRUCTURA DE REPORTE

Los principios para el diseño de un reporte eficaz son similares a los de diseño de formas. Igual que con las formas, *la estructura de un reporte debe reflejar la estructura de la vista implícita*. Esto significa que los datos de una tabla generalmente deben localizarse en un grupo contiguo al reporte. Al igual que las formas, una excepción a esta regla es que la relación básica de la vista (por ejemplo, la relación CLIENTE de la vista Cliente) puede estar por separado en el reporte. Los grupos de atributos, como teléfono, también se deben localizar juntos y distinguirse de alguna manera.

La figura 10-12 muestra un ejemplo de reporte de la galería View Ridge que lista los datos para cada obra de arte y las transacciones para cada una; calcula el margen total de utilidad por trabajo y por artista, así como el gran total.

► FIGURA 10-12

Reporte de lista de ventas

Lista de ventas 15-Nov-01					
NombredelArtista	Título	Copia			
Dennis Frings	South Toward Emerald Sea	106/195			
Fecha- Adquisición	Preciode- Adquisición	Fecha- deVenta	Vendido a	Preciode- Venta	MargenTotal- deUtilidad
4/17/1986	\$750.00	5/3/1986	Heller, Max	\$1,000.00	\$250.00
3/15/2000	\$1,200.00	5/11/2000	Jackson, Elizabeth	\$1,800.00	\$600.00
MargenTotal de South Toward Emerald Sea, Copia 106/195					\$850.00
Margen total de utilidad en el caso de Dennis Frings					\$850.00
Mark Tobey	Patterns III	27/95			
Fecha- Adquisición	Preciode- Adquisición	Fecha- deVenta	Vendido a	Preciode- Venta	MargenTotal- deUtilidad
7/3/1971	\$7,500.00	9/11/1971	Cooper, Tom	\$10,000.00	\$2,500.00
1/4/1986	\$11,500.00	3/18/1986	Jackson, Elizabeth	\$15,000.00	\$3,500.00
9/11/1999	\$17,000.00	10/17/1999	Cooper, Tom	\$21,000.00	\$4,000.00
Margen Total para Patrones III, Copia 27/95					\$10,000.00
Mark Tobey	Rhythm	2/75			
Fecha- Adquisición	Preciode- Adquisición	Fecha- deVenta	Vendido a	Preciode- Venta	MargenTotal- deUtilidad
4/8/2001	\$17,000.00	7/14/2001	Heller, Max	\$27,000.00	\$10,000.00
Margen total de utilidad en el caso de Rhythm, Copia 2/75					\$10,000.00
Margen total de utilidad en el caso de Mark Tobey					\$20,000.00
Gran Total:					\$20,850.00

La estructura del reporte en la figura 10-12 refleja la estructura del objeto TRABAJO. La sección para cada obra comienza con el nombre de ésta, el cual incluye al artista, el título y la copia. Sigue una sección de renglones de repetición que muestra las transacciones de la obra. Dentro de cada sección el nombre del cliente se ha extraído de la tabla CLIENTE.

Hay que estar conscientes de que con la mayoría de los editores de reportes es difícil construir un reporte que siga más de un camino de valores múltiples a través del esquema de la base de datos. El reporte en la figura 10-12 es una materialización de una vista de ARTISTA que sigue el camino desde ARTISTA hasta TRABAJO y luego a TRANSACCIÓN. El diagrama de relación (relationship) de la figura 10-3(e) muestra otra pista —que va de la tabla INTERÉS_ARTÍSTICO_CLIENTE (CUSTOMER_ARTIST_INT) a la de CLIENTES (CUSTOMER) que están interesados en un artista específico—. Con la mayoría de los editores de reportes será difícil construir un reporte que muestre ambos caminos.

Los reportes con frecuencia tienen atributos de cálculo de datos que no son parte de la vista implícita ni están almacenados en la base de datos. El reporte en la figura 10-12 tiene cálculos para MargenTotaldeUtilidad, margen total de utilidad por Trabajo, margen total de utilidad por Artista y Gran Total. Todos estos valores se calculan al momento en que se produce el reporte.

Si bien estos valores calculados se podrían almacenar en la base de datos, hacerlo casi nunca es una buena idea porque los valores usados para el cálculo pueden cambiar.

Si, por ejemplo, un usuario modifica el Precio de Venta de una transacción en particular y no calcula nuevamente el Margen Total de Utilidad y los totales basados en éste, los valores almacenados serán erróneos. Sin embargo, si se hacen todos los cálculos necesarios mientras se procesan las transacciones actualizadas probablemente dará como resultado un procesamiento inaceptablemente lento. Por lo tanto, los totales como este por lo general se calculan mejor sobre la marcha. Así, las fórmulas para calcularlos se consideran parte de la materialización del reporte y no parte de la vista implícita.

OBJETOS IMPLÍCITOS

Considere la solicitud: “Imprimir todos los ARTISTAS ordenados por margen total”. A primera vista, parece ser una solicitud de impresión de un reporte acerca del objeto ARTISTA. Sin embargo, las palabras *ordenados por*, indican que se debe considerar más de un ARTISTA. De hecho, esta solicitud no está basada en el objeto ARTISTA, sino en el objeto CONJUNTO DE TODOS LOS ARTISTAS. El reporte en la figura 10-12 muestra los datos de múltiples ARTISTAS y de hecho está basado en el objeto CONJUNTO DE TODOS LOS ARTISTAS, y no en el objeto ARTISTA.

La mente humana cambia con tanta rapidez del objeto *OBJETO-A* al objeto *CONJUNTO DE TODOS LOS OBJETOS-A* que normalmente ni siquiera sabemos que ha ocurrido un cambio. Sin embargo, cuando se desarrollan las aplicaciones de la base de datos es importante observar este cambio, porque la aplicación necesita comportarse en forma diferente cuando esto sucede. Considere tres maneras en las que un ordenamiento puede cambiar la naturaleza del objeto base: (1) ordenar por identificador de objeto; (2) ordenar por columnas no identificadoras, y columnas no objeto; y (3) ordenar por atributos contenidos en los atributos de objeto.

ORDENAMIENTO POR IDENTIFICADOR DE OBJETO. Si el reporte se va a ordenar por un atributo que es un identificador de objeto, el objeto verdadero es un conjunto de esos objetos. Así, un reporte de ARTISTA ordenado por Nombre del Artista se refiere al objeto CONJUNTO DE TODOS LOS ARTISTAS. Para la mayoría de los productos de escritura de reportes DBMS, un reporte sobre el objeto CONJUNTO DE TODOS LOS X no es más difícil de producir que un reporte acerca del objeto X. Sin embargo, es importante saber que ha ocurrido un cambio en el tipo de objetos.

ORDENAMIENTO POR COLUMNAS NO IDENTIFICADORAS, Y COLUMNAS NO OBJETO. Cuando un usuario necesita un reporte ordenado por un atributo que es un identificador del objeto, es muy probable que en su mente el objeto verdadero sea un tipo de objeto totalmente diferente. Por ejemplo, el usuario desea producir un reporte acerca de ARTISTA ordenado por Nacionalidad. Dicho reporte en realidad es una materialización del objeto NACIONALIDAD, y no del objeto ARTISTA. De manera similar, si el usuario solicita un reporte sobre CLIENTE ordenado por Código de Área, el reporte en realidad está basado en un objeto llamado REGIÓN-TELEFÓNICA, o algún objeto similar. La figura 10-13 muestra un ejemplo del objeto REGIÓN-TELEFÓNICA.

Los objetos tales como NACIONALIDAD y REGIÓN-TELEFÓNICA son **objetos implícitos**; esto es, su existencia se puede inferir por el hecho de que el usuario solicitó tal reporte. Si tiene sentido para el usuario solicitar algo como un valor de ordenamiento, entonces ese algo debe ser un objeto en su mente, sin importar que esté modelado en la base de datos o no.

► FIGURA 10-13

Objeto REGIÓN
TELEFÓNICA



ORDENAMIENTO POR ATRIBUTOS CONTENIDOS EN ATRIBUTOS OBJETO.

