

FUNDAMENTOS DE BASES DE DATOS

Cuarta edición

Abraham Silberschatz

Bell Laboratories

Henry F. Korth

Bell Laboratories

S. Sudarshan

Instituto Indio de Tecnología, Bombay

Traducción

FERNANDO SÁENZ PÉREZ

ANTONIO GARCÍA CORDERO

CAROLINA LÓPEZ MARTÍNEZ

LUIS MIGUEL SÁNCHEZ BREA

OLGA MATA GÓMEZ

M.^a VICTORIA GONZÁLEZ DEL CAMPO RODRÍGUEZ BARBERO

Universidad Complutense de Madrid

Revisión técnica

LUIS GRAU FERNÁNDEZ

Universidad Nacional de Educación a Distancia



MADRID • BUENOS AIRES • CARACAS • GUATEMALA • LISBOA • MÉXICO
NUEVA YORK • PANAMÁ • SAN JUAN • SANTAFÉ DE BOGOTÁ • SANTIAGO • SÃO PAULO
AUCKLAND • HAMBURGO • LONDRES • MILÁN • MONTREAL • NUEVA DELHI • PARÍS
SAN FRANCISCO • SIDNEY • SINGAPUR • ST. LOUIS • TOKIO • TORONTO

FUNDAMENTOS DE BASES DE DATOS. Cuarta edición

No está permitida la reproducción total o parcial de este libro, ni su tratamiento informático, ni la transmisión de ninguna forma o por cualquier medio, ya sea electrónico, mecánico, por fotocopia, por registro u otros métodos, sin el permiso previo y por escrito de los titulares del Copyright.

DERECHOS RESERVADOS © 2002, respecto a la cuarta edición en español, por

McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U.

Edificio Valrealty, 1.^a planta

Basauri, 17

28023 Aravaca (Madrid)

Traducido de la cuarta edición en inglés de
Database System Concepts

Copyright © MMI, por McGraw-Hill Inc.

ISBN: 0-07-228363-7

ISBN: 84-481-3654-3

Depósito legal: M.

Editora: Concepción Fernández Madrid

Editora de mesa: Susana Santos Prieto

Cubierta: DIMA

Compuesto en FER

Impreso en:

IMPRESO EN ESPAÑA - PRINTED IN SPAIN

*En memoria de mi padre, Joseph Silberschatz
y de mis abuelos Stephá y Aaron Resenblum.*

Avi Silberschatz

*A mi esposa, Joan,
mis hijos, Abigail y Joseph,
y mis padres, Henry y Frances*

Hank Korth

*A mi esposa, Sita,
mi hijo, Madhur,
y mi madre, Indira.*

S. Sudarshan

CONTENIDO BREVE

PREFACIO, XVII

CAPÍTULO 1 INTRODUCCIÓN, 1

PARTE PRIMERA: MODELOS DE DATOS

CAPÍTULO 2 MODELO ENTIDAD-RELACIÓN, 19

CAPÍTULO 3 EL MODELO RELACIONAL, 53

PARTE SEGUNDA: BASES DE DATOS RELACIONALES

CAPÍTULO 4 SQL, 87

CAPÍTULO 5 OTROS LENGUAJES RELACIONALES, 119

CAPÍTULO 6 INTEGRIDAD Y SEGURIDAD, 141

CAPÍTULO 7 DISEÑO DE BASES DE DATOS RELACIONALES, 161

PARTE TERCERA: BASES DE DATOS BASADAS EN OBJETOS Y XML

CAPÍTULO 8 BASES DE DATOS ORIENTADAS A OBJETOS, 193

CAPÍTULO 9 BASES DE DATOS RELACIONALES ORIENTADAS A OBJETOS, 211

CAPÍTULO 10 XML, 227

PARTE CUARTA: ALMACENAMIENTO DE DATOS Y CONSULTAS

CAPÍTULO 11 ALMACENAMIENTO Y ESTRUCTURA DE ARCHIVOS, 249

CAPÍTULO 12 INDEXACIÓN Y ASOCIACIÓN, 283

CAPÍTULO 13 PROCESAMIENTO DE CONSULTAS, 319

CAPÍTULO 14 OPTIMIZACIÓN DE CONSULTAS, 343

PARTE QUINTA: GESTIÓN DE TRANSACCIONES

CAPÍTULO 15 TRANSACCIONES, 367

CAPÍTULO 16 CONTROL DE CONCURRENCIA, 383

CAPÍTULO 17 SISTEMA DE RECUPERACIÓN, 413

PARTE SEXTA: ARQUITECTURA DE LOS SISTEMAS DE BASES DE DATOS

CAPÍTULO 18 ARQUITECTURAS DE LOS SISTEMAS DE BASES DE DATOS, 445

CAPÍTULO 19 BASES DE DATOS DISTRIBUIDAS, 463

CAPÍTULO 20 BASES DE DATOS PARALELAS, 493

PARTE SÉPTIMA: OTROS TEMAS

CAPÍTULO 21 DESARROLLO DE APLICACIONES Y ADMINISTRACIÓN, 511

CAPÍTULO 22 CONSULTAS AVANZADAS Y RECUPERACIÓN DE INFORMACIÓN, 537

CAPÍTULO 23 TIPOS DE DATOS AUTOMÁTICOS Y NUEVAS APLICACIONES, 569

CAPÍTULO 24 PROCESAMIENTO AVANZADO DE TRANSACCIONES, 589

CAPÍTULO 25 ORACLE, 611

PARTE OCTAVA: ESTUDIO DE CASOS

CAPÍTULO 26 DB2 DE IBM, 629

CAPÍTULO 27 SQL SERVER DE MICROSOFT, 645

BIBLIOGRAFÍA, 673

DICCIONARIO BILINGÜE, 695

ÍNDICE, 771

CONTENIDO

PREFACIO, XVII

CAPÍTULO 1: INTRODUCCIÓN

- 1.1. APLICACIONES DE LOS SISTEMAS DE BASES DE DATOS, 1
 - 1.2. SISTEMAS DE BASES DE DATOS FRENTE A SISTEMAS DE ARCHIVOS, 2
 - 1.3. VISIÓN DE LOS DATOS, 3
 - 1.4. MODELOS DE LOS DATOS, 5
 - 1.5. LENGUAJES DE BASES DE DATOS, 7
 - 1.6. USUARIOS Y ADMINISTRADORES DE LA BASE DE DATOS, 8
 - 1.7. GESTIÓN DE TRANSACCIONES, 10
 - 1.8. ESTRUCTURA DE UN SISTEMA DE BASES DE DATOS, 10
 - 1.9. ARQUITECTURAS DE APLICACIONES, 12
 - 1.10. HISTORIA DE LOS SISTEMAS DE BASES DE DATOS, 13
 - 1.11. RESUMEN, 14
- TÉRMINOS DE REPASO, 15
- EJERCICIOS, 15
- NOTAS BIBLIOGRÁFICAS, 16
- HERRAMIENTAS, 16

PARTE PRIMERA: MODELOS DE DATOS

CAPÍTULO 2: MODELO ENTIDAD-RELACIÓN

- 2.1. CONCEPTOS BÁSICOS, 19
 - 2.2. RESTRICCIONES, 23
 - 2.3. CLAVES, 24
 - 2.4. CUESTIONES DE DISEÑO, 25
 - 2.5. DIAGRAMA ENTIDAD-RELACIÓN, 28
 - 2.6. CONJUNTOS DE ENTIDADES DÉBILES, 32
 - 2.7. CARACTERÍSTICAS DEL MODELO E-R EXTENDIDO, 33
 - 2.8. DISEÑO DE UN ESQUEMA DE BASE DE DATOS E-R, 39
 - 2.9. REDUCCIÓN DE UN ESQUEMA E-R A TABLAS, 43
 - 2.10. EL LENGUAJE DE MODELADO UNIFICADO UML, 46
 - 2.11. RESUMEN, 48
- TÉRMINOS DE REPASO, 49
- EJERCICIOS, 49
- NOTAS BIBLIOGRÁFICAS, 52
- HERRAMIENTAS, 52

CAPÍTULO 3: EL MODELO RELACIONAL

- 3.1. LA ESTRUCTURA DE LAS BASES DE DATOS RELACIONALES, 53
- 3.2. EL ÁLGEBRA RELACIONAL, 59
- 3.3. OPERACIONES DEL ÁLGEBRA RELACIONAL EXTENDIDA, 67
- 3.4. MODIFICACIÓN DE LA BASE DE DATOS, 71
- 3.5. VISTAS, 73
- 3.6. EL CÁLCULO RELACIONAL DE TUPLAS, 75

- 3.7. EL CÁLCULO RELACIONAL DE DOMINIOS, 78
- 3.8. RESUMEN, 80
- TÉRMINOS DE REPASO, 81
- EJERCICIOS, 81
- NOTAS BIBLIOGRÁFICAS, 83

PARTE SEGUNDA: BASES DE DATOS RELACIONALES

CAPÍTULO 4: SQL

- 4.1. INTRODUCCIÓN, 87
- 4.2. ESTRUCTURA BÁSICA, 88
- 4.3. OPERACIONES SOBRE CONJUNTOS, 92
- 4.4. FUNCIONES DE AGREGACIÓN, 93
- 4.5. VALORES NULOS, 95
- 4.6. SUBCONSULTAS ANIDADAS, 95
- 4.7. VISTAS, 98
- 4.8. CONSULTAS COMPLEJAS, 99
- 4.9. MODIFICACIÓN DE LA BASE DE DATOS, 100
- 4.10. REUNIÓN DE RELACIONES, 103
- 4.11. LENGUAJE DE DEFINICIÓN DE DATOS, 106
- 4.12. SQL INCORPORADO, 109
- 4.13. SQL DINÁMICO, 111
- 4.14. OTRAS CARACTERÍSTICAS DE SQL, 114
- 4.15. RESUMEN, 115
- TÉRMINOS DE REPASO, 115
- EJERCICIOS, 116
- NOTAS BIBLIOGRÁFICAS, 117

CAPÍTULO 5: OTROS LENGUAJES RELACIONALES

- 5.1. QUERY-BY-EXAMPLE, 119
- 5.2. DATALOG, 127
- 5.3. INTERFACES DE USUARIO Y HERRAMIENTAS, 135
- 5.4. RESUMEN, 137
- TÉRMINOS DE REPASO, 137
- EJERCICIOS, 137
- NOTAS BIBLIOGRÁFICAS, 139
- HERRAMIENTAS, 139

CAPÍTULO 6: INTEGRIDAD Y SEGURIDAD

- 6.1. RESTRICCIONES DE LOS DOMINIOS, 141
- 6.2. INTEGRIDAD REFERENCIAL, 142
- 6.3. ASERTOS, 145
- 6.4. DISPARADORES, 146
- 6.5. SEGURIDAD Y AUTORIZACIÓN, 149
- 6.6. AUTORIZACIÓN EN SQL, 153
- 6.7. CIFRADO Y AUTENTICACIÓN, 155
- 6.8. RESUMEN, 156
- TÉRMINOS DE REPASO, 157
- EJERCICIOS, 157
- NOTAS BIBLIOGRÁFICAS, 159

CAPÍTULO 7: DISEÑO DE BASES DE DATOS RELACIONALES

- 7.1. PRIMERA FORMA NORMAL, 161
 - 7.2. DIFICULTADES EN EL DISEÑO DE BASES DE DATOS RELACIONALES, 162
 - 7.3. DEPENDENCIAS FUNCIONALES, 163
 - 7.4. DESCOMPOSICIÓN, 169
 - 7.5. PROPIEDADES DESEABLES DE LA DESCOMPOSICIÓN, 171
 - 7.6. FORMA NORMAL DE BOYCE-CODD, 174
 - 7.7. TERCERA FORMA NORMAL, 177
 - 7.8. CUARTA FORMA NORMAL, 180
 - 7.9. OTRAS FORMAS NORMALES, 182
 - 7.10. PROCESO GENERAL DEL DISEÑO DE BASES DE DATOS, 183
 - 7.11. RESUMEN, 185
- TÉRMINOS DE REPASO, 186
EJERCICIOS, 186
NOTAS BIBLIOGRÁFICAS, 188

PARTE TERCERA: BASES DE DATOS BASADAS EN OBJETOS Y XML**CAPÍTULO 8: BASES DE DATOS ORIENTADAS A OBJETOS**

- 8.1. NECESIDADES DE LOS TIPOS DE DATOS COMPLEJOS, 193
 - 8.2. EL MODELO DE DATOS ORIENTADO A OBJETOS, 194
 - 8.3. LENGUAJES ORIENTADOS A OBJETOS, 200
 - 8.4. LENGUAJES DE PROGRAMACIÓN PERSISTENTE, 200
 - 8.5. SISTEMAS C++ PERSISTENTES, 203
 - 8.6. SISTEMAS JAVA PERSISTENTES, 207
 - 8.7. RESUMEN, 208
- TÉRMINOS DE REPASO, 208
EJERCICIOS, 209
NOTAS BIBLIOGRÁFICAS, 209

CAPÍTULO 9: BASES DE DATOS RELACIONALES ORIENTADAS A OBJETOS

- 9.1. RELACIONES ANIDADAS, 211
 - 9.2. TIPOS COMPLEJOS, 212
 - 9.3. HERENCIA, 215
 - 9.4. TIPOS DE REFERENCIA, 217
 - 9.5. CONSULTAS CON TIPOS COMPLEJOS, 218
 - 9.6. FUNCIONES Y PROCEDIMIENTOS, 220
 - 9.7. COMPARACIÓN ENTRE LAS BASES DE DATOS ORIENTADAS A OBJETOS Y LAS BASES DE DATOS RELACIONALES ORIENTADAS A OBJETOS, 223
 - 9.8. RESUMEN, 223
- TÉRMINOS DE REPASO, 224
EJERCICIOS, 224
NOTAS BIBLIOGRÁFICAS, 225
HERRAMIENTAS, 226

CAPÍTULO 10: XML

- 10.1. ANTECEDENTES, 227
- 10.2. ESTRUCTURA DE LOS DATOS XML, 228
- 10.3. ESQUEMA DE LOS DOCUMENTOS XML, 230
- 10.4. CONSULTA Y TRANSFORMACIÓN, 233

- 10.5. LA INTERFAZ DE PROGRAMACIÓN DE APLICACIONES, 238
- 10.6. ALMACENAMIENTO DE DATOS XML, 239
- 10.7. APLICACIONES XML, 240
- 10.8. RESUMEN, 242
- TÉRMINOS DE REPASO, 243
- EJERCICIOS, 244
- NOTAS BIBLIOGRÁFICAS, 245
- HERRMIENTAS, 245

PARTE CUARTA: ALMACENAMIENTO DE DATOS Y CONSULTAS

CAPÍTULO 11: ALMACENAMIENTO Y ESTRUCTURA DE ARCHIVOS

- 11.1. VISIÓN GENERAL DE LOS MEDIOS FÍSICOS DE ALMACENAMIENTO, 249
- 11.2. DISCOS MAGNÉTICOS, 251
- 11.3. RAID, 255
- 11.4. ALMACENAMIENTO TERCIARIO, 260
- 11.5. ACCESO AL ALMACENAMIENTO, 262
- 11.6. ORGANIZACIÓN DE LOS ARCHIVOS, 264
- 11.7. ORGANIZACIÓN DE LOS REGISTROS EN ARCHIVOS, 268
- 11.8. ALMACENAMIENTO CON DICCIONARIOS DE DATOS, 271
- 11.9. ALMACENAMIENTO PARA LAS BASES DE DATOS ORIENTADAS A OBJETOS, 271
- 11.10. RESUMEN, 278
- TÉRMINOS DE REPASO, 279
- EJERCICIOS, 280
- NOTAS BIBLIOGRÁFICAS, 281

CAPÍTULO 12: INDEXACIÓN Y ASOCIACIÓN

- 12.1. CONCEPTOS BÁSICOS, 283
- 12.2. ÍNDICES ORDENADOS, 284
- 12.3. ARCHIVOS DE ÍNDICES DE ÁRBOL B⁺, 289
- 12.4. ARCHIVOS CON ÍNDICES DE ÁRBOL B, 297
- 12.5. ASOCIACIÓN ESTÁTICA, 298
- 12.6. ASOCIACIÓN DINÁMICA, 302
- 12.7. COMPARACIÓN DE LA INDEXACIÓN ORDENADA Y LA ASOCIACIÓN, 308
- 12.8. DEFINICIÓN DE ÍNDICES EN SQL, 309
- 12.9. ACCESOS MULTICLAVE, 309
- 12.10. RESUMEN, 314
- TÉRMINOS DE REPASO, 315
- EJERCICIOS, 316
- NOTAS BIBLIOGRÁFICAS, 317

CAPÍTULO 13: PROCESAMIENTO DE CONSULTAS

- 13.1. VISIÓN GENERAL, 319
- 13.2. MEDIDAS DEL COSTE DE UNA CONSULTA, 321
- 13.3. OPERACIÓN SELECCIÓN, 321
- 13.4. ORDENACIÓN, 324
- 13.5. OPERACIÓN REUNIÓN, 326
- 13.6. OTRAS OPERACIONES, 333
- 13.7. EVALUACIÓN DE EXPRESIONES, 335
- 13.8. RESUMEN, 339

TÉRMINOS DE REPASO, 339
 EJERCICIOS, 340
 NOTAS BIBLIOGRÁFICAS, 341

CAPÍTULO 14: OPTIMIZACIÓN DE CONSULTAS

14.1. VISIÓN GENERAL, 343
 14.2. ESTIMACIÓN DE LAS ESTADÍSTICAS DE LOS RESULTADOS DE LAS EXPRESIONES, 344
 14.3. TRANSFORMACIÓN DE EXPRESIONES RELACIONALES, 348
 14.4. ELECCIÓN DE LOS PLANES DE EVALUACIÓN, 352
 14.5. VISTAS MATERIALIZADAS, 358
 14.6. RESUMEN, 361
 TÉRMINOS DE REPASO, 362
 EJERCICIOS, 362
 NOTAS BIBLIOGRÁFICAS, 363