La tercera manera en la cual se pueden ordenar los reportes es por atributos contenidos en los atributos del objeto. Por ejemplo, el usuario podría solicitar un reporte acerca de los TRABAJOS ordenados por FechadeNacimiento de ARTISTA, el cual es un atributo del objeto TRABAJO, y la Fecha de Nacimiento es un atributo contenido en ARTISTA. En este caso, el usuario en realidad está solicitando un reporte acerca de un objeto implícito (digamos, TIEMPO o PERIODO ARTÍSTICO) que contiene muchos ARTISTAS, cada uno de los cuales incluye muchos TRABAJOS en la galería.

Comprender este cambio en los objetos puede facilitar la tarea de desarrollar el reporte. Proceder como si TRABAJO fuera el objeto básico del reporte lo haría lógicamente artificial. Si TRABAJO se considera la base, se deben crear las materializaciones de los objetos TRABAJO que incluyen ARTISTA y FechadeNacimiento para todos los objetos TRABAJO, almacenados en el disco, y luego ordenarlos por FechadeNacimiento. Por otra parte, si este reporte se refiere a un objeto implícito que contiene ARTISTA, el cual a su vez abarca TRABAJO, entonces se pueden crear los objetos ARTISTA y ordenarlos por FechadeNacimiento, y tratar a los objetos TRABAJO como renglones de valores múltiples en cada objeto ARTISTA.

► CUMPLIMIENTO DE RESTRICCIONES

Como se observa en la figura 10-1, la tercera función principal de una aplicación de la base de datos es el cumplimiento de restricciones. En muchos casos, el DBMS es más capaz de imponer restricciones que la aplicación, y ésta no es estrictamente una función del programa de aplicación. Sin embargo, nuestra preocupación es describir los tipos de restricciones y cómo se pueden cumplir, sin importar el código de la aplicación o el DBMS que las haga cumplir.

En primer lugar, la razón de que el DBMS sea con frecuencia el mejor lugar para hacer cumplir restricciones es que constituye el punto central a través del cual deben pasar todos los cambios de información. Sin tomar en cuenta la fuente de un cambio de datos (forma, otro programa, importación de un bloque de datos) el DBMS tendrá la oportunidad de examinar y rechazar el cambio, si es necesario. Además, ciertas reglas (la unicidad es una de ellas) pueden requerir la revisión de todos los renglones de la tabla; el DBMS tiene más capacidad para ejecutar esta función que una aplicación. Además, si el DBMS impone una regla, entonces se necesita codificarla sólo una vez. Si las aplicaciones imponen una regla, entonces se necesitará codificar cada nueva aplicación. Esto es una pérdida de tiempo y genera la posibilidad de que los programas de aplicación impongan reglas de manera inconsistente.

Cabe aclarar que no todos los productos DBMS tienen las características y funciones necesarias para hacer cumplir restricciones. A veces es mucho más difícil escribir e instalar el código de aplicación de restricciones en productos DBMS que en un código de aplicación. Además, en algunas arquitecturas (el servidor de transacción de Microsoft, o Microsoft Transaction Server, es una de ellas) el procesamiento es más eficiente si se elimina la verificación de restricciones del servidor de datos. También, hay algunas restricciones que forman parte de la aplicación. Por ejemplo, un usuario de una forma en particular no se puede permitir ciertas acciones o introducir ciertos datos. En este caso, la aplicación tiene más capacidad para imponer la restricción.

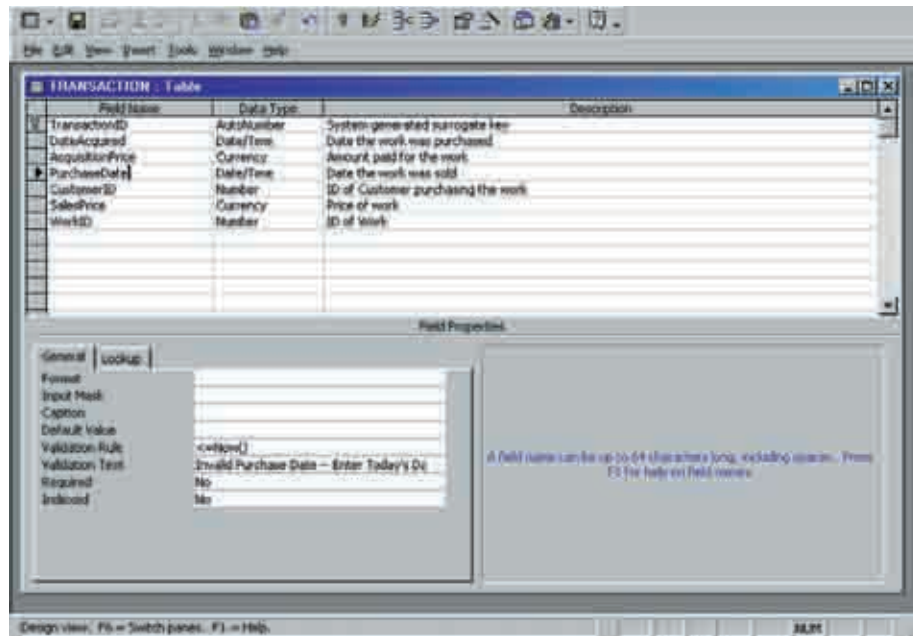
La meta que usted tiene es entender los tipos de restricciones y los medios por los cuales se aplican. Cuando sea posible, hay que imponerlas en el DBMS, cuando no, en la aplicación. Ahora consideraremos cuatro tipos de restricción: dominio, unicidad, integridad referencial y reglas del negocio.

RESTRICCIONES DE DOMINIO

Como establecimos en el capítulo 4, un dominio es el conjunto de valores que un atributo puede tener. Las definiciones de dominio tienen un componente semántico

► FIGURA 10-14

Creación de una restricción de dominio con Access



(el nombre de un artista) y uno físico (texto 25, alfabético). En general, no es posible imponer el componente semántico mediante un proceso automatizado. La industria de la computación aún no está en una etapa de desarrollo tal en que los procesos automatizados puedan determinar que 3M es el nombre de una compañía y no el de un artista. Por lo tanto, actualmente estamos principalmente preocupados por la aplicación de la parte física de una definición de dominio.

Las restricciones de dominio surgen del modelo de datos, como se describió en el capítulo 4. La especificidad de las restricciones de dominio varían ampliamente. En la base de datos de la galería View Ridge, *Artista.Nombre* puede ser cualquier conjunto de 25 o menos caracteres de texto. Sin embargo, un dominio como el de *Copia*, es más específico. El formato usado por la galería es *nn/mm*, donde *nn* es un número particular de copias y *mm* es el número total de copias hechas. Así, *5/100* significa la quinta copia de 100. Claramente, un valor como *105/100* no tiene sentido. Por lo tanto, se debe desarrollar el código de validación para asegurar que se siga este formato.

La figura 10-14 muestra un ejemplo de restricción de dominio en Access. La restricción es que *PurchaseDate* (Fecha de compra) debe ser menor o igual a la fecha del reloj de la computadora en la hora y la fecha en que se introdujo la compra. Observe el renglón etiquetado *Validation Rule* (regla de validación). La expresión `<= Now()` define la regla. Si el usuario intenta introducir datos que violen esta regla, se genera una caja de mensaje usando el texto que se introdujo en el siguiente renglón (texto de validación, o *Validation Text*). La figura 10-15 muestra qué ocurre cuando el usuario intenta ingresar la fecha 10/15/99 cuando la computadora tiene una fecha anterior. En ese caso Access rechaza los datos.

Otro tipo de restricción es si los valores son o no requeridos. Estrictamente hablando, el requerimiento de que se proporcione un valor es una restricción de la tabla, más que una restricción de dominio. Un dominio es un conjunto de valores; si un valor se requiere o no es una pregunta que surge cuando el dominio aparece en una tabla.