PARTE QUINTA: GESTIÓN DE TRANSACCIONES

CAPÍTULO 15: TRANSACCIONES

15.1. CONCEPTO DE TRANSACCIÓN, 367
 15.2. ESTADOS DE UNA TRANSACCIÓN, 369
 15.3. IMPLEMENTACIÓN DE LA ATOMICIDAD Y LA DURABILIDAD, 371
 15.4. EJECUCIONES CONCURRENTES, 372
 15.5. SECUENCIALIDAD, 374
 15.6. RECUPERABILIDAD, 377
 15.7. IMPLEMENTACIÓN DEL AISLAMIENTO, 378
 15.8. DEFINICIÓN DE TRANSACCIONES EN SQL, 378
 15.9. COMPROBACIÓN DE LA SECUENCIALIDAD, 379
 15.10. RESUMEN, 380
 TÉRMINOS DE REPASO, 381
 EJERCICIOS, 381
 NOTAS BIBLIOGRÁFICAS, 382

CAPÍTULO 16: CONTROL DE CONCURRENCIA

16.1. PROTOCOLOS BASADOS EN EL BLOQUEO, 383
 16.2. PROTOCOLOS BASADOS EN MARCAS TEMPORALES, 390
 16.3. PROTOCOLOS BASADOS EN VALIDACIÓN, 393
 16.4. GRANULARIDAD MÚLTIPLE, 394
 16.5. ESQUEMAS MULTIVERSIÓN, 396
 16.6. TRATAMIENTO DE INTERBLOQUEOS, 398
 16.7. OPERACIONES PARA INSERTAR Y BORRAR, 401
 16.8. NIVELES DÉBILES DE CONSISTENCIA, 403
 16.9. CONCURRENCIA EN ESTRUCTURAS DE ÍNDICE, 404
 16.10. RESUMEN, 406
 TÉRMINOS DE REPASO, 408
 EJERCICIOS, 409
 NOTAS BIBLIOGRÁFICAS, 411

CAPÍTULO 17: SISTEMA DE RECUPERACIÓN

17.1. CLASIFICACIÓN DE LOS FALLOS, 413
 17.2. ESTRUCTURA DEL ALMACENAMIENTO, 414
 17.3. RECUPERACIÓN Y ATOMICIDAD, 416

- 17.4. RECUPERACIÓN BASADA EN EL REGISTRO HISTÓRICO, 417
- 17.5. PAGINACIÓN EN LA SOMBRA, 422
- 17.6. TRANSACCIONES CONCURRENTES Y RECUPERACIÓN, 425
- 17.7. GESTIÓN DE LA MEMORIA INTERMEDIA, 427
- 17.8. FALLO CON PÉRDIDA DE ALMACENAMIENTO NO VOLÁTIL, 430
- 17.9. TÉCNICAS AVANZADAS DE RECUPERACIÓN, 430
- 17.10. SISTEMAS REMOTOS DE COPIAS DE SEGURIDAD, 435
- 17.11. RESUMEN, 437
- TÉRMINOS DE REPASO, 439
- EJERCICIOS, 440
- NOTAS BIBLIOGRÁFICAS, 441

PARTE SEXTA: ARQUITECTURA DE LOS SISTEMAS DE BASES DE DATOS

CAPÍTULO 18: ARQUITECTURAS DE LOS SISTEMAS DE BASES DE DATOS

- 18.1. ARQUITECTURAS CENTRALIZADAS Y CLIENTE-SERVIDOR, 445
- 18.2. ARQUITECTURAS DE SISTEMAS SERVIDORES, 448
- 18.3. SISTEMAS PARALELOS, 451
- 18.4. SISTEMAS DISTRIBUIDOS, 455
- 18.5. TIPOS DE REDES, 458
- 18.6. RESUMEN, 459
- TÉRMINOS DE REPASO, 460
- EJERCICIOS, 461
- NOTAS BIBLIOGRÁFICAS, 461

CAPÍTULO 19: BASES DE DATOS DISTRIBUIDAS

- 19.1. BASES DE DATOS HOMOGÉNEAS Y HETEROGENEAS, 463
- 19.2. ALMACENAMIENTO DISTRIBUIDO DE DATOS, 464
- 19.3. TRANSACCIONES DISTRIBUIDAS, 466
- 19.4. PROTOCOLOS DE COMPROMISO, 467
- 19.5. CONTROL DE LA CONCURRENCIA EN LAS BASES DE DATOS DISTRIBUIDAS, 472
- 19.6. DISPONIBILIDAD, 477
- 19.7. PROCESAMIENTO DISTRIBUIDO DE CONSULTAS, 480
- 19.8. BASES DE DATOS DISTRIBUIDAS HETEROGENEAS, 482
- 19.9. SISTEMAS DE DIRECTORIO, 484
- 19.10. RESUMEN, 487
- TÉRMINOS DE REPASO, 488
- EJERCICIOS, 489
- NOTAS BIBLIOGRÁFICAS, 491

CAPÍTULO 20: BASES DE DATOS PARALELAS

- 20.1. INTRODUCCIÓN, 493
- 20.2. PARALELISMO DE E/S, 493
- 20.3. PARALELISMO ENTRE CONSULTAS, 496
- 20.4. PARALELISMO EN CONSULTAS, 497
- 20.5. PARALELISMO EN OPERACIONES, 497
- 20.6. PARALELISMO ENTRE OPERACIONES, 502
- 20.7. DISEÑO DE SISTEMAS PARALELOS, 504
- 20.8. RESUMEN, 505
- TÉRMINOS DE REPASO, 505

EJERCICIOS, 506
NOTAS BIBLIOGRÁFICAS, 507

PARTE SÉPTIMA: OTROS TEMAS

CAPÍTULO 21: DESARROLLO DE APLICACIONES Y ADMINISTRACIÓN

21.1. INTERFACES WEB PARA BASES DE DATOS, 511
21.2. AJUSTE DEL RENDIMIENTO, 517
21.3. PRUEBAS DE RENDIMIENTO, 523
21.4. NORMALIZACIÓN, 525
21.5. COMERCIO ELECTRÓNICO, 528
21.6. SISTEMAS HEREDADOS, 530
21.7. RESUMEN, 531
TÉRMINOS DE REPASO, 531
EJERCICIOS, 532
SUGERENCIAS DE PROYECTOS, 533
NOTAS BIBLIOGRÁFICAS, 534
HERRAMIENTAS, 535

CAPÍTULO 22: CONSULTAS AVANZADAS Y RECUPERACIÓN DE INFORMACIÓN

22.1. SISTEMAS DE AYUDA A LA TOMA DE DECISIONES, 537
22.2. ANÁLISIS DE DATOS Y OLAP, 538
22.3. RECOPILACIÓN DE DATOS, 546
22.4. ALMACENAMIENTO DE DATOS, 554
22.5. SISTEMAS DE RECUPERACIÓN DE LA INFORMACIÓN, 556
22.6. RESUMEN, 563
TÉRMINOS DE REPASO, 564
EJERCICIOS, 566
NOTAS BIBLIOGRÁFICAS, 567
HERRAMIENTAS, 567

CAPÍTULO 23: TIPOS DE DATOS AUTOMÁTICOS Y NUEVAS APLICACIONES

23.1. MOTIVACIÓN, 569
23.2. EL TIEMPO EN LAS BASES DE DATOS, 570
23.3. DATOS ESPACIALES Y GEOGRÁFICOS, 571
23.4. BASES DE DATOS MULTIMEDIA, 579
23.5. COMPUTADORAS PORTÁTILES Y BASES DE DATOS PERSONALES, 581
23.6. RESUMEN, 584
TÉRMINOS DE REPASO, 585
EJERCICIOS, 586
NOTAS BIBLIOGRÁFICAS, 587

CAPÍTULO 24: PROCESAMIENTO AVANZADO DE TRANSACCIONES

24.1. MONITORES DE PROCESAMIENTO DE TRANSACCIONES, 589
24.2. FLUJOS DE TRABAJO DE TRANSACCIONES, 592
24.3. BASES DE DATOS EN MEMORIA PRINCIPAL, 596
24.4. SISTEMAS DE TRANSACCIONES DE TIEMPO REAL, 598
24.5. TRANSACCIONES DE LARGA DURACIÓN, 599
24.6. GESTIÓN DE TRANSACCIONES EN VARIAS BASES DE DATOS, 603
24.7. RESUMEN, 605
TÉRMINOS DE REPASO, 606
EJERCICIOS, 607
NOTAS BIBLIOGRÁFICAS, 608

PARTE OCTAVA: ESTUDIO DE CASOS

CAPÍTULO 25: ORACLE

- 25.1. HERRAMIENTAS PARA EL DISEÑO DE BASES DE DATOS Y LA CONSULTA, 611
- 25.2. VARIACIONES Y EXTENSIONES DE SQL, 612
- 25.3. ALMACENAMIENTO E INDEXACIÓN, 614
- 25.4. PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS, 619
- 25.5. CONTROL DE CONCURRENCIA Y RECUPERACIÓN, 623
- 25.6. ARQUITECTURA DEL SISTEMA, 625
- 25.7. RÉPLICAS, DISTRIBUCIÓN Y DATOS EXTERNOS, 626
- 25.8. HERRAMIENTAS DE GESTIÓN DE BASES DE DATOS, 627
- NOTAS BIBLIOGRÁFICAS, 628

CAPÍTULO 26: DB2 DE IBM

- 26.1. HERRAMIENTAS PARA EL DISEÑO DE BASES DE DATOS Y LA CONSULTA, 630
- 26.2. VARIACIONES Y EXTENSIONES DE SQL, 630
- 26.3. ALMACENAMIENTO E INDEXACIÓN, 631
- 26.4. PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS, 634
- 26.5. CONTROL DE CONCURRENCIA Y RECUPERACIÓN, 637
- 26.6. ARQUITECTURA DEL SISTEMA, 639
- 26.7. RÉPLICAS, DISTRIBUCIÓN Y DATOS EXTERNOS, 641
- 26.8. HERRAMIENTAS DE ADMINISTRACIÓN DE BASES DE DATOS, 641
- 26.9. RESUMEN, 642
- NOTAS BIBLIOGRÁFICAS, 643

CAPÍTULO 27: SQLSERVER DE MICROSOFT

- 27.1. HERRAMIENTAS PARA EL DISEÑO Y CONSULTA DE BASES DE DATOS, 645
- 27.2. VARIACIONES Y EXTENSIONES DE SQL, 650
- 27.3. ALMACENAMIENTO E INDEXACIÓN, 652
- 27.4. PROCESAMIENTO Y OPTIMIZACIÓN DE CONSULTAS, 654
- 27.5. CONCURRENCIA Y RECUPERACIÓN, 657
- 27.6. ARQUITECTURA DEL SISTEMA, 660
- 27.7. ACCESO A DATOS, 661
- 27.8. DISTRIBUCIÓN Y RÉPLICAS, 662
- 27.9. CONSULTAS DE TEXTO COMPLETO SOBRE DATOS RELACIONALES, 665
- 27.10. ALMACENES DE DATOS Y SERVICIOS DE ANÁLISIS, 666
- 27.11. XML Y SOPORTE DE WEB, 667
- 27.12. RESUMEN, 670
- NOTAS BIBLIOGRÁFICAS, 670

BIBLIOGRAFÍA, 673
DICIONARIO BILINGÜE, 695
ÍNDICE, 771

LA gestión de bases de datos ha evolucionado desde una aplicación informática especializada hasta una parte esencial de un entorno informático moderno y, como resultado, el conocimiento acerca de los sistemas de bases de datos se ha convertido en una parte esencial en la enseñanza de la informática. En este libro se presentan los conceptos fundamentales de la administración de bases de datos. Estos conceptos incluyen aspectos de diseño de bases de datos, lenguajes de bases de datos e implementación de sistemas de bases de datos.

Este libro está orientado a un primer curso de bases de datos para niveles técnicos y superiores. Además del material básico para un primer curso, el texto también contiene temas que pueden usarse como complemento del curso o como material introductorio de un curso avanzado.

En este libro se asume que se dispone de los conocimientos elementales sobre estructuras de datos básicas, organización de computadoras y un lenguaje de programación de alto nivel (tipo Pascal). Los conceptos se presentan usando descripciones intuitivas, muchas de las cuales están basadas en el ejemplo propuesto de una empresa bancaria. Se tratan los resultados teóricos importantes, pero se omiten las demostraciones formales. Las notas bibliográficas contienen referencias a artículos de investigación en los que los resultados se presentaron y probaron, y también referencias a material para otras lecturas. En lugar de demostraciones, se usan figuras y ejemplos para sugerir por qué se espera que los resultados en cuestión sean ciertos.

Los conceptos fundamentales y algoritmos tratados en este libro se basan habitualmente en los que se usan en la actualidad en sistemas de bases de datos existentes, comerciales o experimentales. Nuestro deseo es presentar estos conceptos y algoritmos como un conjunto general que no esté ligado a un sistema de bases de datos particular. En la Parte 8 se discuten detalles de sistemas de bases de datos comerciales.

En esta cuarta edición de *Fundamentos de bases de datos* se ha mantenido el estilo global de las primeras tres ediciones, a la vez que se ha tenido en cuenta la evolución de la gestión de bases de datos. Se han añadido varios capítulos nuevos para tratar nuevas tecnologías. Cada capítulo se ha corregido y la mayoría se ha modificado ampliamente. Se describirán los cambios con detalle en breve.

ORGANIZACIÓN

El texto está organizado en ocho partes principales más dos apéndices:

- **Visión general** (Capítulo 1). En el Capítulo 1 se proporciona una visión general de la naturaleza y propósito de los sistemas de bases de datos. Se explica cómo se ha desarrollado el concepto de sistema de bases de datos, cuáles son las características usuales de los sistemas de bases de datos, lo que proporciona al usuario un sistema de bases de datos y cómo un sistema de bases de datos se comunica con los sistemas operativos. También se introduce una aplicación de bases de datos de ejemplo: una empresa bancaria que consta de muchas sucursales. Este ejemplo se usa a lo largo de todo el libro. Este capítulo es histórico, explicativo y motivador por naturaleza.
- **Modelos de datos** (Capítulos 2 y 3). En el Capítulo 2 se presenta el modelo entidad-relación. Este modelo proporciona una visión de alto nivel de los resultados de un diseño de base de datos y de los problemas que se encuentran en la captura de la semántica de las aplicaciones realistas que contienen las restricciones de un modelo de datos. El Capítulo 3 se centra en el modelo de datos relacional, tratando la relevancia del álgebra relacional y el cálculo relacional.
- **Bases de datos relacionales** (Capítulos 4 al 7). El Capítulo 4 se centra en el lenguaje relacional orientado al usuario de mayor influencia: SQL. El Capítulo 5 cubre otros dos lenguajes relacionales, QBE y Datalog. En estos dos capítulos se describe la manipulación de datos: consultas, actualizaciones, inserciones y borrados. Los algoritmos y las cuestiones de diseño

se relegan a capítulos posteriores. Así, estos capítulos son adecuados para aquellas personas o para las clases de nivel más bajo en donde se deseé aprender qué es un sistema de bases de datos, sin entrar en detalles sobre los algoritmos internos y estructuras que contienen.

En el Capítulo 6 se presentan las restricciones desde el punto de vista de la integridad de las bases de datos. En el Capítulo 7 se muestra cómo se pueden usar las restricciones en el diseño de una base de datos relacional. En el Capítulo 6 se presentan la integridad referencial; mecanismos para el mantenimiento de la integridad, tales como disparadores y asertos, y mecanismos de autorización. El tema de este capítulo es la protección de las bases de datos contra daños accidentales y daños intencionados.

En el Capítulo 7 se introduce la teoría del diseño de bases de datos relacionales. Se trata la teoría de las dependencias funcionales y la normalización, con énfasis en la motivación y el significado intuitivo de cada forma normal. También se describe en detalle el proceso de diseño de bases de datos.

- **Bases de datos basadas en objetos y XML** (Capítulos 8 al 10). El Capítulo 8 trata las bases de datos orientadas a objetos. En él se introducen los conceptos de la programación orientada a objetos y se muestra cómo estos conceptos constituyen la base para un modelo de datos. No se asume un conocimiento previo de lenguajes orientados a objetos. El Capítulo 9 trata las bases de datos relacionales de objetos, y muestra cómo la norma SQL:1999 extiende el modelo de datos relacional para incluir características de la programación orientada a objetos, tales como la herencia, los tipos complejos y la identidad de objeto.

En el Capítulo 10 se trata la norma XML para representación de datos, el cual está experimentando un uso cada vez mayor en la comunicación de datos y en el almacenamiento de tipos de datos complejos. El capítulo también describe lenguajes de consulta para XML.

- **Almacenamiento de datos y consultas** (Capítulos 11 al 14). En el Capítulo 11 se estudian los discos, archivos y estructuras de un sistema de archivos y la correspondencia de datos relacionales y de objetos con un sistema de archivos. En el Capítulo 12 se presentan varias técnicas de acceso a los datos, incluyendo la asociación, los índices de árboles B+ y los índices de archivos en retícula. Los capítulos 13 y 14 tratan los algoritmos de evaluación de consultas y optimización de consultas basados en transformación de consultas preservando la equivalencia.

Estos capítulos están orientados a personas que desean conocer los componentes de almacenamiento y consulta internos de una base de datos.

- **Gestión de transacciones** (Capítulos 15 al 17). El Capítulo 15 se centra en los fundamentos de un sistema de procesamiento de transacciones, incluyendo la atomicidad de las transacciones, la consistencia, el aislamiento y la durabilidad, y también la noción de secuencialidad.

El Capítulo 16 se centra en el control de concurrencia y se presentan varias técnicas que aseguran la secuencialidad, incluyendo los bloqueos, las marcas temporales y técnicas optimistas (de validación). Los temas de interbloqueo se tratan también en este capítulo. El Capítulo 17 aborda las técnicas principales para asegurar la ejecución correcta de transacciones a pesar de las caídas del sistema y los fallos de disco. Estas técnicas incluyen el registro histórico, paginación en la sombra, puntos de revisión y volcados de la base de datos.

- **Arquitectura de un sistema de bases de datos** (Capítulos 18 al 20). El Capítulo 18 trata la arquitectura de un sistema informático y en él se describe la influencia de los sistemas informáticos subyacentes en los sistemas de bases de datos. Se discuten los sistemas centralizados, los sistemas cliente-servidor, las arquitecturas paralelas y distribuidas, y los tipos de redes. En el Capítulo 19 se estudian los sistemas de bases de datos distribuidas, revisando los aspectos de diseño de bases de datos, gestión de las transacciones y evaluación y optimización de consultas en el contexto de los sistemas de bases de datos distribuidas. El capítulo también trata aspectos de la disponibilidad del sistema durante fallos y describe el sistema de directorios LDAP.

En el capítulo 20, acerca de las bases de datos paralelas, se exploran varias técnicas de parallelización, incluyendo paralelismo de E/S, paralelismo entre consultas y en consultas, y paralelismo entre operaciones y en operaciones. También se describe el diseño de sistemas paralelos.

- **Otros temas** (Capítulos 21 al 24). El Capítulo 21 trata el desarrollo y administración de aplicaciones de bases de datos. Los temas incluyen las interfaces de las bases de datos, en particular las interfaces Web, el ajuste de rendimiento, los programas de prueba, la estandarización y los aspectos de las bases de datos en el comercio electrónico. El Capítulo 22 presenta

técnicas de consulta, incluyendo sistemas de ayuda a la toma de decisiones y recuperación de la información. Los temas tratados en el área de la ayuda a la toma de decisiones incluyen las técnicas de procesamiento analítico interactivo (OLAP, Online Analytical Processing), el soporte de SQL:1999 para OLAP, recopilación de datos y almacenes de datos. El capítulo también describe técnicas de recuperación de información para la consulta de datos textuales, incluyendo técnicas basadas en hipervínculos usadas en los motores de búsqueda Web.

El Capítulo 23 trata tipos de datos avanzados y nuevas aplicaciones, incluyendo datos temporales, datos espaciales y geográficos, datos multimedia, y aspectos de la gestión de las bases de datos móviles y personales. Finalmente, el Capítulo 24 trata el procesamiento avanzado de transacciones. Se estudian los monitores de procesamiento de transacciones, los sistemas de transacciones de alto rendimiento, los sistemas de transacciones de tiempo real, y los flujos de datos transaccionales.

- **Estudios de casos** (Capítulos 25 al 27). En esta parte presentamos estudios de casos de tres sistemas de bases de datos comerciales: Oracle, IBM DB2 y Microsoft SQL Server. Estos capítulos esbozan características únicas de cada uno de los productos y describen su estructura interna. Proporcionan una gran cantidad de información interesante sobre los productos respectivos, y ayudan a ver cómo las diferentes técnicas de implementación descritas en las partes anteriores se usan en sistemas reales. También se tratan aspectos prácticos en el diseño de sistemas reales.
- **Apéndices en línea.** Aunque la mayoría de las aplicaciones de bases de datos modernas usen, bien el modelo relacional o bien el modelo orientado a objetos, los modelos de datos de redes y jerárquico están en uso todavía. En beneficio de los lectores que deseen aprender estos modelos de datos se proporcionan apéndices que describen los modelos de redes y jerárquico, en los Apéndices A y B, respectivamente. Los apéndices sólo están disponibles en Internet (<http://www.bell-labs.com/topic/books/db-book>).

El Apéndice C describe el diseño avanzado de bases de datos relacionales, incluyendo la teoría de dependencias multivaloradas ¿Multivaluadas?, las dependencias de reunión y las formas normales de proyección-reunión y dominio-clave. Este apéndice es útil para quienes deseen el tratamiento del diseño de bases de datos relacionales en más detalle, y para profesores que deseen explicarlo en sus asignaturas. Este apéndice está también sólo disponible en Internet, en la página Web del libro.

LA CUARTA EDICIÓN

La producción de esta cuarta edición se ha guiado por muchos comentarios y sugerencias referidos a las ediciones anteriores, junto con las propias observaciones en la enseñanza en el IIT de Bombay, y por el análisis de las direcciones que la tecnología de bases de datos está tomando.

El procedimiento básico fue reescribir el material en cada capítulo, actualizando el material más antiguo, añadiendo discusiones en desarrollos recientes en la tecnología de bases de datos, y mejorando las descripciones de los temas que los estudiantes encontraron difíciles de comprender. Cada capítulo tiene ahora una lista de términos de repaso, que pueden ayudar a asimilar los temas clave tratados en cada capítulo. Se ha añadido también una nueva sección al final de la mayoría de los capítulos que proporciona información sobre herramientas software referidas al tema del capítulo. También se han añadido nuevos ejercicios y se han actualizado las referencias.

Se ha incluido un nuevo capítulo que trata XML, y tres capítulos de estudio de los sistemas de bases de datos comerciales líderes: Oracle, IBM DB2 y Microsoft SQL Server.

Los capítulos se han organizado en varias partes y se han reorganizado los contenidos de varios de ellos. En beneficio de aquellos lectores familiarizados con la tercera edición se explican a continuación los principales cambios.

- **Modelo entidad-relación.** Se ha mejorado el tratamiento del modelo entidad-relación (E-R). Se han añadido nuevos ejemplos y algunos se han cambiado para dar una mejor intuición al lector. Se ha incluido un resumen de notaciones E-R alternativas, junto con un nuevo apartado sobre UML.
- **Bases de datos relacionales.** El tratamiento de SQL en el Capítulo 4 ahora se refiere al estándar SQL:1999, que se aprobó después de la publicación de la tercera edición de este libro. El

tratamiento de SQL se ha ampliado significativamente para incluir la cláusula **with**, para un tratamiento ampliado de SQL incorporado y el tratamiento de ODBC y JDBC, cuyo uso ha aumentado notablemente en los últimos años. La parte del capítulo 5 dedicada a Quel se ha eliminado, ya que no se usa ampliamente debido al poco uso que actualmente se hace de este lenguaje. El tratamiento de QBE se ha revisado para eliminar algunas ambigüedades y para añadir el tratamiento de la versión de QBE usada en la base de datos Microsoft Access.

El Capítulo 6 trata ahora de las restricciones de integridad y de la seguridad. El tratamiento de la seguridad, ubicado en la edición anterior en el Capítulo 19, se ha trasladado al Capítulo 6. El Capítulo 6 también trata los disparadores. El Capítulo 7 aborda el diseño de las bases de datos relacionales y las formas normales. La discusión de las dependencias funcionales, ubicada en la edición anterior en el Capítulo 6, se ha trasladado al Capítulo 7. El Capítulo 7 se ha remodelado significativamente, proporcionando varios algoritmos para las dependencias funcionales y un tratamiento extendido del proceso general del diseño de bases de datos. Los axiomas para la inferencia de las dependencias multivaloradas, las formas normales FNRP y FNCD se han trasladado al apéndice.

- **Bases de datos basadas en objetos.** Se ha mejorado el tratamiento de la orientación a objetos del Capítulo 8, y se ha actualizado la discusión de ODMG. Se ha actualizado el tratamiento de las bases de datos relacionales orientadas a objetos del Capítulo 9 y, en particular, el estándar SQL:1999, reemplaza a SQL extendido usado en la tercera edición.
- **XML.** El Capítulo 10, que trata XML, es un nuevo capítulo de la cuarta edición.
- **Almacenamiento, indexación y procesamiento de consultas.** Se ha actualizado el tratamiento del almacenamiento y de las estructuras de archivos del Capítulo 11; este fue el Capítulo 10 en la tercera edición. Muchas características de las unidades de disco y de otros mecanismos de almacenamiento han cambiado en gran medida con el paso de los años, y su tratamiento se ha actualizado correspondientemente. El tratamiento de RAID se ha actualizado para reflejar las tendencias tecnológicas. El tratamiento de diccionarios de datos (catálogos) se ha extendido.

El Capítulo 12, sobre indexación, incluye ahora el estudio de los índices de mapa de bits; este capítulo fue el Capítulo 11 en la tercera edición. El algoritmo de inserción en árboles B⁺ se ha simplificado y se ha proporcionado un pseudocódigo para su examen. La asociación dividida se ha eliminado, ya que no tiene un uso significativo.

El tratamiento del procesamiento de consultas se ha reorganizado, con el capítulo anterior (Capítulo 12 en la tercera edición) dividido en dos capítulos, uno sobre procesamiento de consultas (Capítulo 13) y otro sobre optimización de consultas (Capítulo 14). Todos los detalles referidos a la estimación de costes y a la optimización de consultas se han trasladado al Capítulo 14, permitiendo al Capítulo 13 centrarse en los algoritmos de procesamiento de consultas. Se han eliminado varias fórmulas detalladas (y tediosas) para el cálculo del número exacto de operaciones de E/S para diferentes operaciones. El Capítulo 14 se presenta ahora con un pseudocódigo para la optimización de algoritmos, y nuevos apartados sobre la optimización de subconsultas anidadas y sobre vistas materializadas.

- **Procesamiento de transacciones.** El Capítulo 15, que proporciona una introducción a las transacciones, se ha actualizado; este capítulo era el Capítulo 13 en la tercera edición. Se han eliminado los tests de secuenciaabilidad.

El Capítulo 16, sobre el control de concurrencia, incluye un nuevo apartado sobre la implementación de los gestores de bloqueo, y otro sobre los niveles débiles de consistencia, que estaban en el Capítulo 20 de la tercera edición. Se ha ampliado el control de concurrencia de estructuras de índices, proporcionando detalles del protocolo cangrejo, que es una alternativa más simple al protocolo de enlace B, y el bloqueo de siguiente clave para evitar el problema fantasma. El Capítulo 17, que trata sobre recuperación, incluye ahora un estudio del algoritmo de recuperación ARIES. Este capítulo trata ahora los sistemas de copia de seguridad remota para proporcionar una alta disponibilidad a pesar de los fallos, característica cada vez más importante en las aplicaciones «24x7».

Como en la tercera edición, esta organización permite a los profesores elegir entre conceptos de procesamiento de transacciones introductorios únicamente (cubiertos sólo en el Capítulo 15) u ofrecer un conocimiento detallado (basado en los Capítulos 15 al 17).

- **Arquitecturas de sistemas de bases de datos.** El Capítulo 18, que proporciona una visión general de las arquitecturas de sistemas de bases de datos, se ha actualizado para tratar la tecnología actual; esto se encontraba en el Capítulo 16 de la tercera edición. El orden del capí-

tulo de bases de datos paralelas y de los capítulos de bases de datos distribuidas se ha intercambiado. Mientras que el tratamiento de las técnicas de procesamiento de consultas de bases de datos del Capítulo 20 (que fue el Capítulo 16 en la tercera edición) es de primordial interés para quienes deseen aprender los interiores de las bases de datos, las bases de datos distribuidas, ahora tratadas en el Capítulo 19, son un tema más fundamental con el que debería estar familiarizado cualquiera que trabaje con bases de datos.

El Capítulo 19 sobre bases de datos distribuidas se ha rehecho significativamente para reducir el énfasis en la denominación y la transparencia, y para aumentar el tratamiento de la operación durante fallos, incluyendo las técnicas de control de concurrencia para proporcionar alta disponibilidad. El tratamiento del protocolo de compromiso de tres fases se ha abreviado, al tener detección distribuida de interbloqueos globales, ya que no se usa mucho en la práctica. El estudio de los aspectos de procesamiento de consultas se ha trasladado del Capítulo 20 de la tercera edición. Hay un nuevo apartado sobre los sistemas de directorio, en particular LDAP, ya que se usan ampliamente como un mecanismo para hacer disponible la información en una configuración distribuida.

- **Otros temas.** Aunque se ha modificado y actualizado el texto completo, nuestra presentación del material que tiene relación con el continuo desarrollo de bases de datos y las nuevas aplicaciones de bases de datos se tratan en cuatro nuevos capítulos, del Capítulo 21 al 24.

El Capítulo 21 es nuevo en la cuarta edición y trata el desarrollo y administración de aplicaciones. La descripción de la construcción de interfaces Web para bases de datos, incluyendo servlets y otros mecanismos para las secuencias de comandos para el lado del servidor, es nueva. La sección sobre ajuste de rendimiento, que estaba anteriormente en el Capítulo 19, tiene nuevo material sobre la famosa regla de 5 minutos y sobre la regla de 1 minuto, así como algunos nuevos ejemplos. El tratamiento de la selección de vistas materializadas también es nuevo. El tratamiento de los programas de prueba y de los estándares se ha actualizado. Hay una nueva sección sobre comercio electrónico, centrándose en los aspectos de las bases de datos en el comercio electrónico y un nuevo apartado que trata los sistemas heredados.

El Capítulo 22, que trata consultas avanzadas y recuperación de la información, incluye nuevo material sobre OLAP, particularmente sobre las extensiones de SQL:1999 para análisis de datos. El estudio de los almacenes de datos y de la recopilación de datos también se ha ampliado en gran medida. El tratamiento de la recuperación de la información se ha aumentado significativamente, en particular en el área de la búsqueda Web. Las versiones anteriores de este material estaban en el Capítulo 21 de la tercera edición.

El Capítulo 23, que trata tipos de datos avanzados y nuevas aplicaciones, contiene material sobre datos temporales, datos espaciales, datos multimedia y bases de datos móviles. Este material es una versión actualizada del material que se encontraba en el Capítulo 21 de la tercera edición. El Capítulo 24, que trata el procesamiento de transacciones avanzado, contiene versiones actualizadas de los monitores TP, sistemas de flujo de datos, bases de datos en memoria principal y de tiempo real, transacciones de larga duración y gestión de transacciones en múltiples bases de datos, que aparecieron en el Capítulo 20 de la tercera edición.

NOTA PARA EL PROFESOR

El libro contiene tanto material básico como material avanzado, que podría no ser abordado en un único semestre. Se han marcado varios apartados como avanzados, usando el símbolo «**». Estos apartados se pueden omitir, si se desea, sin pérdida de continuidad.

Es posible diseñar cursos usando varios subconjuntos de los capítulos. A continuación se muestran varias posibilidades:

- El Capítulo 5 se puede omitir si los estudiantes no van a usar QBE o Datalog como parte del curso.
- Si la programación orientada a objetos se va a tratar en un curso avanzado por separado, los Capítulos 8 y 9 y el Apartado 11.9 se pueden omitir. Alternativamente, con ellos se puede constituir la base de un curso avanzado de bases de datos orientadas a objetos.
- El Capítulo 10 (XML) y el Capítulo 14 (optimización de consultas) se pueden omitir para un curso introductorio.

- Tanto el tratamiento del procesamiento de transacciones (Capítulos 15 al 17) como de la arquitectura de sistemas de bases de datos (Capítulos 18 al 20) poseen un capítulo de visión de conjunto (Capítulos 15 y 18 respectivamente), seguidos de capítulos más detallados. Se podría elegir usar los Capítulos 15 y 18, omitiendo los Capítulos 16, 17, 19 y 20, si se relegan estos capítulos para un curso avanzado.
- Los Capítulos 21 al 24 son adecuados para un curso avanzado o para autoaprendizaje de los estudiantes, aunque el apartado 21.1 se puede tratar en un primer curso de bases de datos.

Se puede encontrar un modelo de plan de estudios del curso, a partir del texto, en la página inicial Web del libro (véase el siguiente apartado).

PÁGINA WEB Y SUPLEMENTOS PARA LA ENSEÑANZA

Está disponible una página World Wide Web para este libro en el URL:

<http://www.bell-labs.com/topic/books/db-book>

La página Web contiene:

- Transparencias de todos los capítulos del libro.
- Respuestas a ejercicios seleccionados.
- Los tres apéndices.
- Una lista de erratas actualizada.
- Material suplementario proporcionado por usuarios del libro.

Se puede proporcionar un manual completo de soluciones sólo a las facultades. Para obtener más información sobre cómo obtener una copia del manual de soluciones envíe por favor un correo electrónico a customer.service@mcgraw-hill.com. En los Estados Unidos se puede llamar al 800-338-3987. La página Web de McGraw-Hill para este libro es:

<http://www.mcgraw-hill.es/olc/silberschatz>

CÓMO CONTACTAR CON LOS AUTORES Y OTROS USUARIOS

Se ha creado una lista de correo electrónico con la que los usuarios de este libro pueden comunicarse entre sí y con los autores. Si desea incorporarse a la lista, por favor mande un mensaje a db-book@research.bell-labs.com, incluyendo su nombre, afiliación, puesto y dirección de correo electrónico.

Nos hemos esforzado para eliminar erratas y problemas del texto, pero, como en los nuevos desarrollos de software, probablemente queden fallos; hay una lista de erratas actualizada accesible desde la página inicial del libro*. Agradeceríamos que se nos notificara cualquier error u omisión del libro que no se encuentre en la lista actual de erratas.

También desearíamos recibir sugerencias sobre la mejora del libro. Damos la bienvenida a aquellas contribuciones a la página Web del libro que pudiesen ser usadas por otros lectores, como ejercicios de programación, sugerencias sobre proyectos, laboratorios y tutoriales en línea, y consejos de enseñanza.