La modelación de datos debe indicar si se requieren los valores del atributo. En el modelo de objeto semántico si la cardinalidad mínima de un atributo es 1, entonces se requiere ese atributo. Para imponer esta restricción con Access todo lo que se necesita es establecer la propiedad *Required* de una columna en *Yes*. Se hizo lo mismo con *Name* en la definición de la tabla *ARTIST* en la figura 10-16. Como verá, se usan diferentes medios con otros productos DBMS.

Las restricciones de valor que se requieren son importantes porque eliminan la posibilidad de valores nulos. Otra forma de hacer esto es definir un valor inicial que el

► FIGURA 10-15

Resultado de violar una regla de validación

DBMS asigne a la columna cuando se cree el renglón. Con Access esto se hace colocando un valor o expresión en la propiedad Default Value (Valor Predeterminado). Véanse la figura 10-16 y el análisis de los valores nulos en el capítulo 6.

RESTRICCIONES DE UNICIDAD

La unicidad es el segundo tipo de restricción. Como se estableció, las restricciones de este tipo las aplica mejor el DBMS debido a que puede crear estructuras de datos para efectuar más rápido la verificación de la unicidad. Véase el apéndice A para obtener una descripción de la manera en que se pueden usar los índices con este propósito.

► FIGURA 10-16

Definición de la unicidad para Nombre en ARTISTA (Name in ARTIST)

La figura 10-16 muestra cómo Name en ARTIST puede ser definido como único en Access. Observe que Indizar se ha puesto en Yes (No Duplicates). Con esa precisión Access se asegurará de que no se dupliquen los nombres de los artistas que ingresen desde cualquier fuente.

RESTRICCIONES DE RELACIÓN

Hay dos tipos de restricciones de relación: **integridad referencial** y **cardinalidad de relación**. La integridad referencial consiste en las restricciones a los valores de las llaves externas. Por ejemplo, las restricciones que surgen de la normalización cuando una relación individual se divide en dos, y eso ocurre entre entidades fuertes y débiles. Las restricciones de cardinalidad de relación ocurren debido a los valores de cardinalidad máxima y mínima en las relaciones. En esta sección consideramos ambos tipos.

RESTRICCIONES DE INTEGRIDAD REFERENCIAL. Todas las restricciones de integridad referencial son restricciones a valores de llaves externas. Considere la primera restricción de integridad referencial en la figura 10-3(d): ArtistaID en TRABAJO debe existir en ArtistaID en ARTISTA. ArtistaID en TRABAJO es una llave externa y la restricción sólo significa que su valor debe concordar con un valor en ArtistaID en la tabla ARTISTA, como ya se explicó.

Para que no exista violación alguna a la composición de la llave externa es necesario que se cumplan las siguientes condiciones: que todas las aplicaciones estén codificadas en forma correcta, que todas las transacciones se procesen adecuadamente y que ningún usuario borre por accidente un artista sin haber eliminado lo que se relaciona con TRABAJO (el administrador debe borrar cualquier TRANSACCIÓN relacionada, conforme a lo que se estipula en la segunda condición y en eliminar todo CLIENTE relacionado, según lo que se establece en la tercera condición). Después de todo, la base de datos comienza sin problemas de integridad; cuando éstos ocurren es que fueron introducidos.

Lo anterior, por supuesto, es esperar demasiado. Los errores suceden; por lo tanto, los productos DBMS proporcionan la facilidad de que estas restricciones de la llave externa las pueda definir e imponer el DBMS. Esto significa que el DBMS no permitirá ninguna inserción, eliminación, o modificación de un valor de llave externa que ocasione una violación a la restricción de una llave externa. Por ejemplo, observe la figura 10-20, donde la marca frente a la opción Enforce Referential Integrity (Imponer Integridad Referencial) significa que se le ha pedido a Access que cumpla la restricción de que EmployeeName (NombredeEmpleado) en:

EMPLOYEE_PhoneNumber (EMPLEADO_Número de Teléfono) exista en Name (Nombre) en EMPLOYEE (EMPLEADO)

Facilidades similares existen para todos los productos DBMS. Además, los productos del servidor DBMS tales como los ORACLE y SQL Server proporcionan utilidades que pueden estar corriendo periódicamente para asegurar que ninguna restricción de llave externa haya sido permitida en la base de datos debido a transacciones parcialmente terminadas, datos importados, u otros misterios del universo. Estas utilidades corren normalmente como parte del plan de mantenimiento de la base de datos (véase el capítulo 11).

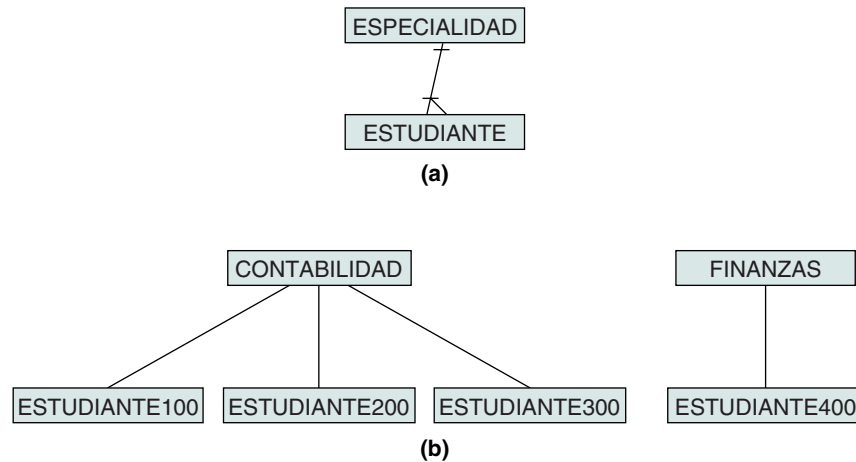
RESTRICCIONES DE CARDINALIDAD DE LA RELACIÓN. Las restricciones de cardinalidad de la relación surgen del establecimiento de la cardinalidad en los atributos de conexión del objeto. Por ejemplo, en la figura 10-3(a) la conexión CLIENTE en TRABAJO. La transacción tiene cardinalidad de 1:1; por lo tanto, una Transacción debe tener una conexión con CLIENTE.

En general, estas restricciones surgen de dos fuentes: establecimiento de no cero en la cardinalidad mínima, o asignaciones de cardinalidad máxima que no son 1 ni N. Así, las cardinalidades de 1:1, 1:N, 2:N ocasionarán que las restricciones de cardinalidad se presenten, como lo harán las cardinalidades de 0:3, 1:4, y 2:4. Con una excepción, estas restricciones se deben imponer mediante un código de aplicación.

La excepción son las cardinalidades 1:1 en el lado hijo de relaciones 1:N. En este caso, la restricción se puede imponer haciendo obligatoria la llave externa. Así, la res-

► FIGURA 10-17

Ejemplo de una restricción obligatoria-a-obligatoria:
(a) muestra de relación obligatoria-a-obligatoria, y
(b) muestra de datos para ésta



tricción 1.1 de CLIENTE en TRABAJO.Transacción se puede imponer haciendo ClientID obligatoria en la tabla TRANSACCIÓN.

Además de esta salvedad, el programador de la base de datos debe escribir un código para imponer restricciones de cardinalidad. Dicho código se puede colocar en los procedimientos almacenados (stored procedures) para que la invoque el DBMS cuando se hagan cambios de relación, se puede poner en programas de aplicación, o llamarlo durante ciertos eventos de forma, tal como Antesde la Actualización (BeforeUpdate), que analizaremos más adelante.

Para simplificar el análisis sólo consideraremos restricciones 1.1 y 1.N. La lógica para las otras restricciones es una extensión directa de la que se presenta aquí.