El correo electrónico se deberá dirigir a db-book@research.bell-labs.com. Cualquier otra correspondencia se debe enviar a Avi Silberschatz, Bell Laboratories, Room 2T-310, 600 Mountain Avenue, Murray Hill, NJ 07974, EE.UU.

* N. del T. Todas las erratas de la versión original en inglés incluidas en esta página Web en el momento de finalizar la traducción se han corregido en este libro.

AGRADECIMIENTOS

Esta edición se ha beneficiado de los muchos y útiles comentarios que nos han proporcionado los muchos estudiantes que han usado la tercera edición. Además, gran cantidad de personas nos han escrito o hablado acerca del libro, y nos han ofrecido sugerencias y comentarios. Aunque no podemos mencionar aquí a todas, agradecemos especialmente a las siguientes:

- Phil Bernhard, Instituto de Tecnología de Florida; Eitan M. Gurari, Universidad del estado de Ohio; Irwin Levinstein, Universidad Old Dominion; Ling Liu, Instituto de Tecnología de Georgia; Ami Motro, Universidad George Mason; Bhagirath Narahari, Meral Ozsoyoglu, Universidad Case Western Reserve; y Odinaldo Rodríguez, King's College de Londres; que sirvieron como revisores del libro y cuyos comentarios nos ayudaron en gran medida en la formulación de esta cuarta edición.
- Soumen Chakrabarti, Sharad Mehrotra, Krithi Ramamirtham, Mike Reiter, Sunita Sarawagi, N. L. Sarda y Dilys Thomas, por su amplia y valiosa realimentación sobre varios capítulos del libro.
- Phil Bohannon, por escribir el primer borrador del Capítulo 10 describiendo XML.
- Hakan Jakobsson (Oracle), Sriram Padmanabhan (IBM) y César Galindo-Legareia, Goetz Graefe, José A. Blakeley, Kalen Delaney, Michael Rys, Michael Zwilling, Sameet Agarwal, Thomas Casey (todos de Microsoft), por escribir los apéndices sobre los sistemas de bases de datos Oracle, IBM DB2 y Microsoft SQL Server.
- Yuri Breitbart, por su ayuda con el capítulo sobre bases de datos distribuidas; Mark Reiter, por su ayuda con los apartados de seguridad; y Jim Melton, por las aclaraciones sobre SQL:1999.
- Marilyn Turnamian y Nandprasad Joshi, cuya excelente asistencia secretarial fue esencial para terminar a tiempo esta cuarta edición.

La editora fue Betsy Jones. El editor de desarrollo senior fue Kelly Butcher. El director de proyecto fue Jill Peter. El director de marketing ejecutivo fue John Wannemacher. El ilustrador de la portada fue Paul Tumbaugh mientras que el diseñador de la portada fue JoAnne Schopler. El editor de copia fue George Watson. El corrector de pruebas fue Marie Zartman. El productor del material complementario fue Jodi Banowetz. El diseñador fue Rick Noel. El indexador fue Tobiah Waldron.

Esta edición está basada en las tres ediciones previas, así que los autores dan las gracias una vez más a las muchas personas que ayudaron con las tres primeras ediciones, incluyendo a R.B. Abhyankar, Don Batory, Haran Boral, Paul Bourgeois, Robert Brazile, Michael Carey, J. Edwards, Christos Faloutsos, Homma Farian, Alan Fekete, Shashi Gadia, Jim Gray, Le Gruenwald, Yannis Ioannidis, Hyoung-Joo Kim, Henry Korth (padre de Henry F.), Carol Kroll, Gary Lindsrom, Dave Maier, Keith Marzullo, Fletcher Mattox, Alberto Mendelzon, Héctor García-Molina, Ami Motro, Anil Nigam, Cyril Orji, Bruce Porter, Jim Peterson, K.V. Raghavan, Mark Roth, Marek Rusinkiewicz, S. Seshadri, Shashi Shekhar, Amit Sheth, Nandit Soparkar, Greg Speegle y Marianne Winslett. Lyn Dupré editó y corrigió la tercera edición del libro y Sara Strandtmann editó el texto de la tercera edición. Greg Speegle, Dawn Bezwiner y K. V. Raghavan nos ayudaron a preparar el manual del profesor para ediciones anteriores. La nueva portada es una evolución de las portadas de las tres primeras ediciones. Marilyn Turnamian creó un primer boceto del diseño de la portada para esta edición. La idea de usar barcos como parte del concepto de la portada fue sugerida originalmente por Bruce Stephan.

Finalmente, Sudarshan desearía agradecer a su esposa, Sita, por su amor y apoyo, a su hijo de dos años Madhur por su amor, y a su madre, Indira, por su apoyo. Hank desearía agradecer a su esposa, Joan, y a sus hijos, Abby y Joe, por su amor y comprensión. Avi desearía agradecer a su esposa Haya y a su hijo Aaron por su paciencia y apoyo durante la revisión de este libro.

A.S.

H.F.K.

S.S.

Un sistema gestor de bases de datos (SGBD) consiste en una colección de datos interrelacionados y un conjunto de programas para acceder a dichos datos. La colección de datos, normalmente denominada **base de datos**, contiene información relevante para una empresa. El objetivo principal de un SGBD es proporcionar una forma de almacenar y recuperar la información de una base de datos de manera que sea tanto *práctica* como *eficiente*.

Los sistemas de bases de datos se diseñan para gestionar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para almacenar la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la fiabilidad de la información almacenada, a pesar de las caídas del sistema o los intentos de acceso sin autorización. Si los datos van a ser compartidos entre diversos usuarios, el sistema debe evitar posibles resultados anómalos.

Dado que la información es tan importante en la mayoría de las organizaciones, los científicos informáticos han desarrollado un amplio conjunto de conceptos y técnicas para la gestión de los datos. En este capítulo se presenta una breve introducción a los principios de los sistemas de bases de datos.

1.1. APLICACIONES DE LOS SISTEMAS DE BASES DE DATOS

Las bases de datos son ampliamente usadas. Las siguientes son algunas de sus aplicaciones más representativas:

- **Banca.** Para información de los clientes, cuentas y préstamos, y transacciones bancarias.
- **Líneas aéreas.** Para reservas e información de planificación. Las líneas aéreas fueron de los primeros en usar las bases de datos de forma distribuida geográficamente (los terminales situados en todo el mundo accedían al sistema de bases de datos centralizado a través de las líneas telefónicas y otras redes de datos).
- **Universidades.** Para información de los estudiantes, matrículas de las asignaturas y cursos.
- **Transacciones de tarjetas de crédito.** Para compras con tarjeta de crédito y generación mensual de extractos.
- **Telecomunicaciones.** Para guardar un registro de las llamadas realizadas, generación mensual de facturas, manteniendo el saldo de las tarjetas telefónicas de prepago y para almacenar información sobre las redes de comunicaciones.
- **Finanzas.** Para almacenar información sobre grandes empresas, ventas y compras de documentos formales financieros, como bolsa y bonos.
- **Ventas.** Para información de clientes, productos y compras.

- **Producción.** Para la gestión de la cadena de producción y para el seguimiento de la producción de elementos en las factorías, inventarios de elementos en almacenes y pedidos de elementos.

- **Recursos humanos.** Para información sobre los empleados, salarios, impuestos y beneficios, y para la generación de las nóminas.

Como esta lista ilustra, las bases de datos forman una parte esencial de casi todas las empresas actuales.

A lo largo de las últimas cuatro décadas del siglo veinte, el uso de las bases de datos creció en todas las empresas. En los primeros días, muy pocas personas interactuaron directamente con los sistemas de bases de datos, aunque sin darse cuenta interactuaron con bases de datos indirectamente (con los informes impresos como extractos de tarjetas de crédito, o mediante agentes como cajeros de bancos y agentes de reserva de líneas aéreas). Después vinieron los cajeros automáticos y permitieron a los usuarios interactuar con las bases de datos. Las interfaces telefónicas con los computadores (sistemas de respuesta vocal interactiva) también permitieron a los usuarios manejar directamente las bases de datos. Un llamador podía marcar un número y pulsar teclas del teléfono para introducir información o para seleccionar opciones alternativas, para determinar las horas de llegada o salida, por ejemplo, o para matricularse de asignaturas en una universidad.

La revolución de Internet a finales de la década de 1990 aumentó significativamente el acceso directo del

usuario a las bases de datos. Las organizaciones convirtieron muchas de sus interfaces telefónicas a las bases de datos en interfaces Web, y pusieron disponibles en línea muchos servicios. Por ejemplo, cuando se accede a una tienda de libros en línea y se busca un libro o una colección de música se está accediendo a datos almacenados en una base de datos. Cuando se solicita un pedido en línea, el pedido se almacena en una base de datos. Cuando se accede a un banco en un sitio Web y se consulta el estado de la cuenta y los movimientos, la información se recupera del sistema de bases de datos del banco. Cuando se accede a un sitio Web, la información personal puede ser recuperada de una base de datos para seleccionar los anuncios que se deberían mostrar. Más aún, los datos sobre

los accesos Web pueden ser almacenados en una base de datos.

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

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

1.2. SISTEMAS DE BASES DE DATOS FRENTE A SISTEMAS DE ARCHIVOS

Considérese parte de una empresa de cajas de ahorros que mantiene información acerca de todos los clientes y cuentas de ahorros. Una manera de mantener la información en un computador es almacenarla en archivos del sistema operativo. Para permitir a los usuarios manipular la información, el sistema tiene un número de programas de aplicación que manipula los archivos, incluyendo:

- Un programa para efectuar cargos o abonos en una cuenta.
- Un programa para añadir una cuenta nueva.
- Un programa para calcular el saldo de una cuenta.
- Un programa para generar las operaciones mensuales.

Estos programas de aplicación se han escrito por programadores de sistemas en respuesta a las necesidades de la organización bancaria.

Si las necesidades se incrementan, se añaden nuevos programas de aplicación al sistema. Por ejemplo, supóngase que las regulaciones de un nuevo gobierno permiten a las cajas de ahorros ofrecer cuentas corrientes. Como resultado se crean nuevos archivos permanentes que contengan información acerca de todas las cuentas corrientes mantenidas por el banco, y puede ser necesario escribir nuevos programas de aplicación para tratar situaciones que no existían en las cuentas de ahorro, tales como manejar descubiertos. Así, sobre la marcha, se añaden más archivos y programas de aplicación al sistema.

Este **sistema de procesamiento de archivos** típico que se acaba de describir se mantiene mediante un sistema operativo convencional. Los registros permanentes son almacenados en varios archivos y se escriben diferentes programas de aplicación para extraer registros y para añadir registros a los archivos adecuados. Antes de la llegada de los sistemas de gestión de bases de datos (SGBDs), las organizaciones normalmente han almacenado la información usando tales sistemas.

Mantener información de la organización en un sistema de procesamiento de archivos tiene una serie de inconvenientes importantes:

- **Redundancia e inconsistencia de datos.** Debido a que los archivos y programas de aplicación son creados por diferentes programadores en un largo período de tiempo, los diversos archivos tienen probablemente diferentes formatos y los programas pueden estar escritos en diferentes lenguajes. Más aún, la misma información puede estar duplicada en diferentes lugares (archivos). Por ejemplo, la dirección y número de teléfono de un cliente particular puede aparecer en un archivo que contenga registros de cuentas de ahorros y en un archivo que contenga registros de una cuenta corriente. Esta redundancia conduce a un almacenamiento y coste de acceso más altos. Además, puede conducir a **inconsistencia de datos**; es decir, las diversas copias de los mismos datos pueden no coincidir. Por ejemplo, un cambio en la dirección del cliente puede estar reflejado en los registros de las cuentas de ahorro pero no estarlo en el resto del sistema.
- **Dificultad en el acceso a los datos.** Supóngase que uno de los empleados del banco necesita averiguar los nombres de todos los clientes que viven en el distrito postal 28733 de la ciudad. El empleado pide al departamento de procesamiento de datos que genere dicha lista. Debido a que esta petición no fue prevista cuando el sistema original fue diseñado, no hay un programa de aplicación a mano para satisfacerla. Hay, sin embargo, un programa de aplicación que genera la lista de *todos* los clientes. El empleado del banco tiene ahora dos opciones: bien obtener la lista de todos los clientes y obtener la información que necesita manualmente, o bien pedir al departamento de procesamiento de datos que haga

que un programador de sistemas escriba el programa de aplicación necesario. Ambas alternativas son obviamente insatisfactorias. Supóngase que se escribe tal programa y que, varios días más tarde, el mismo empleado necesita arreglar esa lista para incluir sólo aquellos clientes que tienen una cuenta con saldo de 10.000 € o más. Como se puede esperar, un programa para generar tal lista no existe. De nuevo, el empleado tiene que elegir entre dos opciones, ninguna de las cuales es satisfactoria.

La cuestión aquí es que el entorno de procesamiento de archivos convencional no permite que los datos necesarios sean obtenidos de una forma práctica y eficiente. Se deben desarrollar sistemas de recuperación de datos más interesantes para un uso general.

- **Aislamiento de datos.** Debido a que los datos están dispersos en varios archivos, y los archivos pueden estar en diferentes formatos, es difícil escribir nuevos programas de aplicación para recuperar los datos apropiados.
- **Problemas de integridad.** Los valores de los datos almacenados en la base de datos deben satisfacer ciertos tipos de **restricciones de consistencia**. Por ejemplo, el saldo de una cuenta bancaria no puede nunca ser más bajo de una cantidad predeterminada (por ejemplo 25 €). Los desarrolladores hacen cumplir esas restricciones en el sistema añadiendo el código apropiado en los diversos programas de aplicación. Sin embargo, cuando se añaden nuevas restricciones, es difícil cambiar los programas para hacer que se cumplan. El problema es complicado cuando las restricciones implican diferentes elementos de datos de diferentes archivos.
- **Problemas de atomicidad.** Un sistema de un computador, como cualquier otro dispositivo mecánico o eléctrico, está sujeto a fallo. En muchas aplicaciones es crucial asegurar que, una vez que un fallo ha ocurrido y se ha detectado, los datos se restauran al estado de consistencia que existía antes del fallo. Consideremos un programa para transferir 50 € desde la cuenta A a la B. Si ocurre un fallo del sistema durante la ejecución del programa, es posible que los 50 € fueron eliminados de la cuenta A pero no abonados a la cuenta B, resultando un estado de la base de datos inconsistente. Claramente, es esencial para la consistencia de la base de datos que ambos, el abono y el cargo tengan lugar, o que ninguno tenga lugar. Es decir, la trans-

ferencia de fondos debe ser *atómica*: ésta debe ocurrir en ellos por completo o no ocurrir en absoluto. Es difícil asegurar esta propiedad en un sistema de procesamiento de archivos convencional.

- **Anomalías en el acceso concurrente.** Conforme se ha ido mejorando el conjunto de ejecución de los sistemas y ha sido posible una respuesta en tiempo más rápida, muchos sistemas han ido permitiendo a múltiples usuarios actualizar los datos simultáneamente. En tales sistemas un entorno de interacción de actualizaciones concurrentes puede dar lugar a datos inconsistentes. Considérese una cuenta bancaria A, que contiene 500 €. Si dos clientes retiran fondos (por ejemplo 50 € y 100 € respectivamente) de la cuenta A en aproximadamente el mismo tiempo, el resultado de las ejecuciones concurrentes puede dejar la cuenta en un estado incorrecto (o inconsistente). Supongamos que los programas se ejecutan para cada retirada y escriben el resultado después. Si los dos programas funcionan concurrentemente, pueden leer ambos el valor 500 €, y escribir después 450 € y 400 €, respectivamente. Dependiendo de cuál escriba el último valor, la cuenta puede contener bien 450 € o bien 400 €, en lugar del valor correcto, 350 €. Para protegerse contra esta posibilidad, el sistema debe mantener alguna forma de supervisión. Sin embargo, ya que se puede acceder a los datos desde muchos programas de aplicación diferentes que no han sido previamente coordinados, la supervisión es difícil de proporcionar.
- **Problemas de seguridad.** No todos los usuarios de un sistema de bases de datos deberían poder acceder a todos los datos. Por ejemplo, en un sistema bancario, el personal de nóminas necesita ver sólo esa parte de la base de datos que tiene información acerca de varios empleados del banco. No necesitan acceder a la información acerca de las cuentas de clientes. Como los programas de aplicación se añaden al sistema de una forma ad hoc, es difícil garantizar tales restricciones de seguridad.

Estas dificultades, entre otras, han motivado el desarrollo de los sistemas de bases de datos. En este libro se verán los conceptos y algoritmos que han sido incluidos en los sistemas de bases de datos para resolver los problemas mencionados anteriormente. En la mayor parte de este libro se usa una empresa bancaria como el ejemplo de una aplicación corriente de procesamiento de datos típica encontrada en una empresa.

1.3. VISIÓN DE LOS DATOS

Un sistema de bases de datos es una colección de archivos interrelacionados y un conjunto de programas que permitan a los usuarios acceder y modificar estos archivos. Uno de los propósitos principales de un sistema

de bases de datos es proporcionar a los usuarios una visión *abstracta* de los datos. Es decir, el sistema esconde ciertos detalles de cómo se almacenan y mantienen los datos.

1.3.1. Abstracción de datos

Para que el sistema sea útil debe recuperar los datos eficientemente. Esta preocupación ha conducido al diseño de estructuras de datos complejas para la representación de los datos en la base de datos. Como muchos usuarios de sistemas de bases de datos no están familiarizados con computadores, los desarrolladores esconden la complejidad a los usuarios a través de varios niveles de abstracción para simplificar la interacción de los usuarios con el sistema:

- **Nivel físico:** El nivel más bajo de abstracción describe *cómo* se almacenan realmente los datos. En el nivel físico se describen en detalle las estructuras de datos complejas de bajo nivel.
- **Nivel lógico:** El siguiente nivel más alto de abstracción describe *qué* datos se almacenan en la base de datos y *qué* relaciones existen entre esos datos. La base de datos completa se describe así en términos de un número pequeño de estructuras relativamente simples. Aunque la implementación de estructuras simples en el nivel lógico puede involucrar estructuras complejas del nivel físico, los usuarios del nivel lógico no necesitan preocuparse de esta complejidad. Los administradores de bases de datos, que deben decidir la información que se mantiene en la base de datos, usan el nivel lógico de abstracción.
- **Nivel de vistas:** El nivel más alto de abstracción describe sólo parte de la base de datos completa. A pesar del uso de estructuras más simples en el nivel lógico, queda algo de complejidad, debido a la variedad de información almacenada en una gran base de datos. Muchos usuarios del sistema de base de datos no necesitan toda esta información. En su lugar, tales usuarios necesitan acceder sólo a una parte de la base de datos. Para que su interacción con el sistema se simplifique, se define la abstracción del nivel de vistas. El sistema puede proporcionar muchas vistas para la misma base de datos.

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

Una analogía con el concepto de tipos de datos en lenguajes de programación puede clarificar la distinción entre los niveles de abstracción. La mayoría de lenguajes de programación de alto nivel soportan la estructura de tipo registro. Por ejemplo, en un lenguaje tipo Pascal, se pueden declarar registros como sigue:

```
type cliente = record
    nombre-cliente : string;
    id-cliente : string;
    calle-cliente : string;
    ciudad-cliente : string;
end;
```

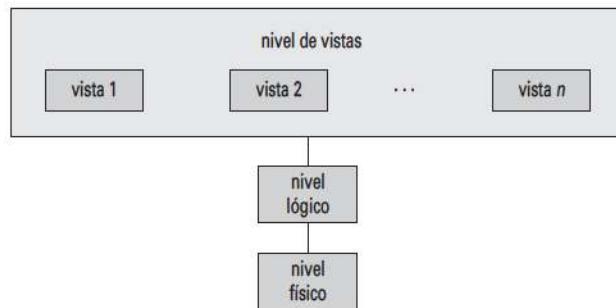


FIGURA 1.1. Los tres niveles de abstracción de datos.

Este código define un nuevo registro llamado *cliente* con cuatro campos. Cada campo tiene un nombre y un tipo asociado a él. Una empresa bancaria puede tener varios tipos de registros, incluyendo

- *cuenta*, con campos *número-cuenta* y *saldo*
- *empleado*, con campos *nombre-empleado* y *sUELDO*

En el nivel físico, un registro *cliente*, *cuenta* o *empleado* se puede describir como un bloque de posiciones almacenadas consecutivamente (por ejemplo, palabras o bytes). El compilador del lenguaje esconde este nivel de detalle a los programadores. Análogamente, el sistema de base de datos esconde muchos de los detalles de almacenamiento de nivel inferior a los programadores de bases de datos. Los administradores de bases de datos pueden ser conscientes de ciertos detalles de la organización física de los datos.

En el nivel lógico cada registro de este tipo se describe mediante una definición de tipo, como se ha ilustrado en el fragmento de código previo, y se define la relación entre estos tipos de registros. Los programadores, cuando usan un lenguaje de programación, trabajan en este nivel de abstracción. De forma similar, los administradores de bases de datos trabajan habitualmente en este nivel de abstracción.

Finalmente, en el nivel de vistas, los usuarios de computadores ven un conjunto de programas de aplicación que esconden los detalles de los tipos de datos. Análogamente, en el nivel de vistas se definen varias vistas de una base de datos y los usuarios de la misma ven única y exclusivamente esas vistas. Además de esconder detalles del nivel lógico de la base de datos, las vistas también proporcionan un mecanismo de seguridad para evitar que los usuarios accedan a ciertas partes de la base de datos. Por ejemplo, los cajeros de un banco ven únicamente la parte de la base de datos que tiene información de cuentas de clientes; no pueden acceder a la información referente a los sueldos de los empleados.

1.3.2. Ejemplares y esquemas

Las bases de datos van cambiando a lo largo del tiempo conforme la información se inserta y borra. La colección de información almacenada en la base de datos en

un momento particular se denomina un *ejemplar* de la base de datos. El diseño completo de la base de datos se llama el *esquema* de la base de datos. Los esquemas son raramente modificados, si es que lo son alguna vez.

El concepto de esquemas y ejemplares de bases de datos se puede entender por analogía con un programa escrito en un lenguaje de programación. Un *esquema* de base de datos corresponde a las declaraciones de variables (junto con definiciones de tipos asociadas) en un programa. Cada variable tiene un valor particular en un instante de tiempo. Los valores de las variables en un programa en un instante de tiempo corresponde a un *ejemplar* de un esquema de bases de datos.

Los sistemas de bases de datos tienen varios esquemas divididos de acuerdo a los niveles de abstracción que se han discutido. El **esquema físico** describe el diseño físico en el nivel físico, mientras que el **esquema lógico** des-

cribe el diseño de la base de datos en el nivel lógico. Una base de datos puede tener también varios esquemas en el nivel de vistas, a menudo denominados **subesquemas**, que describen diferentes vistas de la base de datos.

De éstos, el esquema lógico es con mucho el más importante, en términos de su efecto en los programas de aplicación, ya que los programadores construyen las aplicaciones usando el esquema lógico. El esquema físico está oculto bajo el esquema lógico, y puede ser fácilmente cambiado usualmente sin afectar a los programas de aplicación. Los programas de aplicación se dice que muestran independencia física de datos si no dependen del esquema físico y, por tanto, no deben ser modificados si cambia el esquema físico.

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

1.4. MODELOS DE LOS DATOS

Bajo la estructura de la base de datos se encuentra el **modelo de datos**: una colección de herramientas conceptuales para describir los datos, las relaciones, la semántica y las restricciones de consistencia. Para ilustrar el concepto de un modelo de datos, describimos dos modelos de datos en este apartado: el modelo entidad-relación y el modelo relacional. Los diferentes modelos de datos que se han propuesto se clasifican en tres grupos diferentes: modelos lógicos basados en objetos, modelos lógicos basados en registros y modelos físicos.

1.4.1. Modelo entidad-relación

El modelo de datos entidad-relación (E-R) está basado en una percepción del mundo real que consta de una colección de objetos básicos, llamados *entidades*, y de *relaciones* entre estos objetos. Una entidad es una «cosa» u «objeto» en el mundo real que es distingible de otros objetos. Por ejemplo, cada persona es una entidad, y las cuentas bancarias pueden ser consideradas entidades.

Las entidades se describen en una base de datos mediante un conjunto de **atributos**. Por ejemplo, los atributos *número-cuenta* y *saldo* describen una cuenta particular de un banco y pueden ser atributos del conjunto de entidades *cuenta*. Análogamente, los atributos *nombre-cliente*, *calle-cliente* y *ciudad-cliente* pueden describir una entidad *cliente*.

Un atributo extra, *id-cliente*, se usa para identificar únicamente a los clientes (dado que puede ser posible que haya dos clientes con el mismo nombre, direc-

ción y ciudad. Se debe asignar un identificador único de cliente a cada cliente. En los Estados Unidos, muchas empresas utilizan el número de la seguridad social de una persona (un número único que el Gobierno de los Estados Unidos asigna a cada persona en los Estados Unidos) como identificador de cliente*.

Una **relación** es una asociación entre varias entidades. Por ejemplo, una relación *impositor* asocia un cliente con cada cuenta que tiene. El conjunto de todas las entidades del mismo tipo, y el conjunto de todas las relaciones del mismo tipo, se denominan respectivamente **conjunto de entidades** y **conjunto de relaciones**.

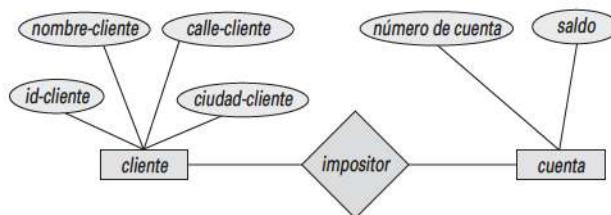
La estructura lógica general de una base de datos se puede expresar gráficamente mediante un *diagrama E-R*, que consta de los siguientes componentes:

- **Rectángulos**, que representan conjuntos de entidades.
- **Elipses**, que representan atributos.
- **Rombos**, que representan relaciones entre conjuntos de entidades.
- **Líneas**, que unen los atributos con los conjuntos de entidades y los conjuntos de entidades con las relaciones.

Cada componente se etiqueta con la entidad o relación que representa.

Como ilustración, considérese parte de una base de datos de un sistema bancario consistente en clientes y cuentas que tienen esos clientes. En la Figura 1.2 se

* N. del T. En España, muchas empresas usan el D.N.I. como identificador único, pero a veces encuentran problemas con los números de D.N.I. que por desgracia aparecen repetidos. Para resolverlo, o bien se usa otro identificador propio de la empresa o se añade un código al número de D.N.I.

**FIGURA 1.2.** Ejemplo de diagrama E-R.

muestra el diagrama E-R correspondiente. El diagrama E-R indica que hay dos conjuntos de entidades *cliente* y *cuenta*, con los atributos descritos anteriormente. El diagrama también muestra la relación *impositor* entre cliente y cuenta.

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

El modelo entidad-relación se utiliza habitualmente en el proceso de diseño de bases de datos, y se estudiará en profundidad en el Capítulo 2.

1.4.2. Modelo relacional

En el modelo relacional se utiliza un grupo de tablas para representar los datos y las relaciones entre ellos. Cada tabla está compuesta por varias columnas, y cada columna tiene un nombre único. En la Figura 1.3 se presenta un ejemplo de base de datos relacional consistente en tres tablas: la primera muestra los clientes de un banco, la segunda, las cuentas, y la tercera, las cuentas que pertenecen a cada cliente.

<i>id-cliente</i>	<i>nombre-cliente</i>	<i>calle-cliente</i>	<i>ciudad-cliente</i>
19.283.746	González	Arenal	La Granja
01.928.374	Gómez	Carretas	Cerceda
67.789.901	López	Mayor	Peguerinos
18.273.609	Abrial	Preciados	Valsaín
32.112.312	Santos	Mayor	Peguerinos
33.666.999	Rupérez	Ramblas	León
01.928.374	Gómez	Carretas	Cerceda

(a) La tabla *cliente*

<i>número-cuenta</i>	<i>saldo</i>	<i>id-cliente</i>	<i>número-cuenta</i>
C-101	500	19.283.746	C-101
C-215	700	19.283.746	C-201
C-102	400	01.928.374	C-215
C-305	350	67.789.901	C-102
C-201	900	18.273.609	C-305
C-217	750	32.112.312	C-217
C-222	700	33.666.999	C-222
		01.928.374	C-201

(b) La tabla *cuenta*(b) La tabla *impositor***FIGURA 1.3.** Ejemplo de base de datos relacional.

La primera tabla, la tabla *cliente*, muestra, por ejemplo, que el cliente cuyo identificador es 19.283.746 se llama González y vive en la calle Arenal sita en La Granja. La segunda tabla, *cuenta*, muestra que las cuentas C-101 tienen un saldo de 500 € y la C-201 un saldo de 900 € respectivamente.

La tercera tabla muestra las cuentas que pertenecen a cada cliente. Por ejemplo, la cuenta C-101 pertenece al cliente cuyo identificador es 19.283.746 (González), y los clientes 19.283.746 (González) y 01.928.374 (Gómez) comparten el número de cuenta A-201 (pueden compartir un negocio).

El modelo relacional es un ejemplo de un modelo basado en registros. Los modelos basados en registros se denominan así porque la base de datos se estructura en registros de formato fijo de varios tipos. Cada tabla contiene registros de un tipo particular. Cada tipo de registro define un número fijo de campos, o atributos. Las columnas de la tabla corresponden a los atributos del tipo de registro.

No es difícil ver cómo se pueden almacenar las tablas en archivos. Por ejemplo, un carácter especial (como una coma) se puede usar para delimitar los diferentes atributos de un registro, y otro carácter especial (como un carácter de nueva línea) se puede usar para delimitar registros. El modelo relacional oculta tales detalles de implementación de bajo nivel a los desarrolladores de bases de datos y usuarios.

El modelo de datos relacional es el modelo de datos más ampliamente usado, y una amplia mayoría de sistemas de bases de datos actuales se basan en el modelo relacional. Los Capítulos 3 a 7 tratan el modelo relacional en detalle.

El modelo relacional se encuentra a un nivel de abstracción inferior al modelo de datos E-R. Los diseños de bases de datos a menudo se realizan en el modelo E-R, y después se traducen al modelo relacional; el Capítulo 2 describe el proceso de traducción. Por ejemplo, es fácil ver que las tablas *cliente* y *cuenta* corresponden a los conjuntos de entidades del mismo nombre, mientras que la tabla *impositor* corresponde al conjunto de relaciones *impositor*.

Nótese también que es posible crear esquemas en el modelo relacional que tengan problemas tales como información duplicada innecesariamente. Por ejemplo, supongamos que se almacena *número-cuenta* como un atributo del registro *cliente*. Entonces, para representar el hecho de que las cuentas C-101 y C-201 pertenecen ambas al cliente González (con identificador de cliente 19.283.746) sería necesario almacenar dos filas en la tabla *cliente*. Los valores de *nombre-cliente*, *calle-cliente* y *ciudad-cliente* de González estarían innecesariamente duplicados en las dos filas. En el Capítulo 7 se estudiará cómo distinguir buenos diseños de esquema de malos diseños.

1.4.3. Otros modelos de datos

El **modelo de datos orientado a objetos** es otro modelo de datos que está recibiendo una atención creciente.

El modelo orientado a objetos se puede observar como una extensión del modelo E-R con las nociones de encapsulación, métodos (funciones) e identidad de objeto. El Capítulo 8 examina el modelo de datos orientado a objetos.

El modelo de datos relacional orientado a objetos combina las características del modelo de datos orientado a objetos y el modelo de datos relacional. El Capítulo 9 lo examina.

Los modelos de datos semiestructurados permiten la especificación de datos donde los elementos de datos individuales del mismo tipo pueden tener diferentes conjuntos de atributos. Esto es diferente de los modelos de datos mencionados anteriormente, en los que cada ele-

mento de datos de un tipo particular debe tener el mismo conjunto de atributos. El **lenguaje de marcas extensible (XML, eXtensible Markup Language)** se usa ampliamente para representar datos semiestructurados. El Capítulo 10 lo trata.

Históricamente, otros dos modelos de datos, el **modelo de datos de red** y el **modelo de datos jerárquico**, precedieron al modelo de datos relacional. Estos modelos estuvieron ligados fuertemente a la implementación subyacente y complicaban la tarea del modelado de datos. Como resultado se usan muy poco actualmente, excepto en el código de bases de datos antiguo que aún está en servicio en algunos lugares. Se describen en los apéndices A y B para los lectores interesados.

1.5. LENGUAJES DE BASES DE DATOS

Un sistema de bases de datos proporciona un **lenguaje de definición de datos** para especificar el esquema de la base de datos y un **lenguaje de manipulación de datos** para expresar las consultas a la base de datos y las modificaciones. En la práctica, los lenguajes de definición y manipulación de datos no son dos lenguajes separados; en su lugar simplemente forman partes de un único lenguaje de bases de datos, tal como SQL, ampliamente usado.

1.5.1. Lenguaje de definición de datos

Un esquema de base de datos se especifica mediante un conjunto de definiciones expresadas mediante un lenguaje especial llamado **lenguaje de definición de datos (LDD)**.

Por ejemplo, la siguiente instrucción en el lenguaje SQL define la tabla *cuenta*:

```
create table cuenta
  (número-cuenta char(10),
   saldo integer)
```

La ejecución de la instrucción LDD anterior crea la tabla *cuenta*. Además, actualiza un conjunto especial de tablas denominado **diccionario de datos o directorio de datos**.

Un diccionario de datos contiene **metadatos**, es decir, datos acerca de los datos. El esquema de una tabla es un ejemplo de metadatos. Un sistema de base de datos consulta el diccionario de datos antes de leer o modificar los datos reales.

Especificamos el almacenamiento y los métodos de acceso usados por el sistema de bases de datos por un conjunto de instrucciones en un tipo especial de LDD denominado lenguaje de **almacenamiento y definición de datos**. Estas instrucciones definen los detalles de implementación de los esquemas de base de datos, que se ocultan usualmente a los usuarios.

Los valores de datos almacenados en la base de datos deben satisfacer ciertas **restricciones de consistencia**. Por ejemplo, supóngase que el saldo de una cuenta no debe caer por debajo de 100 €. El LDD proporciona facilidades para especificar tales restricciones. Los sistemas de bases de datos comprueban estas restricciones cada vez que se actualiza la base de datos.

1.5.2. Lenguaje de manipulación de datos

La **manipulación de datos** es:

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

Un **lenguaje de manipulación de datos (LMD)** es un lenguaje que permite a los usuarios acceder o manipular los datos organizados mediante el modelo de datos apropiado. Hay dos tipos básicamente:

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

Los LMDs declarativos son más fáciles de aprender y usar que los LMDs procedimentales. Sin embargo, como el usuario no especifica cómo conseguir los datos, el sistema de bases de datos tiene que determinar un medio eficiente de acceder a los datos. El componente LMD del lenguaje SQL es no procedimental.

Una **consulta** es una instrucción de solicitud para recuperar información. La parte de un LMD que implica recuperación de información se llama **lenguaje de consultas**. Aunque técnicamente sea incorrecto, en la práctica se usan los términos *lenguaje de consultas* y *lenguaje de manipulación de datos* como sinónimos.