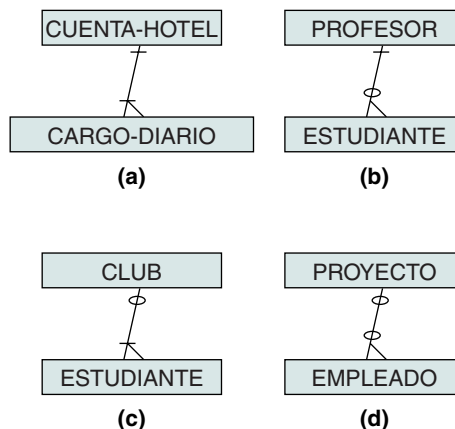
La figura 10-17 describe la relación entre las relaciones ESPECIALIDAD y ESTUDIANTE. Como se muestra en la figura 10-17(a), una ESPECIALIDAD debe tener cuando menos un ESTUDIANTE y un ESTUDIANTE debe tener exactamente una ESPECIALIDAD. Cuando los usuarios actualizan cualquiera de estas relaciones, se debe invocar el código para imponer la restricción. Por ejemplo, en la figura 10-17(b) si un usuario intenta eliminar el renglón de Estudiante 400, el código debe rechazar esa solicitud. Si ésta fuera permitida, el renglón para FINANZAS no tendría un renglón hijo, y la restricción obligatoria sería violada. De manera similar, una nueva ESPECIALIDAD, digamos BIOLOGÍA, no se puede agregar hasta que haya un estudiante que se esté especializando en esa materia.

Un renglón que existe en forma inadecuada sin un padre o hijo con frecuencia se llama **fragmento**, y a los renglones hijos que existen sin un padre obligatorio a veces se les llama **huérfanos**. Una de las funciones de un programa de aplicación es evitar la creación de fragmentos y huérfanos.

Los medios para evitar los fragmentos dependen del tipo de restricción. La figura 10-18 muestra ejemplos de las cuatro restricciones posibles en las relaciones 1:N: obli-

► FIGURA 10-18

Ejemplo de los cuatro tipos de restricciones:
(a) restricción obligatoria a obligatoria (M-M);
(b) restricción obligatoria a opcional (M-O); (c) restricción opcional a obligatoria (O-M), y
(d) restricción opcional a opcional (O-O)



gatoria a obligatoria (M-M); obligatoria a opcional (M-O); opcional a obligatorio (O-M), y opcional a opcional (O-O). Estas restricciones se muestran en las relaciones uno a muchos, pero los mismos cuatro tipos se aplican también a otras.

Se pueden violar las restricciones cuando hay cambios en los atributos de la llave. Por ejemplo, en la figura 10-17(b), cambiar la especialidad de Estudiante 300 de CONTABILIDAD a FINANZAS reasigna a ese estudiante al departamento de finanzas. Aunque esto produce un cambio de padre, no viola la restricción.

Sin embargo, dicha violación existirá si la especialidad de estudiante 400 se cambia a CONTABILIDAD. Cuando se hace esto, FINANZAS ya no tiene ningún estudiante, y también se viola la restricción M-M entre ESPECIALIDAD y ESTUDIANTE.

La figura 10-19 presenta reglas para evitar fragmentos en cada uno de estos tipos de restricciones. La figura 10-19(a) muestra acciones sobre el renglón padre, y la figura 10-19(b), acciones en los renglones hijos. Como estas figuras lo indican, las posibles acciones son insertar nuevos renglones, modificar los datos llave y eliminar renglones. La

► FIGURA 10-19

Reglas para evitar fragmentos: (a) condiciones para permitir cambios en los registros padres, y (b) condiciones para permitir cambios en los registros hijos

		Acción propuesta en padre		
		Insertar	Modificar (llave)	Eliminar
Tipo de relación	M-M	Crear cuando menos un hijo	Cambiar llaves coincidentes de todos los hijos	Eliminar todos los hijos o reasignar todos los hijos
	M-O	OK	Cambiar llaves coincidentes de todos los hijos	Eliminar todos los hijos o reasignar todos los hijos
	O-M	Insertar nuevo hijo o Existe hijo apropiado	Cambiar la llave de cuando menos un hijo o Existe hijo apropiado	OK
	O-O	OK	OK	OK

(a)

		Acción propuesta en hijo		
		Insertar	Modificar (llave)	Eliminar
Tipo de relación	M-M	Existe padre o Crear padre	Existe padre con un nuevo valor (o crear uno) y Existe hermano	Existe hermano
	M-O	Existe padre o Crear padre	Existe padre con un nuevo valor o Crear padre	OK
	O-M	OK	Existe hermano	Existe hermano
	O-O	OK	OK	OK

(b)

figura 10-19 lista las reglas para relaciones uno a muchos; las reglas para relaciones uno a uno son similares.

RESTRICCIONES PARA ACTUALIZAR LOS RENGLONES PADRE. El primer renglón de la figura 10-19(a) se refiere a restricciones M-M. Sólo se puede insertar un nuevo renglón padre si al mismo tiempo se creó cuando menos un renglón hijo, lo cual se puede hacer insertando un nuevo renglón hijo o reasignando un hijo de un padre diferente (no obstante, esta última acción puede ocasionar en sí misma una violación a la restricción).

En una relación M-M sólo se permite un cambio de la llave de un padre si también se modifican los nuevos valores correspondientes a la llave externa en los renglones hijo (es posible reasignar todos los hijos a otro padre y después crear cuando menos un nuevo hijo para el padre, pero esto se hace muy rara vez). Así, se permite cambiar la factura en CUENTA-HOTEL siempre y cuando también se cambie Factura en todos los renglones CARGO-DIARIO. Observe que si se usan las llaves sustitutas esta acción nunca ocurrirá.

Por último, un padre de una relación M-M se puede eliminar siempre y cuando todos los hijos también sean eliminados o reasignados.

En lo que se refiere a las restricciones M-O, se puede agregar un nuevo padre sin ninguna restricción, debido a que no necesitan tener hijos. Para la relación en la figura 10-18(b) se puede agregar un nuevo renglón PROFESOR sin restricción. Sin embargo, sólo se permite un cambio en el valor de la llave padre si los valores correspondientes en los renglones hijos también se cambian. Si un PROFESOR en la relación de la figura 10-18(b) cambia su llave, también se debe alterar el valor del Asesor en todos los renglones de los estudiantes que asesora el profesor.

Finalmente, en una relación con una restricción M-O, el renglón padre se puede suprimir sólo si todos los hijos son eliminados o reasignados. Para la relación PROFESOR-ESTUDIANTE, probablemente todos los renglones de estudiante serían reasignados.

Para las restricciones O-M, sólo puede insertarse un padre si al mismo tiempo se agrega cuando menos un hijo, o si éste ya existe. Por ejemplo, para la relación O-M entre CLUB y ESTUDIANTE de la figura 10-18(c) se puede agregar un nuevo club sólo si se puede crear un renglón ESTUDIANTE (ya sea agregando un nuevo estudiante, o cambiando el valor de CLUB en un ESTUDIANTE). Alternativamente puede existir un renglón adecuado de estudiante.

De manera similar, la llave del padre en una relación O-M sólo se puede cambiar si se crea un hijo o si ya existe un renglón hijo adecuado. Esto es, el Club de Esquí puede cambiar su nombre por el de Buceo sólo si cuando menos un esquiador está dispuesto a unirse a Buceo, o si un estudiante se ha inscrito ya en Buceo. No hay restricciones para la eliminación de un renglón padre en una relación O-M.

En la figura 10-18(d) se muestra el último tipo de restricción de relación: O-O. No existe ninguna restricción a las actualizaciones en los renglones de una relación O-O. Los renglones PROYECTO y EMPLEADO se pueden actualizar conforme sea necesario.