Esta consulta en el lenguaje SQL encuentra el nombre del cliente cuyo identificador de cliente es 19.283.746:

```
select cliente.nombre-cliente
from cliente
where cliente.id-cliente = '19 283 746'
```

La consulta especifica que las filas *de* (from) la tabla *cliente donde* (where) el id-cliente es 19 283 746 se debe recuperar, y que se debe mostrar el atributo *nombre-cliente* de estas filas. Si se ejecutase la consulta con la tabla de la Figura 1.3, se mostraría el nombre González.

Las consultas pueden involucrar información de más de una tabla. Por ejemplo, la siguiente consulta encuentra el saldo de todas las cuentas pertenecientes al cliente cuyo identificador de cliente es 19 283 746.

```
select cuenta.saldo
from impositor, cuenta
where impositor.id-cliente = '19-283-746' and
      impositor.número-cuenta = cuenta.número-
cuenta
```

Si la consulta anterior se ejecutase con las tablas de la Figura 1.3, el sistema encontraría que las dos cuentas denominadas C-101 y C-201 pertenecen al cliente 19 283 746 e imprimiría los saldos de las dos cuentas, es decir, 500 y 900 €.

Hay varios lenguajes de consulta de bases de datos en uso, ya sea comercialmente o experimentalmente. Se estudiará el lenguaje de consultas más ampliamente usado, SQL, en el Capítulo 4. También se estudiarán otros lenguajes de consultas en el Capítulo 5.

Los niveles de abstracción que se discutieron en el Apartado 1.3 se aplican no solo a la definición o estructuración de datos, sino también a la manipulación de datos.

En el nivel físico se deben definir algoritmos que permitan un acceso eficiente a los datos. En los niveles superiores de abstracción se enfatiza la facilidad de uso. El objetivo es proporcionar una interacción humana eficiente con el sistema. El componente procesador de consultas del sistema de bases de datos (que se estudia en los Capítulos 13 y 14) traduce las consultas LMD en secuencias de acciones en el nivel físico del sistema de bases de datos.

1.5.3. Acceso a la base de datos desde programas de aplicación

Los **programas de aplicación** son programas que se usan para interactuar con la base de datos. Los programas de aplicación se escriben usualmente en un lenguaje *anfitrión*, tal como Cobol, C, C++ o Java. En el sistema bancario algunos ejemplos son programas que emiten los cheques de las nóminas, las cuentas de débito, las cuentas de crédito o las transferencias de fondos entre cuentas.

Para acceder a la base de datos, las instrucciones LMD necesitan ser ejecutadas desde el lenguaje anfitrión. Hay dos maneras de hacerlo:

- Proporcionando una interfaz de programas de aplicación (conjunto de procedimientos) que se pueden usar para enviar instrucciones LMD y LDD a la base de datos, y recuperar los resultados.

El estándar de conectividad abierta de bases de datos (ODBC, Open Data Base Connectivity) definido por Microsoft para el uso con el lenguaje C es un estándar de interfaz de programas de aplicación usado comúnmente. El estándar conectividad de Java con bases de datos (JDBC, Java Data Base Connectivity) proporciona características correspondientes para el lenguaje Java.

- Extendiendo la sintaxis del lenguaje anfitrión para incorporar llamadas LMD dentro del programa del lenguaje anfitrión. Usualmente, un carácter especial precede a las llamadas LMD, y un preprocesador, denominado el **precompilador LMD**, convierte las instrucciones LMD en llamadas normales a procedimientos en el lenguaje anfitrión.

1.6. USUARIOS Y ADMINISTRADORES DE LA BASE DE DATOS

Un objetivo principal de un sistema de bases de datos es recuperar información y almacenar nueva información en la base de datos. Las personas que trabajan con una base de datos se pueden catalogar como usuarios de bases de datos o como administradores de bases de datos.

1.6.1. Usuarios de bases de datos e interfaces de usuario

Hay cuatro tipos diferentes de usuarios de un sistema de base de datos, diferenciados por la forma en que ellos

esperan interactuar con el sistema. Se han diseñado diferentes tipo de interfaces de usuario para diferentes tipos de usuarios.

- **Usuarios normales.** Son usuarios no sofisticados que interactúan con el sistema mediante la invocación de alguno de los programas de aplicación permanentes que se ha escrito previamente. Por ejemplo, un cajero bancario que necesita transferir 50 € de la cuenta A a la cuenta B invoca un programa llamado *transferir*. Este programa pide al

cajero el importe de dinero a transferir, la cuenta de la que el dinero va a ser transferido y la cuenta a la que el dinero va a ser transferido.

Como otro ejemplo, considérese un usuario que desee encontrar su saldo de cuenta en World Wide Web. Tal usuario podría acceder a un formulario en el que introduce su número de cuenta. Un programa de aplicación en el servidor Web recupera entonces el saldo de la cuenta, usando el número de cuenta proporcionado, y pasa la información al usuario.

La interfaz de usuario normal para los usuarios normales es una interfaz de formularios, donde el usuario puede llenar los campos apropiados del formulario. Los usuarios normales pueden también simplemente leer *informes* generados de la base de datos.

- **Programadores de aplicaciones.** Son profesionales informáticos que escriben programas de aplicación. Los programadores de aplicaciones pueden elegir entre muchas herramientas para desarrollar interfaces de usuario. Las herramientas de **desarrollo rápido de aplicaciones (DRA)** son herramientas que permiten al programador de aplicaciones construir formularios e informes sin escribir un programa. Hay también tipos especiales de lenguajes de programación que combinan estructuras de control imperativo (por ejemplo, para bucles for, bucles while e instrucciones if-then-else) con instrucciones del lenguaje de manipulación de datos. Estos lenguajes, llamados a veces *lenguajes de cuarta generación*, a menudo incluyen características especiales para facilitar la generación de formularios y la presentación de datos en pantalla. La mayoría de los sistemas de bases de datos comerciales incluyen un lenguaje de cuarta generación.
- **Los usuarios sofisticados** interactúan con el sistema sin programas escritos. En su lugar, ellos forman sus consultas en un lenguaje de consulta de bases de datos. Cada una de estas consultas se envía al *procesador de consultas*, cuya función es transformar instrucciones LMD a instrucciones que el gestor de almacenamiento entienda. Los analistas que envían las consultas para explorar los datos en la base de datos entran en esta categoría.

Las herramientas de **procesamiento analítico en línea (OLAP, Online Analytical Processing)** simplifican la labor de los analistas permitiéndoles ver resúmenes de datos de formas diferentes. Por ejemplo, un analista puede ver las ventas totales por región (por ejemplo, norte, sur, este y oeste), o por producto, o por una combinación de la región y del producto (es decir, las ventas totales de cada producto en cada región). Las herramientas también permiten al analista seleccionar regiones específicas, examinar los datos con más deta-

lle (por ejemplo, ventas por ciudad dentro de una región) o examinar los datos con menos detalle (por ejemplo, agrupando productos por categoría).

Otra clase de herramientas para los analistas son las herramientas de **recopilación de datos**, que les ayudan a encontrar ciertas clases de patrones de datos.

En el Capítulo 22 se estudiarán las herramientas de recopilación de datos.

- **Usuarios especializados.** Son usuarios sofisticados que escriben aplicaciones de bases de datos especializadas que no son adecuadas en el marco de procesamiento de datos tradicional. Entre estas aplicaciones están los sistemas de diseño asistido por computador, sistemas de bases de conocimientos y sistemas expertos, sistemas que almacenan los datos con tipos de datos complejos (por ejemplo, datos gráficos y datos de audio) y sistemas de modelado del entorno. Varias de estas aplicaciones se tratan en los Capítulos 8 y 9.

1.6.2. Administrador de la base de datos

Una de las principales razones de usar SGBDs es tener un control centralizado tanto de los datos como de los programas que acceden a esos datos. La persona que tiene este control central sobre el sistema se llama **administrador de la base de datos (ABD)**. Las funciones del ABD incluyen las siguientes:

- **Definición del esquema.** El ABD crea el esquema original de la base de datos escribiendo un conjunto de instrucciones de definición de datos en el LDD.
- **Definición de la estructura y del método de acceso.**
- **Modificación del esquema y de la organización física.** Los ABD realizan cambios en el esquema y en la organización física para reflejar las necesidades cambiantes de la organización, o para alterar la organización física para mejorar el rendimiento.
- **Concesión de autorización para el acceso a los datos.** La concesión de diferentes tipos de autorización permite al administrador de la base de datos determinar a qué partes de la base de datos puede acceder cada usuario. La información de autorización se mantiene en una estructura del sistema especial que el sistema de base de datos consulta cuando se intenta el acceso a los datos en el sistema.
- **Mantenimiento rutinario.** Algunos ejemplos de actividades rutinarias de mantenimiento del administrador de la base de datos son:
 - Copia de seguridad periódica de la base de datos, bien sobre cinta o sobre servidores remotos, para prevenir la pérdida de datos en caso de desastres como inundaciones.

- Asegurarse de que haya suficiente espacio libre en disco para las operaciones normales y aumentar el espacio en disco según sea necesario.
- Supervisión de los trabajos que se ejecuten en la base de datos y asegurarse de que el rendimiento no se degrada por tareas muy costosas iniciadas por algunos usuarios.

1.7. GESTIÓN DE TRANSACCIONES

Varias operaciones sobre la base de datos forman a menudo una única unidad lógica de trabajo. Un ejemplo que se vio en el Apartado 1.2 es la transferencia de fondos, en el que una cuenta (*A*) se carga y otra cuenta (*B*) se abona. Claramente es esencial que, o bien tanto el cargo como el abono tengan lugar, o bien no ocurra ninguno. Es decir, la transferencia de fondos debe ocurrir por completo o no ocurrir en absoluto. Este requisito de todo o nada se denomina **atomicidad**. Además, es esencial que la ejecución de la transferencia de fondos preserve la consistencia de la base de datos. Es decir, el valor de la suma $A + B$ se debe preservar. Este requisito de corrección se llama **consistencia**. Finalmente, tras la ejecución correcta de la transferencia de fondos, los nuevos valores de las cuentas *A* y *B* deben persistir, a pesar de la posibilidad de fallo del sistema. Este requisito de persistencia se llama **durabilidad**.

Una **transacción** es una colección de operaciones que se lleva a cabo como una única función lógica en una aplicación de bases de datos. Cada transacción es una unidad de atomicidad y consistencia. Así, se requiere que las transacciones no violen ninguna restricción de consistencia de la base de datos. Es decir, si la base de datos era consistente cuando la transacción comenzó, la base de datos debe ser consistente cuando la transacción termine con éxito. Sin embargo, durante la ejecución de una transacción, puede ser necesario permitir inconsistencias temporales, ya que o el cargo de *A* o el abono de *B* se debe realizar uno antes que otro. Esta inconsistencia temporal, aunque necesaria, puede conducir a dificultades si ocurre un fallo.

Es responsabilidad del programador definir adecuadamente las diferentes transacciones, de tal manera que cada una preserve la consistencia de la base de datos. Por ejemplo, la transacción para transferir fondos de la cuenta *A* a la cuenta *B* se podría definir como compuesta de dos programas separados: uno que carga la cuenta *A* y otro que abona la cuenta *B*. La ejecución de estos dos programas uno después del otro preservará realmente

la consistencia. Sin embargo, cada programa en sí mismo no transforma la base de datos de un estado consistente en otro nuevo estado consistente. Así, estos programas no son transacciones.

Asegurar las propiedades de atomicidad y durabilidad es responsabilidad del propio sistema de bases de datos, específicamente del **componente de gestión de transacciones**. En ausencia de fallos, toda transacción completada con éxito y átomica se archiva fácilmente. Sin embargo, debido a diversos tipos de fallos, una transacción puede no siempre completar su ejecución con éxito. Si se asegura la propiedad de atomicidad, una transacción que falle no debe tener efecto en el estado de la base de datos. Así, la base de datos se restaura al estado en que estaba antes de que la transacción en cuestión comenzara su ejecución. El sistema de bases de datos debe realizar la **recuperación de fallos**, es decir, detectar los fallos del sistema y restaurar la base de datos al estado que existía antes de que ocurriera el fallo.

Finalmente, cuando varias transacciones actualizan la base de datos concurrentemente, la consistencia de los datos puede no ser preservada, incluso aunque cada transacción individualmente sea correcta. Es responsabilidad del **gestor de control de concurrencia** controlar la interacción entre las transacciones concurrentes para asegurar la consistencia de la base de datos.

Los sistemas de bases de datos diseñados para uso sobre pequeños computadores personales pueden no tener todas las características vistas. Por ejemplo, muchos sistemas pequeños imponen la restricción de permitir el acceso a un único usuario a la base de datos en un instante de tiempo. Otros dejan las tareas de copias de seguridad y recuperación a los usuarios. Estas restricciones permiten un gestor de datos más pequeño, con menos requisitos de recursos físicos, especialmente de memoria principal. Aunque tales enfoques de bajo coste y prestaciones son suficientes para bases de datos personales pequeñas, son inadecuadas para satisfacer las necesidades de una empresa de media a gran escala.

1.8. ESTRUCTURA DE UN SISTEMA DE BASES DE DATOS

Un sistema de bases de datos se divide en módulos que se encargan de cada una de las responsabilidades del sistema completo. Los componentes funcionales de un sistema de bases de datos se pueden dividir a grandes

rasgos en los componentes gestor de almacenamiento y procesador de consultas.

El gestor de consultas es importante porque las bases de datos requieren normalmente una gran cantidad de

espacio de almacenamiento. Las bases de datos corporativas tienen un tamaño de entre cientos de gigabytes y, para las mayores bases de datos, terabytes de datos. Un gigabyte son 1.000 megabytes (1.000 millones de bytes), y un terabyte es 1 millón de megabytes (1 billón de bytes). Debido a que la memoria principal de los computadores no puede almacenar esta gran cantidad de información, esta se almacena en discos. Los datos se trasladan entre el disco de almacenamiento y la memoria principal cuando es necesario. Como la transferencia de datos a y desde el disco es lenta comparada con la velocidad de la unidad central de procesamiento, es fundamental que el sistema de base de datos estructure los datos para minimizar la necesidad de movimiento de datos entre el disco y la memoria principal.

El procesador de consultas es importante porque ayuda al sistema de bases de datos a simplificar y facilitar el acceso a los datos. Las vistas de alto nivel ayudan a conseguir este objetivo. Con ellas, los usuarios del sistema no deberían ser molestados innecesariamente con los detalles físicos de implementación del sistema. Sin embargo, el rápido procesamiento de las actualizaciones y de las consultas es importante. Es trabajo del sistema de bases de datos traducir las actualizaciones y las consultas escritas en un lenguaje no procedimental, en el nivel lógico, en una secuencia de operaciones en el nivel físico.

1.8.1. Gestor de almacenamiento

Un *gestor de almacenamiento* es un módulo de programa que proporciona la interfaz entre los datos de bajo nivel en la base de datos y los programas de aplicación y consultas emitidas al sistema. El gestor de almacenamiento es responsable de la interacción con el gestor de archivos. Los datos en bruto se almacenan en disco usando un sistema de archivos, que está disponible habitualmente en un sistema operativo convencional. El gestor de almacenamiento traduce las diferentes instrucciones LMD a órdenes de un sistema de archivos de bajo nivel. Así, el gestor de almacenamiento es responsable del almacenamiento, recuperación y actualización de los datos en la base de datos.

Los componentes del gestor de almacenamiento incluyen:

- **Gestor de autorización e integridad**, que comprueba que se satisfagan las restricciones de integridad y la autorización de los usuarios para acceder a los datos.
- **Gestor de transacciones**, que asegura que la base de datos quede en un estado consistente (correc-

to) a pesar de los fallos del sistema, y que las ejecuciones de transacciones concurrentes ocurran si conflictos.

- **Gestor de archivos**, que gestiona la reserva de espacio de almacenamiento de disco y las estructuras de datos usadas para representar la información almacenada en disco.
- **Gestor de memoria intermedia**, que es responsable de traer los datos del disco de almacenamiento a memoria principal y decidir qué datos tratar en memoria caché. El gestor de memoria intermedia es una parte crítica del sistema de bases de datos, ya que permite que la base de datos maneje tamaños de datos que son mucho mayores que el tamaño de la memoria principal.

El gestor de almacenamiento implementa varias estructuras de datos como parte de la implementación física del sistema:

- **Archivos de datos**, que almacenan la base de datos en sí.
- **Diccionario de datos**, que almacena metadatos acerca de la estructura de la base de datos, en particular, el esquema de la base de datos.
- **Índices**, que proporcionan acceso rápido a elementos de datos que tienen valores particulares.

1.8.2. Procesador de consultas

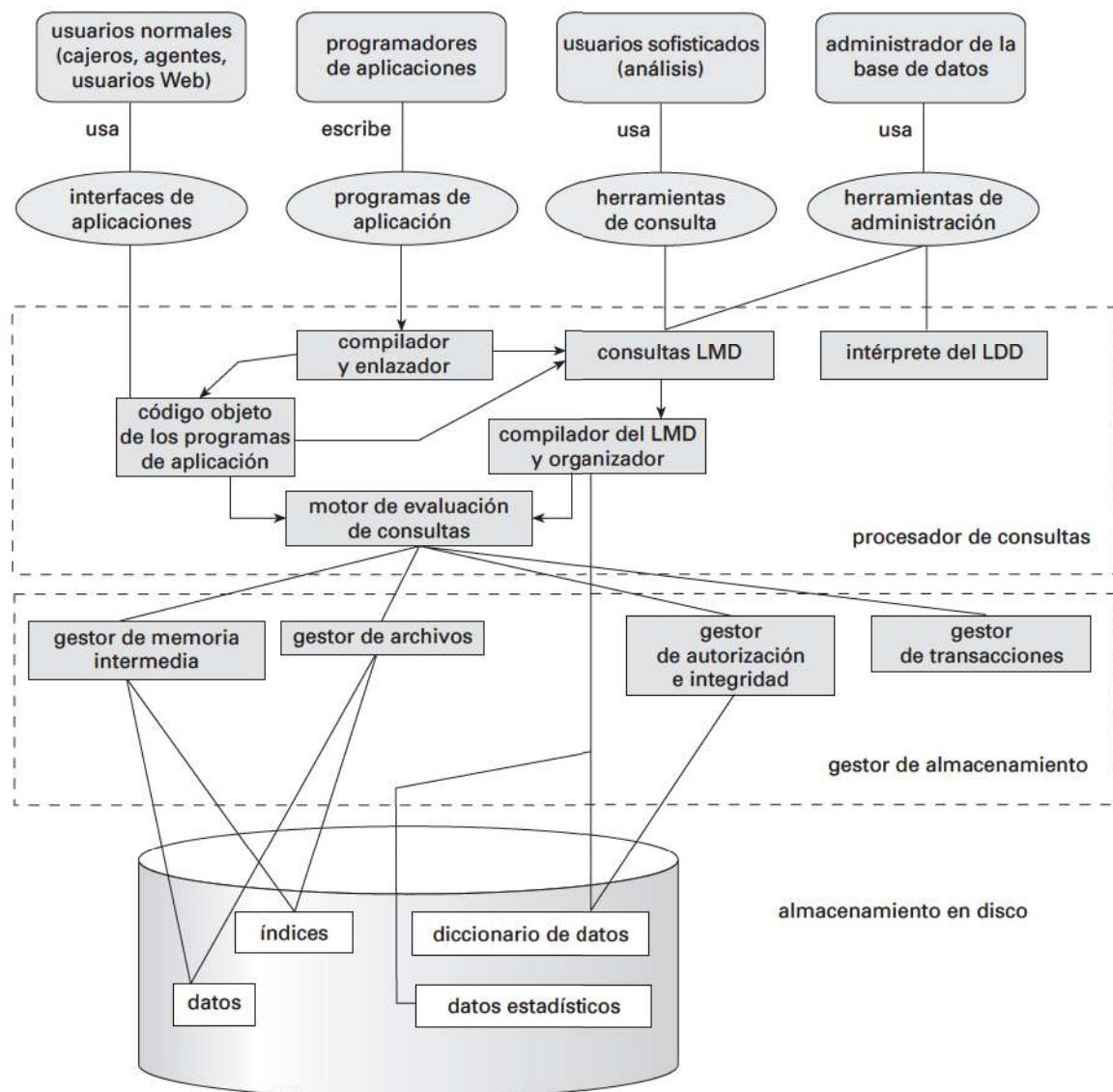
Los componentes del procesador de consultas incluyen:

- **Intérprete del LDD**, que interpreta las instrucciones del LDD y registra las definiciones en el diccionario de datos.
- **Compilador del LMD**, que traduce las instrucciones del LMD en un lenguaje de consultas a un plan de evaluación que consiste en instrucciones de bajo nivel que entiende el motor de evaluación de consultas.

Una consulta se puede traducir habitualmente en varios planes de ejecución alternativos que proporcionan el mismo resultado. El compilador del LMD también realiza **optimización de consultas**, es decir, elige el plan de evaluación de menor coste de entre todas las alternativas.

- **Motor de evaluación de consultas**, que ejecuta las instrucciones de bajo nivel generadas por el compilador del LMD.

En la Figura 1.4 se muestran estos componentes y sus conexiones.

**FIGURA 1.4.** Estructura del sistema.

1.9. ARQUITECTURAS DE APLICACIONES

La mayoría de usuarios de un sistema de bases de datos no están situados actualmente junto al sistema de bases de datos, sino que se conectan a él a través de una red. Se puede diferenciar entonces entre las máquinas **cliente**, en donde trabajan los usuarios remotos de la base de datos, y las máquinas **servidor**, en las que se ejecuta el sistema de bases de datos.

Las aplicaciones de bases de datos se dividen usualmente en dos o tres partes, como se ilustra en la Figura 1.5. En una **arquitectura de dos capas**, la aplicación se divide en un componente que reside en la máquina cliente, que llama a la funcionalidad del sistema de bases de datos en la máquina servidor mediante instrucciones del lenguaje de consultas. Los estándares de interfaces de programas de aplicación como

ODBC y JDBC se usan para la interacción entre el cliente y el servidor.

En cambio, en una **arquitectura de tres capas**, la máquina cliente actúa simplemente como frontal y no contiene ninguna llamada directa a la base de datos. En su lugar, el cliente se comunica con un **servidor de aplicaciones**, usualmente mediante una interfaz de formularios. El servidor de aplicaciones, a su vez, se comunica con el sistema de bases de datos para acceder a los datos. La **lógica de negocio** de la aplicación, que establece las acciones a realizar bajo determinadas condiciones, se incorpora en el servidor de aplicaciones, en lugar de ser distribuida a múltiples clientes. Las aplicaciones de tres capas son más apropiadas para grandes aplicaciones, y para las aplicaciones que se ejecutan en World Wide Web.

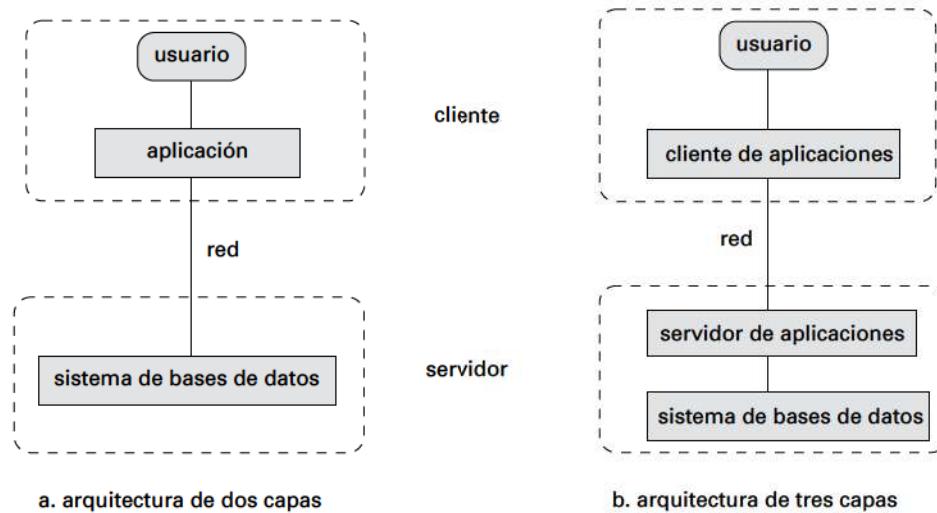


FIGURA 1.5. Arquitecturas de dos y tres capas.

1.10. HISTORIA DE LOS SISTEMAS DE BASES DE DATOS

El procesamiento de datos impulsa el crecimiento de los computadores, como ocurriera en los primeros días de los computadores comerciales. De hecho, la automatización de las tareas de procesamiento de datos precede a los computadores. Las tarjetas perforadas, inventadas por Hollerith, se usaron en los principios del siglo XX para registrar los datos del censo de los EE.UU., y se usaron sistemas mecánicos para procesar las tarjetas y para tabular los resultados. Las tarjetas perforadas posteriormente se usaron ampliamente como medio para introducir datos en los computadores.

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

- **Década de 1950 y principios de la década de 1960.** Se desarrollaron las cintas magnéticas para el almacenamiento de datos. Las tareas de procesamiento de datos tales como las nóminas fueron automatizadas, con los datos almacenados en cintas. El procesamiento de datos consistía en leer datos de una o más cintas y escribir datos en una nueva cinta. Los datos también se podían introducir desde paquetes de tarjetas perforadas e impresos en impresoras. Por ejemplo, los aumentos de sueldo se procesaban introduciendo los aumentos en las tarjetas perforadas y leyendo el paquete de cintas perforadas en sincronización con una cinta que contenía los detalles maestros de los salarios. Los registros debían estar igualmente ordenados. Los aumentos de sueldo tenían que añadirse a los sueldos leídos de la cinta maestra, y escribirse en una nueva cinta; esta nueva cinta se convertía en la nueva cinta maestra.

Las cintas (y los paquetes de tarjetas perforadas) sólo se podían leer secuencialmente, y los tamaños de datos eran mucho mayores que la

memoria principal; así, los programas de procesamiento de datos tenían que procesar los datos según un determinado orden, leyendo y mezclando datos de cintas y paquetes de tarjetas perforadas.

- **Finales de la década de 1960 y la década de 1970.** El amplio uso de los discos fijos a finales de la década de 1960 cambió en gran medida el escenario del procesamiento de datos, ya que los discos fijos permitieron el acceso directo a los datos. La ubicación de los datos en disco no era importante, ya que a cualquier posición del disco se podía acceder en sólo decenas de milisegundo. Los datos se liberaron de la tiranía de la secuencialidad. Con los discos pudieron desarrollarse las bases de datos de red y jerárquicas, que permitieron que las estructuras de datos tales como listas y árboles pudieran almacenarse en disco. Los programadores pudieron construir y manipular estas estructuras de datos.

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

- **Década de 1980.** Aunque académicamente interesante, el modelo relacional no se usó inicialmente en la práctica debido a sus inconvenientes por el rendimiento; las bases de datos relacionales no pudieron competir con el rendimiento de las bases

de datos de red y jerárquicas existentes. Esta situación cambió con System R, un proyecto innovador en IBM Research que desarrolló técnicas para la construcción de un sistema de bases de datos relacionales eficiente. En Astrahan et al. [1976] y Chamberlin et al. [1981] se pueden encontrar excepcionales visiones generales de System R. El prototipo de System R completamente funcional condujo al primer producto de bases de datos relacionales de IBM: SQL/DS. Los primeros sistemas de bases de datos relacionales, como DB2 de IBM, Oracle, Ingres y Rdb de DEC, jugaron un importante papel en el desarrollo de técnicas para el procesamiento eficiente de consultas declarativas. En los principios de la década de 1980 las bases de datos relacionales llegaron a competir con los sistemas de bases de datos jerárquicas y de red incluso en el área de rendimiento. Las bases de datos relacionales fueron tan sencillas de usar que finalmente reemplazaron a las bases de datos jerárquicas y de red; los programadores que usaban estas bases de datos estaban forzados a tratar muchos detalles de implementación de bajo nivel y tenían que codificar sus consultas de forma procedimental. Aún más importante, debían tener presente el rendimiento durante el diseño de sus programas, lo que implicaba un gran esfuerzo. En cambio, en una base de datos relacional, casi todas estas tareas de bajo nivel se realizan automáticamente por la base de datos, liberando al programador en el nivel lógico. Desde su escalada en el dominio en la década de 1980, el modelo relacional ha conseguido el reinado supremo entre todos los modelos de datos.

La década de 1980 también fue testigo de una gran investigación en las bases de datos paralelas y distribuidas, así como del trabajo inicial en las bases de datos orientadas a objetos.

- **Principios de la década de 1990.** El lenguaje SQL se diseñó fundamentalmente para las aplicaciones de ayuda a la toma de decisiones, que son intensivas en consultas, mientras que el objetivo principal de las bases de datos en la década de 1980 fue las aplicaciones de procesamiento de transacciones, que son intensivas en actualizaciones. La ayuda a la toma de decisiones y las consultas reemergieron como una importante área de aplicación para las bases de datos. Las herramientas para analizar grandes cantidades de datos experimentaron un gran crecimiento de uso.

Muchos vendedores de bases de datos introdujeron productos de bases de datos paralelas en este periodo, así como también comenzaron ofrecer bases de datos relacionales orientadas a objeto.

- **Finales de la década de 1990.** El principal acontecimiento fue el crecimiento explosivo de World Wide Web. Las bases de datos se implantaron mucho más extensivamente que nunca antes. Los sistemas de bases de datos tienen ahora soporte para tasas de transacciones muy altas, así como muy alta fiabilidad y disponibilidad 24x7 (disponibilidad 24 horas al día y 7 días a la semana, que significa que no hay tiempos de inactividad debidos a actividades de mantenimiento planificadas). Los sistemas de bases de datos también tuvieron interfaces Web a los datos.

1.11. RESUMEN

- Un **sistema gestor de bases de datos** (SGBD) consiste en una colección de datos interrelacionados y una colección de programas para acceder a esos datos. Los datos describen una empresa particular.
- El objetivo principal de un SGBD es proporcionar un entorno que sea tanto conveniente como eficiente para las personas que lo usan para la recuperación y almacenamiento de la información.
- Los sistemas de bases de datos se diseñan para almacenar grandes cantidades de información. La gestión de los datos implica tanto la definición de estructuras para el almacenamiento de la información como la provisión de mecanismos para la manipulación de la información. Además, los sistemas de bases de datos deben proporcionar la seguridad de la información almacenada, en caso de caídas del sistema o intentos de accesos sin autorización. Si los datos están compartidos por varios usuarios, el sistema debe evitar posibles resultados anómalos.
- Un propósito principal de un sistema de bases de datos es proporcionar a los usuarios una visión abstracta de los datos. Es decir, el sistema esconde ciertos detalles de cómo los datos se almacenan y mantienen.
- Por debajo de la estructura de la base de datos está el **modelo de datos**: una colección de herramientas conceptuales para describir los datos, las relaciones entre los datos, la semántica de los datos y las restricciones de los datos. El modelo de datos entidad-relación es un modelo de datos ampliamente usado, y proporciona una representación gráfica conveniente para ver los datos, las relaciones y las restricciones. El modelo de datos relacional se usa ampliamente para almacenar datos en las bases de datos. Otros modelos de datos son el modelo de datos orientado a objetos, el relacional orientado a objetos y modelos de datos semiestructurados.
- El diseño general de la base de datos se denomina el **esquema** de la base de datos. Un esquema de base de datos se especifica con un conjunto de definiciones

que se expresan usando un **lenguaje de definición de datos (LDD)**.

- Un **lenguaje de manipulación de datos (LMD)** es un lenguaje que permite a los usuarios acceder o manipular los datos. Los LMD no procedimentales, que requieren que un usuario especifique sólo los datos que necesita, se usan ampliamente hoy día.
- Los usuarios de bases de datos se pueden catalogar en varias clases, y cada clase de usuario usa habitualmente diferentes tipos de interfaces de la base de datos.
- Un sistema de bases de datos tiene varios subsistemas:
 - El subsistema gestor de transacciones es el responsable de asegurar que la base de datos permanezca en un estado consistente (correcto) a pesar de los fallos del sistema. El gestor de transacciones también asegura que las ejecuciones de transacciones concurrentes ocurran sin conflictos.
 - El subsistema procesador de consultas compila y ejecuta instrucciones LDD y LMD.
 - El subsistema gestor de almacenamiento es un módulo de programa que proporciona la interfaz entre los datos de bajo nivel almacenados en la base de datos y los programas de aplicación y las consultas enviadas al sistema.
- Las aplicaciones de bases de datos se dividen normalmente en un parte frontal que se ejecuta en las máquinas cliente y una parte que se ejecuta en el dorsal. En las arquitecturas de dos capas, el frontal se comunica directamente con una base de datos que se ejecuta en el dorsal. En las arquitecturas de tres capas, la parte dorsal se divide asimismo en un servidor de aplicaciones y en un servidor de bases de datos.

TÉRMINOS DE REPASO

- Abstracción de datos.
- Administrador de la base de datos (ADB).
- Aplicaciones de sistemas de bases de datos.
- Concurrencia.
- Diccionario de datos.
- Ejemplar de la base de datos.
- Esquema.
 - Esquema de la base de datos.
 - Esquema físico.
 - Esquema lógico.
- Inconsistencia de datos.
- Independencia física de los datos.
- Lenguajes de bases de datos.
 - Lenguaje de consultas.
 - Lenguaje de definición de datos.
- Lenguaje de manipulación de datos.
- Máquinas cliente y servidor.
- Metadatos.
- Modelos de datos.
 - Modelo de datos orientado a objetos.
 - Modelo de datos relacional.
 - Modelo de datos relacional orientado a objetos.
 - Modelo entidad-relación.
- Programa de aplicación.
- Restricciones de consistencia.
- Sistema de gestión de bases de datos (SGBD).
- Sistemas de archivos.
- Transacciones.
- Vistas de datos.

EJERCICIOS

- 1.1. ¿Cuáles son las cuatro diferencias principales entre un sistema de procesamiento de archivos y un SGBD?
- 1.2. En este capítulo se han descrito las diferentes ventajas principales de un sistema gestor de bases de datos. ¿Cuáles son los dos inconvenientes?
- 1.3. Explíquese la diferencia entre independencia de datos física y lógica.
- 1.4. Lístense las cinco responsabilidades del sistema gestor de la base de datos. Para cada responsabilidad explíquense los problemas que ocurrirían si no se realizara esa función.
- 1.5. ¿Cuáles son las cinco funciones principales del administrador de la base de datos?
- 1.6. Lístense siete lenguajes de programación que sean procedimentales y dos que sean no procedimentales. ¿Qué grupo es más fácil de aprender a usar? Explíquese la respuesta.
- 1.7. Lístense los seis pasos principales que se deberían dar en la realización de una base de datos para una empresa particular.
- 1.8. Considérese un *array* de enteros bidimensional de tamaño $n \times m$ que se va a usar en su lenguaje de programación preferido. Usando el *array* como ejemplo, ilústrese la diferencia (a) entre los tres niveles de abstracción y (b) entre esquema y ejemplares.