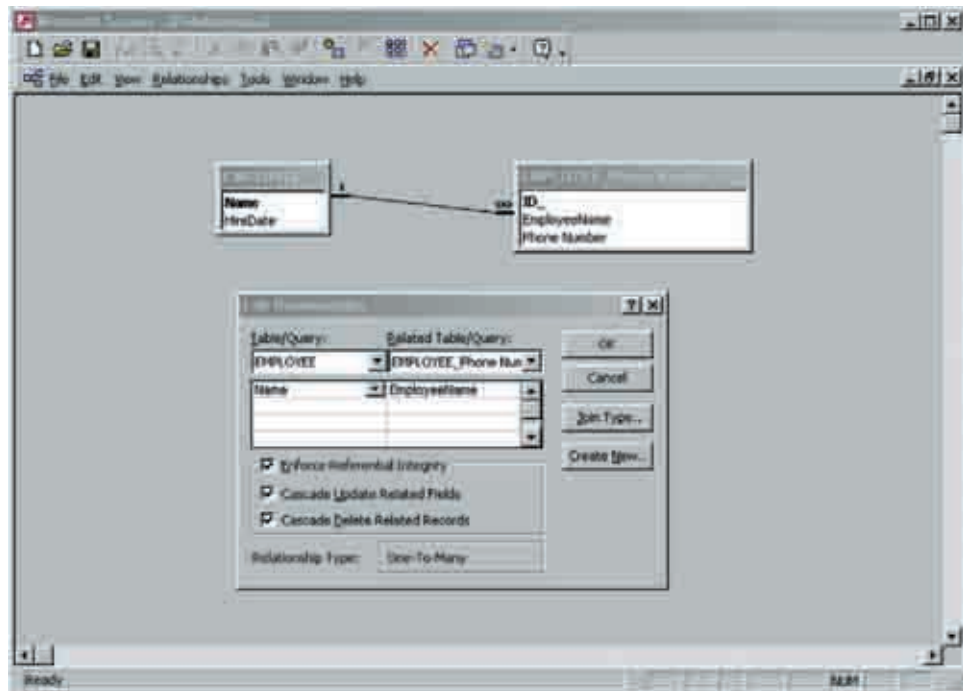
RESTRICCIONES A LA ACTUALIZACIÓN DE RENGLONES HIJO. Las reglas para evitar fragmentos cuando se actualizan renglones hijo se muestran en la figura 10-19(b) y son similares a las de la figura 10-19(a). La única diferencia notable es que, en varios casos, los renglones hijo se pueden modificar o eliminar siempre y cuando existan renglones hermano. Por ejemplo, en una restricción M-M, un renglón hijo se puede eliminar siempre y cuando existan hermanos. (¡El último hijo nunca se va de casa!) Para la restricción M-M de la figura 10-18(a), se puede eliminar un renglón particular CARGO-DIARIO siempre y cuando quede cuando menos uno de ellos.

Con excepción de las consideraciones relacionadas con los hermanos, las reglas para evitar fragmentos cuando se procesan renglones hijo son similares a las de los padres. Asegúrese de entender cada enunciado de la figura 10-19(b).

USO DEL DBMS PARA IMPONER RESTRICCIONES DE CARDINALIDAD. Considerando el análisis anterior es útil tomar en cuenta las restricciones en la figura 10-19 en el contexto de las facilidades de definición de la relación en Access. Suponga que

► FIGURA 10-20

Propiedades de relación en Access



una base de datos tiene un objeto EMPLEADO con múltiples números de teléfono. Para mostrar la relevancia de las columnas al centro de la figura 10-19 suponga que la llave de EMPLOYEE (EMPLEADO) no es sustituta, sino EMPLOYEE.Name (EMPLEADO.Nombre).

La figura 10-20 muestra la ventana de definición de relación para este ejemplo. La marca en la caja Enforce Referential Integrity (Imponer Integridad Referencial) indica que Access no permitirá crear un nuevo renglón EMPLOYEE_PhoneNum, a menos que el valor de EmployeeName ya esté presente en un renglón en Employee. Éste es el significado de integridad referencial, como se analizó previamente. La segunda marca, en Cascade Update Related Fields (Campos relacionados de actualización en cascada), significa que si se cambia el nombre de un empleado en la tabla Employee, entonces ese cambio se propagará a todos los renglones relacionadas con EMPLOYEE_PhoneNum. Esto impone las reglas en las columnas de enmedio de la figura 10-19. (Nuevamente, esto no sería necesario si la llave sustituta se usara en su lugar.)

La última marca, en Cascade Delete Related Records (Registros relacionados de eliminación en cascada), significa que cuando se borra un renglón EMPLOYEE se eliminarán también todos los que están conectados con EMPLOYEE_PhoneNum. Esto es similar a lo que se analizó anteriormente en la sección de eliminación de instancias de vistas.

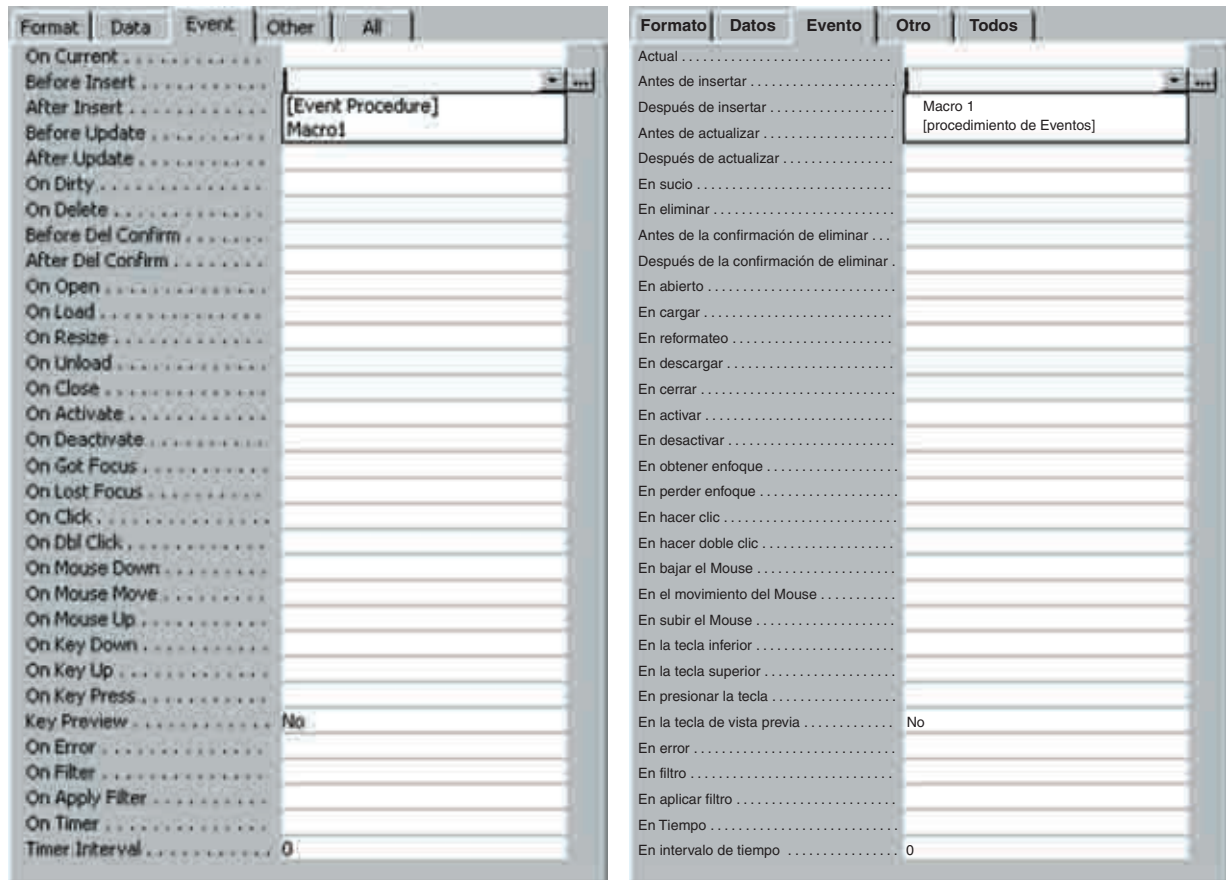
Esta característica de Access direcciona las reglas en la segunda y tercera columnas de la figura 10-19(a) y las de la primera y segunda columnas de la figura 10-19(b). No direcciona las reglas en la primera columna de la figura 10-19(a), tampoco en la tercera columna de la figura 10-19(b). Esas reglas necesitan ser codificadas en procedimientos almacenados (stored procedures) o en la aplicación.

RESTRICCIONES DE LAS REGLAS DEL NEGOCIO

Las restricciones de las reglas del negocio son específicas para la lógica y los requerimientos de una aplicación determinada. Surgen de los procedimientos y las políticas que existen en la empresa los cuales usará la aplicación de la base de datos. Ejemplos de reglas del negocio en una aplicación de ventas son:

► FIGURA 10-21

Eventos para formas de Access



- Ningún cheque de comisión puede exceder el 50% del total de la “bolsa de comisión”
- No se generará ninguna cancelación de pedido si el valor total de los artículos es menor a \$200
- Los costos de envío no se le cobran a los clientes distinguidos
- Los vendedores no pueden crear pedidos para ellos mismos
- Para que un empleado sea gerente de ventas primero debe ser vendedor

Debido a que las reglas del negocio son dependientes de la aplicación, no existen características genéricas de los productos DBMS para imponerlas. Más bien, los productos DBMS proporcionan medios para insertar un código antes o después de los eventos más importantes. La figura 10-21 muestra una lista de los eventos que pueden ser capturados en las formas de Access. En esta figura, el programador está en el proceso de añadir lógica al evento Antes de insertar (Before Insert). La lógica puede tener una de varias formas: un procedimiento del evento en el lenguaje de expresión de Access, o de Visual Basic (u otro lenguaje de programación), o un macro de Access. Todos los datos en la forma y en la base de datos son accesibles para el procedimiento del evento o macro. Así, se pueden imponer cualesquiera de las reglas de la lista de viñetas mediante los acontecimientos que se capturan.

Con los productos servidores DBMS tales como ORACLE y SQL Server, se usa un medio similar. La lógica puede ser codificada en disparadores (trigger), los cuales son procedimientos almacenados (stored procedures) que se invocan cuando tienen lugar eventos en la base de datos. Los eventos que se pueden capturar son similares a los que se muestran en la figura 10-21.

► SEGURIDAD Y CONTROL

La cuarta función principal de una aplicación de base de datos listada en la figura 10-1 es proporcionar mecanismos de seguridad y control. El objetivo es crear aplicaciones en las que sólo los usuarios autorizados puedan realizar actividades apropiadas en el tiempo correcto.

SEGURIDAD

La mayoría de los productos DBMS proporcionan nombre del usuario y contraseña de seguridad. Una vez que el usuario se registra, se puede limitar el acceso a ciertas formas, reportes, tablas e incluso a columnas de tablas. Esto es tan apropiado y útil como se quiera. Sin embargo, no ayuda a limitar los datos que los usuarios pueden ver.

Por ejemplo, en la aplicación de la base de datos de recursos humanos cada empleado sólo debería poder ver sus propios registros. Ciertos empleados de recursos humanos deberían poder consultar alguna información acerca de todos los empleados, y los gerentes de recursos humanos deberían poder ver toda la información acerca de todos los empleados.

Limitar el acceso a ciertas formas y reportes no ayuda. Cada empleado necesita ver la Forma del Empleado; la restricción necesita ser que el empleado sólo pueda ver la información de la forma que le pertenece (con las excepciones señaladas). Algunas veces usted escuchará los términos **seguridad horizontal** y **seguridad vertical**. Para entenderlos, piense en una tabla. La seguridad vertical limitaría el acceso a ciertas columnas, pero se podrían ver todos los renglones. La seguridad horizontal limitaría el acceso a ciertos renglones, pero se podrían ver todas las columnas.

Las aplicaciones que limitan a los usuarios a ciertas formas, reportes, tablas, o columnas proporcionan seguridad vertical. Las que limitan a los usuarios a ciertos datos en formas, reportes, tablas o columnas proporcionan seguridad horizontal. Los nombres de usuarios y las contraseñas se pueden usar con facilidad para proporcionar seguridad vertical. La seguridad horizontal generalmente requiere que el programador escriba un código de aplicación.

Por ejemplo, para proporcionar seguridad horizontal en la aplicación de empleado, el código de aplicación obtendría el nombre del usuario del sistema de seguridad del DBMS y limitaría el acceso a los renglones que contienen el nombre, o que están ligadas a renglones con ese nombre a través de joins. Una manera de hacer esto es anexas el nombre del usuario como una cláusula WHERE en los enunciados SQL.

Debido a que cada situación es diferente, no podemos decir más. Sólo esté consciente de que cuando los productos DBMS dicen que apoyan la seguridad, con frecuencia sólo significa que tienen seguridad vertical vía el nombre del usuario y la contraseña.

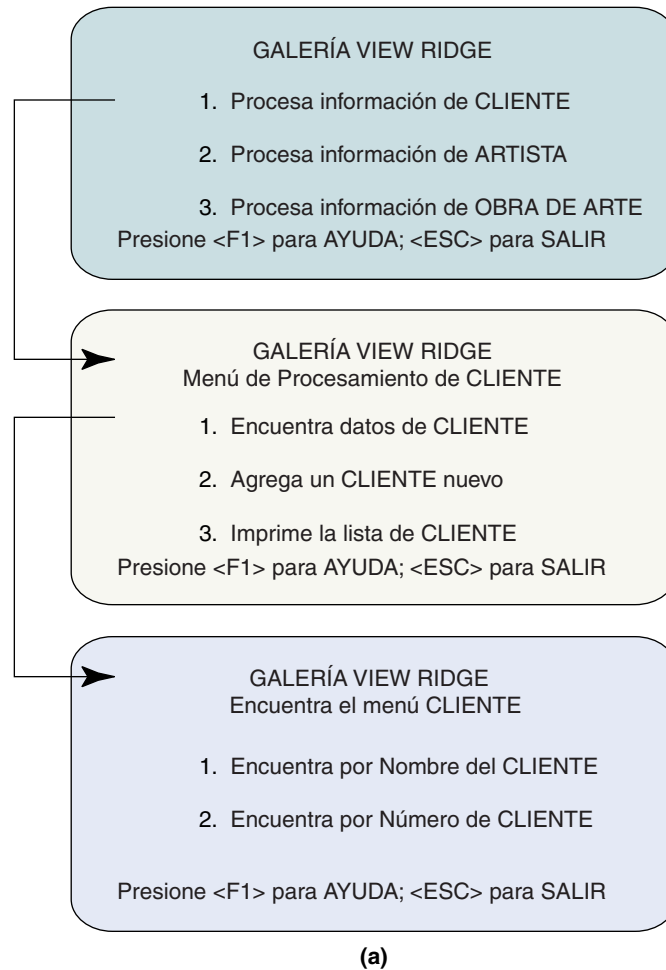
CONTROL

La mayoría de las aplicaciones de base de datos proporcionan control a través de menús. La figura 10-22(a) muestra un sistema de menús para una aplicación pre-GUI, y la figura 10-22(b) muestra los mismos menús para una aplicación GUI.

Los menús en la figura 10-22 son estáticos. El control más eficaz se puede proporcionar cambiando dinámicamente el contenido del menú conforme el usuario cambia el contexto. Puede ver menús de ese tipo en Access cuando la barra de herramientas cambia, dependiendo de si usted está en las herramientas de definición de la tabla, de definición de la forma, o de definición del reporte. Los programadores de aplicación de la base de datos pueden usar una estrategia similar para cambiar las opciones de menú, dependiendo de la forma o reporte que un usuario esté viendo, e incluso de la acción que el usuario esté llevando a cabo en la forma. Con el uso de Access, un programador puede cambiar opciones de menú cuando captura eventos como los que se muestran en la figura 10-21, y reestructurar drásticamente las opciones del menú.

► FIGURA 10-22

Jerarquía de menús de la galería View Ridge: (a) sin usar una interfaz de usuario gráfica, y (b) usando una interfaz de usuario gráfica



(b)

Un tipo diferente de control es el que se refiere a las transacciones. En el capítulo 11 aprenderá los medios de control del procesamiento multiusuarios para que las acciones de uno no tengan consecuencias inapropiadas en las acciones de otro. Una parte clave del control multiusuarios es identificar las fronteras de trabajo que se deben completar como unidad, a veces se les llama **fronteras de transacción**. Por ejemplo, la serie de enunciados SQL para crear la vista que se muestra al principio de este capítulo se necesita completar como una sola transacción.

No anticiparemos el análisis en ese capítulo, sólo diremos que especificar las fronteras de la transacción es trabajo de la aplicación. Por lo general esto se lleva a cabo mediante la ejecución de un enunciado como BEGIN TRANSACTION al comienzo de una unidad de trabajo y END TRANSACTION cuando el trabajo concluye.

Establecer fronteras es fácil si las vistas del usuario están bien diseñadas: las instrucciones de la aplicación BEGIN TRANSACTION al inicio de la vista y END TRANSACTION al final de la vista. Continuaremos con este análisis en el capítulo 12.

► LÓGICA DE LA APLICACIÓN

La última función de la aplicación de una base de datos que se lista en la figura 10-1 es ejecutar la lógica de la aplicación. Este tema por lo general se analiza en las clases de desarrollo de sistemas y en diversos textos, y por eso aquí diremos muy poco al respecto.

Las necesidades de la lógica de aplicación surgen de los requerimientos de los sistemas. En un sistema de registro de pedidos la lógica de la aplicación se refiere a cómo llevar inventarios de almacén, cómo encargarse de las existencias insuficientes en inventario, cómo programar un pedido cancelado y actividades similares. En una aplicación de base de datos basada en una forma como la de la galería, el código para avalar esa lógica se relaciona con eventos como los que se muestran en la figura 10-21.

Para otras aplicaciones de la base de datos, donde las formas se materializan mediante la aplicación, en lugar de que sea por medio del DBMS (común en las aplicaciones de macrocomputadoras), la aplicación procesa los datos en una forma similar a la del procesamiento de archivos. La lógica se codifica en un renglón dentro de la aplicación conforme recibe los datos y los pone en el DBMS. Algunos programas de aplicación reciben información de otros; en este caso, también la lógica de la aplicación está contenida en línea.

Así, los medios por los cuales la lógica de la aplicación se implanta en las aplicaciones de la base de datos depende tanto de la lógica como del medio ambiente. Se usan diferentes medios para las aplicaciones de escritorio, servidor del cliente, macrocomputadora, y tecnología de Internet. Usted ha visto cómo capturar hechos en las aplicaciones de escritorio. En los siguientes capítulos consideraremos otros medios.

► RESUMEN

Las cinco funciones principales de la aplicación de una base de datos son: (1) crear, leer, actualizar y eliminar las vistas (CRUD), (2) materializar o formatear vistas, (3) imponer restricciones, (4) proporcionar mecanismos de seguridad y control, y (5) ejecutar la lógica del negocio.

Una vista es una lista estructurada de elementos de datos de las entidades u objetos semánticos en el modelo de datos. Una instancia de vista es una que está llena de datos. Debido a que las vistas están estructuradas, un elemento de datos puede aparecer más de una vez. Para leer una vista se emiten uno o más enunciados SQL con el fin de obtener valores de datos. Se requerirá más de un enunciado SQL si la vista incluye dos o más caminos de relaciones a través del esquema. Un recordset es una relación con un empaquetador del objeto de programación.

Crear una vista requiere almacenar uno o más nuevos renglones en tablas, y posiblemente crear o cambiar valores de llaves externas, así como establecer relaciones. Hay tres tipos de actualizaciones: cambiar los datos existentes, cambiar las relaciones, y crear nuevos renglones para los atributos de multivaluados. Eliminar una vista implica borrar uno o más renglones y ajustar las llaves externas. El problema cuando se elimina es saber qué tanto se debe descartar. Hay que suprimir las entidades débiles si la entidad de la que dependen se elimina. También hay que quitar los atributos multivaluado dentro de un objeto semántico. Con algunos productos DBMS las relaciones se pueden marcar para la eliminación en cascada, de tal forma que el DBMS remueva los renglones dependientes que sea apropiado.

Una forma es una pantalla desplegada que se usa para introducir datos y editarlos. Los principios del diseño de formas son que la estructura de ésta debe reflejar la de la vista, la semántica de los datos debe ser gráficamente evidente, y la estructura de la forma debe fomentar la acción apropiada. Las cajas de lista desplegables, los botones de opción y las cajas de verificación se pueden usar para aumentar la utilidad de las formas.

Los reportes también se deben diseñar de tal forma que su estructura refleje la de la vista que materializan. El ordenamiento de los reportes con frecuencia implica la

existencia de otros objetos. Con la mayoría de los editores de reportes es difícil construir reportes que sigan más de una pista multivaluada a través del esquema. Los reportes con frecuencia calculan atributos de datos; por lo general es mejor no almacenar esos atributos en la base de datos.

Las restricciones se pueden imponer ya sea mediante el DBMS o a través de un programa de aplicación. En la mayoría de los casos es mucho mejor para el DBMS aplicarlas cuando sea posible, principalmente debido a que el DBMS es un punto central a través del cual pasan todos los cambios de datos. En algunos casos, el DBMS no tiene las características para imponer restricciones; sin embargo, se deben imponer mediante la aplicación. Las restricciones de dominio imponen la parte física de las definiciones de dominio. Los valores requeridos son otro tipo de restricción. Tener un valor requerido evita la ambigüedad de los valores nulos. Las restricciones de unicidad se aplican mejor con el DBMS; por lo general se hace construyendo índices.

Hay dos tipos de restricciones de relación: integridad referencial y cardinalidad de relación. Las restricciones de integridad referencial se aplican mejor por medio del DBMS. Las restricciones de cardinalidad de relación surgen del establecimiento de la cardinalidad en los vínculos del objeto, de cualquier establecimiento de cardinalidad mínima que no sea cero, o de una cardinalidad máxima que no sea 1 ni N. Excepto en el caso de la cardinalidad 1.1 en el lado de los hijos de las relaciones 1:N, estas restricciones se deben imponer mediante un código de aplicación. Las reglas para la aplicación de restricciones en las relaciones 1:N se resumen en la figura 10-19. Las restricciones de la regla de negocio mediante un código de aplicación que se invoca a través de eventos capturados, de disparadores (triggers), o de programas de aplicación en línea.

La mayoría de los productos DBMS proporcionan el nombre del usuario y la contraseña de seguridad. Esto se usa para proporcionar seguridad vertical; la seguridad horizontal se debe proveer mediante el código de aplicación. La mayoría de las aplicaciones ofrecen control a través de menús. El mejor control ocurre cuando se cambian los menús conforme cambia el contexto del usuario. Los programas de aplicación desempeñan un papel importante en la definición de las fronteras de transacción. La lógica de la aplicación se codifica y se invoca capturando eventos, así como a través de otros medios que serán explicados en los siguientes capítulos.

► PREGUNTAS DEL GRUPO I

- 10.1 Enumere las cinco funciones principales de una aplicación de la base de datos.
- 10.2 Explique el significado del acrónimo CRUD.
- 10.3 Defina el término *vista* como se utilizó en este capítulo.
- 10.4 ¿Qué es una instancia de vista?
- 10.5 Explique en qué difiere una vista de una materialización.
- 10.6 ¿Un atributo puede aparecer más de una vez en una vista? ¿Por qué?
- 10.7 ¿Bajo qué condiciones se puede leer una vista con un enunciado SQL?
- 10.8 ¿En qué casos leer una vista requiere más de un enunciado SQL?
- 10.9 Explique los dos caminos que existen en la vista de Cliente en la figura 10-4.
- 10.10 Defina el término *recordset*.
- 10.11 Describa en términos generales el trabajo que se requiere cuando se crea una instancia de vista.
- 10.12 ¿Cómo se crean las nuevas relaciones cuando se desarrolla una instancia de vista?
- 10.13 ¿Qué técnica se puede usar para obtener el valor de una llave sustituta cuando se insertan nuevos renglones en una tabla?
- 10.14 Liste los tres tipos de cambio que pueden ocurrir cuando se actualiza una instancia de vista.

- 10.15 Explique cómo cambiar las relaciones 1:N, y las relaciones N:M.
- 10.16 ¿Cuál es la dificultad principal cuando se escribe un código para eliminar una instancia de vista?
- 10.17 ¿Cómo puede ayudar un modelo E-R a determinar qué tanto se debe eliminar?
- 10.18 ¿Cómo puede ayudar un objeto semántico a determinar qué tanto se debe suprimir?
- 10.19 ¿Qué son las eliminaciones en cascada y por qué son importantes?
- 10.20 Explique el enunciado “la estructura de la forma debería reflejar la estructura de la vista”.
- 10.21 ¿Cómo se pueden diseñar las formas para hacer la semántica de la información gráficamente evidente?
- 10.22 ¿Cómo se pueden diseñar las formas para fomentar la acción apropiada?
- 10.23 Explique el papel de las cajas de listas desplegables, los grupos de opción y las cajas de verificación en el diseño de formas.
- 10.24 ¿Qué limitación existe para la materialización de reportes de las vistas?
- 10.25 Explique por qué los valores calculados en reportes por lo regular no deberían almacenarse en la base de datos.
- 10.26 Explique cómo el requerimiento de objetos de reporte ordenados por un valor cambia el objeto implícito del reporte.
- 10.27 ¿Por qué las restricciones normalmente deberían ser impuestas por el DBMS y no mediante un programa particular de formas, reportes o aplicación?
- 10.28 ¿Por qué a veces las restricciones se imponen en programas de aplicación?
- 10.29 Dé un ejemplo sobre una restricción de dominio y explique cómo se puede imponer con Access.
- 10.30 Describa la ambigüedad que surge cuando los valores son nulos, así como dos maneras para que éstos valores puedan ser eliminados.
- 10.31 ¿Por qué el DBMS es el que normalmente debería imponer restricciones de unicidad?
- 10.32 Describa las dos fuentes de restricciones de cardinalidad.
- 10.33 Nombre dos tipos de restricción de relación.
- 10.34 ¿Cuál es la mejor manera de imponer restricciones sobre valores de llave externa?
- 10.35 ¿Cómo se puede imponer una restricción de cardinalidad 1.1 en el lado de los hijos de una relación 1:N?
- 10.36 Defina *fragmento* y *huérfano*.
- 10.37 Explique las entradas en la primera columna de la figura 10-19(a).
- 10.38 Explique por qué la columna central de la figura 10-19(a) es innecesaria cuando se usan las llaves sustitutas.
- 10.39 Explique las entradas en la tercera columna de la figura 10-19(a).
- 10.40 Explique las entradas en la primera columna de la figura 10-19(b).
- 10.41 Explique las entradas en la tercera columna de la figura 10-19(b).
- 10.42 Explique por qué la primera columna en la figura 10-19(a) y la tercera columna en la figura 10-19(b) no están impuestas mediante las propiedades de relación de Access que se muestran en la figura 10-20.
- 10.43 Dé un ejemplo sobre restricción de una regla del negocio que se podría aplicar al modelo de datos de la figura 10-3. Explique cómo se puede imponer esta restricción al capturar un evento.
- 10.44 Defina seguridad *horizontal* y *vertical*.
- 10.45 ¿Qué tipo de seguridad se avala con el nombre del usuario y la contraseña?
- 10.46 ¿Qué tipo de seguridad se debe apoyar mediante un código de aplicación?

- 10.47 Explique por qué los menús dinámicos son mejores que los estáticos.
- 10.48 ¿Cómo es la lógica de negocios conectada a una base de datos cuando se usa Access?

► PREGUNTAS DEL GRUPO II

Las preguntas 10.49 a la 10.52 pertenecen a la siguiente vista de Artista, la cual está basada en el modelo de datos de la figura 10-3.

ARTISTA.Nombre
 ARTISTA.Nacionalidad
 TRANSACCIÓN.FechadeCompra
 TRANSACCIÓN.PreciodeVenta. . .
 CLIENTE.Nombre
 CLIENTE.Teléfono.CódigodeÁrea
 CLIENTE.Teléfono.NúmeroLocal
 CLIENTE.Nombre. . .

Las elipsis (. . .) se refieren a estructuras que se pueden repetir.

- 10.49 Codifique los enunciados SQL para leer la instancia de “Mark Tobey” en esta vista.
- 10.50 Codifique los enunciados SQL para crear una nueva instancia de esta vista. Suponga que usted tiene datos para ARTISTA, una TRANSACCIÓN y muchos CLIENTE.Nombre(s) para la segunda instancia de CLIENTE.Nombre. Suponga que estos datos se localizan en una estructura llamada NuevoArtista. Use sintaxis similar a la del texto.
- 10.51 Codifique los enunciados SQL para actualizar esta vista como sigue:
- Cambie la ortografía de Mark Tobey a Mark Toby
 - Cree una nueva Transacción para Mark Toby. Suponga que tiene los datos necesarios de la transacción, obra y cliente en una estructura llamada NuevaTransacción
 - Agregue nuevos clientes interesados en Mark Toby. Suponga que se encuentran almacenados en un conjunto al que usted puede tener acceso con el enunciado “Para cada NuevoCliente.Nombre”
- 10.52 Codifique los enunciados SQL para eliminar el renglón de Mark Toby y todos los renglones TRABAJO y TRANSACCIÓN relacionados.

► PROYECTOS

A. Usando Access, cree la base de datos que se muestra en la figura 10-3. Cree una forma para la vista de Artista que se muestra en la pregunta 10.49. Justifique el diseño de su forma usando los principios de este capítulo. Sugerencia: puede utilizar un asistente (wizard) para crear uno de los subformularios, pero necesitará agregar el segundo de forma manual. También, agregue las cajas de conjunto manualmente después de que haya creado las formas para la subforma.

B. Termine el Proyecto A al final de los capítulos 3 y 4, si aún no lo ha hecho.

- Liste y describa el propósito de las tres vistas, las tres formas y los tres reportes que considere serán necesarios para esta aplicación.
- Muestre la estructura de un menú GUI desplegable para esta aplicación. Usando su modelo, diseñe una de las formas para introducir nuevas propiedades. Expli-

que qué tipo de control (caja de texto, lista desplegable) se usa para cada campo. Justifique la estructura usando los conceptos que presentamos en este capítulo.

► PREGUNTAS DEL PROYECTO FIREDUP

Lea el proyecto FiredUp al final del capítulo 9. Use las cuatro tablas que se describen ahí para resolver lo siguiente:

- A. Construya las siguientes vistas. Use la figura 10-4 como ejemplo.
 1. Construya una vista que comience con ESTUFA y contenga todas las tablas y los datos. Llame a la vista ESTUFA_VISTA.
 2. Construya una vista que inicie en CLIENTE e incluya todas las tablas y los datos. Llame a la vista CLIENTE_VISTA.
 3. Construya una vista que dé comienzo en REGISTRO y contenga todas las tablas, excepto ESTUFA_REPARACIÓN. Llame a la vista REGISTRO_VISTA.
 4. Construya una vista que inicie con ESTUFA_REPARACIÓN y contenga todas las tablas y los datos. Llame a la vista ESTUFA_REPARACIÓN_VISTA.
- B. Construya enunciados SQL para procesar vistas como sigue. Use el comienzo de SQL de la página 262 como ejemplo.
 1. Muestre los enunciados SQL necesarios para leer ESTUFA_VISTA. Suponga que inicia con un NúmeroSerie en particular.
 2. Muestre los enunciados SQL que se necesitan para construir una nueva instancia de REGISTRO_VISTA. Suponga que los datos de estufa que se necesitan ya están en la base de datos, pero falta la información necesaria del cliente.
 3. Muestre los enunciados SQL para construir una nueva instancia de ESTUFA_REPARACIÓN. Suponga que los datos de ESTUFA están en la base de datos, pero falta de información de CLIENTE. Registre la ESTUFA mientras anota la reparación.
 4. Muestre los enunciados SQL para eliminar todos los registros referentes a una estufa en particular. Use la vista de ESTUFA.