NOTAS BIBLIOGRÁFICAS

A continuación se listan libros de propósito general, colecciones de artículos de investigación y sitios Web de bases de datos.

Libros de texto que tratan los sistemas de bases de datos incluyen Abiteboul et al. [1995], Date [1995], Elmasri y Navathe [2000], O’Neil y O’Neil [2000], Ramakrishnan y Gehrke [2000] y Ullman [1988]. El tratamiento del procesamiento de transacciones en libros de texto se puede encontrar en Bernstein y Newcomer [1997] y Gray y Reuter [1993].

Varios libros incluyen colecciones de artículos de investigación sobre la gestión de bases de datos. Entre estos están Bancilhon y Buneman [1990], Date [1986], Date [1990], Kim [1995], Zaniolo et al. [1997], y Stoenbraker y Hellerstein [1998].

Una revisión de los logros en la gestión de bases de datos y una valoración de los desafíos en la investigación futura aparece en Silberschatz et al. [1990], Silberschatz et al. [1996] y Bernstein et al. [1998]. La página inicial del grupo especial de interés de la ACM en gestión de datos (véase www.acm.org/sigmod) proporciona una gran cantidad de información sobre la investigación en bases de datos. Los sitios Web de los vendedores de bases de datos (véase el apartado Herramientas a continuación) proporciona detalles acerca de sus respectivos productos.

Codd [1970] es el artículo histórico que introdujo el modelo relacional. En Fry y Sibley [1976] y Sibley [1976] se ofrecen discusiones referentes a la evolución de los SGBDs y al desarrollo de la tecnología de bases de datos.

HERRAMIENTAS

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

Hay también una serie de sistemas de bases de datos gratuitos/públicos; algunos ampliamente usados incluyen MySQL (www.mysql.com) y PostgreSQL (www.postgresql.org).

Una lista más completa de enlaces a vendedores y otra información se encuentra disponible en la página inicial de este libro en www.research.bell-labs.com/topic/books/db-book.

MODELOS DE DATOS

Un **modelo de datos** es una colección de herramientas conceptuales para la descripción de datos, relaciones entre datos, semántica de los datos y restricciones de consistencia. En esta parte se estudiarán dos modelos de datos —el modelo entidad-relación y el modelo relacional.

El modelo entidad-relación (E-R) es un modelo de datos de alto nivel. Está basado en una percepción de un mundo real que consiste en una colección de objetos básicos, denominados *entidades*, y de *relaciones* entre estos objetos.

El modelo relaciona es un modelo de menor nivel. Usa una colección de tablas para representar tanto los datos como las relaciones entre los datos. Su simplicidad conceptual ha conducido a su adopción general; actualmente, una vasta mayoría de productos de bases de datos se basan en el modelo relacional. Los diseñadores formulaen generalmente el diseño del esquema de la base de datos modelando primero los datos en alto nivel, usando el modelo E-R, y después traduciéndolo al modelo relacional.

Se estudiarán otros modelos de datos más tarde en este libro. El modelo de datos orientado a objetos, por ejemplo, extiende la representación de entidades añadiendo nociones de encapsulación, métodos (funciones) e identidad de objeto. El modelo de datos relacional orientado a objetos combina características del modelo de datos orientado a objetos y del modelo de datos relacional. Los Capítulos 8 y 9 tratan respectivamente estos dos modelos de datos.

El modelo de datos **entidad-relación** (**E-R**) está basado en una percepción del mundo real consistente en objetos básicos llamados *entidades* y de *relaciones* entre estos objetos. Se desarrolló para facilitar el diseño de bases de datos permitiendo la especificación de un *esquema de la empresa* que representa la estructura lógica completa de una base de datos. El modelo de datos E-R es uno de los diferentes modelos de datos semánticos; el aspecto semántico del modelo yace en la representación del significado de los datos. El modelo E-R es extremadamente útil para hacer corresponder los significados e interacciones de las empresas del mundo real con un esquema conceptual. Debido a esta utilidad, muchas herramientas de diseño de bases de datos se basan en los conceptos del modelo E-R.

2.1. CONCEPTOS BÁSICOS

Hay tres nociones básicas que emplea el modelo de datos E-R: conjuntos de entidades, conjuntos de relaciones y atributos.

2.1.1. Conjuntos de entidades

Una **entidad** es una «cosa» u «objeto» en el mundo real que es distingible de todos los demás objetos. Por ejemplo, cada persona en un desarrollo es una entidad. Una entidad tiene un conjunto de propiedades, y los valores para algún conjunto de propiedades pueden identificar una entidad de forma unívoca. Por ejemplo, el D.N.I. 67.789.901 identifica únicamente una persona particular en la empresa. Análogamente, se puede pensar en los préstamos bancarios como entidades, y un número de préstamo P-15 en la sucursal de Castellana identifica únicamente una entidad de préstamo. Una entidad puede ser concreta, como una persona o un libro, o puede ser abstracta, como un préstamo, unas vacaciones o un concepto.

Un **conjunto de entidades** es un conjunto de entidades del mismo tipo que comparten las mismas propiedades, o atributos. El conjunto de todas las personas que son clientes en un banco dado, por ejemplo, se pueden definir como el conjunto de entidades *cliente*. Análogamente, el conjunto de entidades *préstamo* podría representar el conjunto de todos los préstamos concedidos por un banco particular. Las entidades individuales que constituyen un conjunto se llaman la *extensión* del conjunto de entidades. Así, todos los clientes de un banco son la extensión del conjunto de entidades *cliente*.

Los conjuntos de entidades no son necesariamente disjuntos. Por ejemplo, es posible definir el conjunto de entidades de todos los empleados de un banco (*empleado*) y el conjunto de entidades de todos los clientes del banco (*cliente*). Una entidad *persona* puede ser una entidad *empleado*, una entidad *cliente*, ambas cosas, o ninguna.

Una entidad se representa mediante un conjunto de **atributos**. Los atributos describen propiedades que posee cada miembro de un conjunto de entidades. La designación de un atributo para un conjunto de entidades expresa que la base de datos almacena información similar concerniente a cada entidad del conjunto de entidades; sin embargo, cada entidad puede tener su propio valor para cada atributo. Posibles atributos del conjunto de entidades *cliente* son *id-cliente*, *nombre-cliente*, *calle-cliente* y *ciudad-cliente*. En la vida real, habría más atributos, tales como el número de la calle, el número del portal, la provincia, el código postal, y la comunidad autónoma, pero no se incluyen en el ejemplo simple. Posibles atributos del conjunto de entidades *préstamo* son *número-préstamo* e *importe*.

Cada entidad tiene un **valor** para cada uno de sus atributos. Por ejemplo, una entidad *cliente* en concreto puede tener el valor 32.112.312 para *id-cliente*, el valor Santos para *nombre-cliente*, el valor Mayor para *calle-cliente* y el valor Peguerinos para *ciudad-cliente*.

El atributo *id-cliente* se usa para identificar únicamente a los clientes, dado que no hay más de un cliente con el mismo nombre, calle y ciudad. En los Estados Unidos, muchas empresas encuentran conveniente usar el número *seguridad-social* de una persona¹ como un

¹ En España se asigna a cada persona del país un número único, denominado número del documento nacional de identidad (D.N.I.) para identificarla únicamente. Se supone que cada persona tiene un único D.N.I., y no hay dos personas con el mismo D.N.I.

atributo cuyo valor identifica únicamente a la persona. En general la empresa tendría que crear y asignar un identificador a cada cliente.

Para cada atributo hay un conjunto de valores permitidos, llamados el **dominio**, o el **conjunto de valores**, de ese atributo. El dominio del atributo *nombre-cliente* podría ser el conjunto de todas las cadenas de texto de una cierta longitud. Análogamente, el dominio del atributo *número-préstamo* podría ser el conjunto de todas las cadenas de la forma «P-n», donde n es un entero positivo.

Una base de datos incluye así una colección de conjuntos de entidades, cada una de las cuales contiene un número de entidades del mismo tipo. En la Figura 2.1 se muestra parte de una base de datos de un banco que consta de dos conjuntos de entidades, *cliente* y *préstamo*.

Formalmente, un atributo de un conjunto de entidades es una función que asigna al conjunto de entidades un dominio. Como un conjunto de entidades puede tener diferentes atributos, cada entidad se puede describir como un conjunto de pares (atributo, valor), un par para cada atributo del conjunto de entidades. Por ejemplo, una entidad concreta *cliente* se puede describir mediante el conjunto $\{(id\text{-}cliente}, 67.789.901\}, (nombre\text{-}cliente, \text{López}), (calle\text{-}cliente, \text{Mayor}), (ciudad\text{-}cliente, \text{Peguerinos})\}$, queriendo decir que la entidad describe una persona llamada López que tiene D.N.I. número 67.789.901, y reside en la calle Mayor en Peguerinos. Se puede ver, en este punto, que existe una integración del esquema abstracto con el desarrollo real de la empresa que se está modelando. Los valores de los atributos que describen una entidad constituirán una porción significante de los datos almacenados en la base de datos.

Un atributo, como se usa en el modelo E-R, se puede caracterizar por los siguientes tipos de atributo.

- Atributos **sencillos** y **compuestos**. En los ejemplos considerados hasta ahora, los atributos han sido simples; es decir, no están divididos en subpartes. Los

atributos compuestos, en cambio, se pueden dividir en subpartes (es decir, en otros atributos). Por ejemplo, *nombre-cliente* podría estar estructurado como un atributo compuesto consistente en *nombre*, *primer-apellido* y *segundo-apellido*. Usar atributos compuestos en un esquema de diseño es una buena elección si el usuario desea referirse a un atributo completo en algunas ocasiones y, en otras, a algún componente del atributo. Se podrían haber sustituido los atributos del conjunto de entidades *cliente*, *calle-cliente* y *ciudad-cliente*, por el atributo compuesto *dirección-cliente*, con los atributos *calle*, *ciudad*, *provincia*, y *código-postal*². Los atributos compuestos ayudan a agrupar los atributos relacionados, haciendo los modelos más claros.

Nótese también que un atributo compuesto puede aparecer como una jerarquía. Volviendo al ejemplo del atributo compuesto *dirección-cliente*, su componente *calle* puede ser a su vez dividido en *número-calle*, *nombre-calle* y *piso*. Estos ejemplos de atributos compuestos para el conjunto de entidades *cliente* se representa en la Figura 2.2.

- Atributos **monovalorados** y **multivalorados**. Los atributos que se han especificado en los ejemplos tienen todos un valor sólo para una entidad concreta. Por ejemplo, el atributo *número-préstamo* para una entidad préstamo específico, referencia a un único número de préstamo. Tales atributos se llaman **monovalorados**. Puede haber ocasiones en las que un atributo tiene un conjunto de valores para una entidad específica. Considérese un conjunto de entidades *empleado* con el atributo *número-teléfono*. Cualquier empleado particular puede tener cero, uno o más números de teléfono. Este tipo de atributo se llama **multivalorado**. En ellos, se pueden colocar apropiadamente límites inferior y superior en el número de valores en el atributo **multivalorado**. Como otro ejemplo, un atributo *nombre-subordinado* del conjunto de entidades *empleado*

² Se asume el formato de *calle-cliente* y *dirección* usado en España, que incluye un código postal numérico llamado «código postal».

Santos	32.112.312	Mayor	Peguerinos
Gómez	01.928.374	Carretas	Cerceda
López	67.789.901	Mayor	Peguerinos
Sotoca	55.555.555	Real	Cádiz
Pérez	24.466.880	Carretas	Cerceda
Valdivieso	96.396.396	Goya	Vigo
Fernández	33.557.799	Jazmín	León

cliente

P-17	1.000
P-23	2.000
P-15	1.500
P-14	1.500
P-19	500
P-11	900
P-16	1.300

préstamo

FIGURA 2.1. Conjunto de entidades *cliente* y *préstamo*.

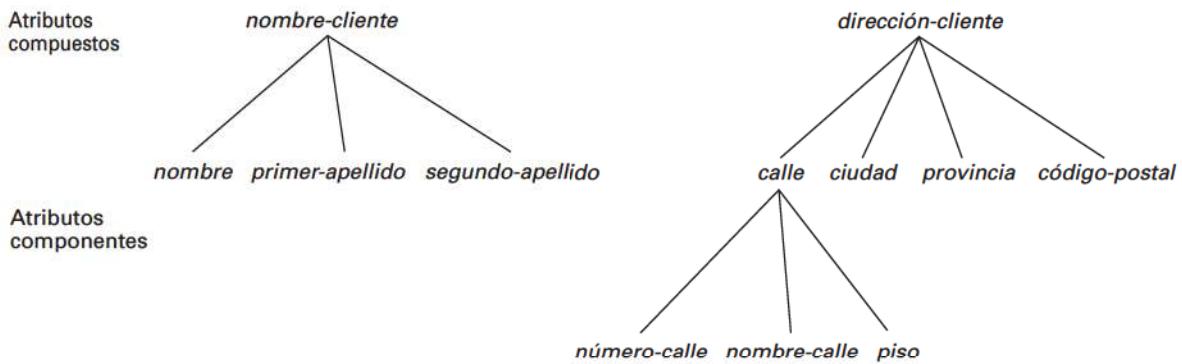


FIGURA 2.2. Atributos compuestos *nombre-cliente* y *dirección-cliente*.

sería multivalorado, ya que un empleado en concreto podría tener cero, uno o más subordinados.

Cuando sea apropiado se pueden establecer límites superior e inferior en el número de valores de un atributo multivalorado. Por ejemplo, un banco puede limitar el número de números de teléfono almacenados para un único cliente a dos. Colocando límites en este caso, se expresa que el atributo *número-teléfono* del conjunto de entidades *cliente* puede tener entre cero y dos valores.

- **Atributos derivados.** El valor para este tipo de atributo se puede derivar de los valores de otros atributos o entidades relacionados. Por ejemplo, sea el conjunto de entidades *cliente* que tiene un atributo *préstamos* que representa cuántos préstamos tiene un cliente en el banco. Ese atributo se puede derivar contando el número de entidades *préstamo* asociadas con ese *cliente*.

Como otro ejemplo, considérese que el conjunto de entidades *empleado* tiene un atributo *edad*, que indica la edad del cliente. Si el conjunto de entidades *cliente* tiene también un atributo *fecha-de-nacimiento*, se puede calcular *edad* a partir de *fecha-de-nacimiento* y de la fecha actual. Así, *edad* es un atributo derivado. En este caso, *fecha-de-nacimiento* y *antigüedad* pueden serlo, ya que representan el primer día en que el empleado comenzó a trabajar para el banco y el tiempo total que el empleado lleva trabajando para el banco, respectivamente. El valor de *antigüedad* se puede derivar del valor de *fecha-comienzo* y de la fecha actual. En este caso, *fecha-comienzo* se puede conocer como atributo *base* o atributo *almacenable*. El valor de un atributo derivado no se almacena, sino que se calcula cuando sea necesario.

Un atributo toma un valor **nulo** cuando una entidad no tiene un valor para un atributo. El valor *nulo* también puede indicar «no aplicable», es decir, que el valor no existe para la entidad. Por ejemplo, una persona puede no tener segundo nombre de pila. *Nulo* puede también designar que el valor de un atributo es desconocido. Un valor desconocido puede ser, bien *perdido* (el

valor existe pero no se tiene esa información) o *desconocido* (no se conoce si el valor existe realmente o no).

Por ejemplo, si el valor *nombre* para un *cliente* particular es *nulo*, se asume que el valor es perdido, ya que cada cliente debe tener un nombre. Un valor *nulo* para el atributo *piso* podría significar que la dirección no incluye un piso (no aplicable), que existe piso pero no se conoce cuál es (perdido), o que no se sabe si el piso forma parte o no de la dirección del cliente (desconocido).

Una base de datos para una empresa bancaria puede incluir diferentes conjuntos de entidades. Por ejemplo, además del mantenimiento de clientes y préstamos, el banco también proporciona cuentas, que se representan mediante el conjunto de entidades *cuenta* con atributos *número-cuenta* y *saldo*. También, si el banco tiene un número de sucursales diferentes, se puede mantener información acerca de todas las sucursales del banco. Cada conjunto de entidades *sucursal* se describe mediante los atributos *nombre-sucursal*, *ciudad-sucursal* y *activo*.

2.1.2. Conjuntos de relaciones

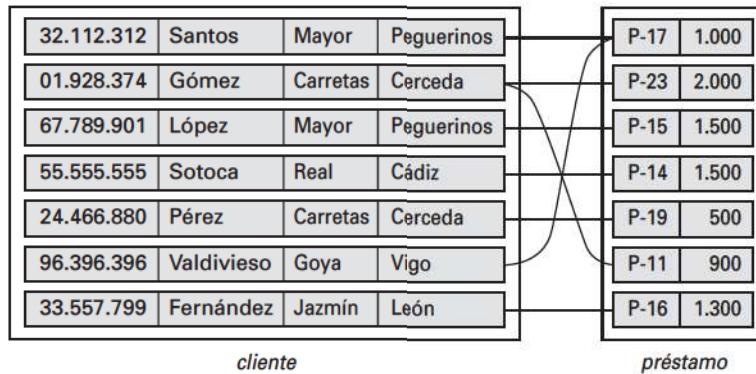
Una **relación** es una asociación entre diferentes entidades. Por ejemplo, se puede definir una relación que asocie al cliente López con el préstamo P-15. Esta relación especifica que López es un cliente con el préstamo número P-15.

Un **conjunto de relaciones** es un conjunto de relaciones del mismo tipo. Formalmente es una relación matemática con $n \geq 2$ de conjuntos de entidades (posiblemente no distintos). Si E_1, E_2, \dots, E_n son conjuntos de entidades, entonces un conjunto de relaciones R es un subconjunto de:

$$\{(e_1, e_2, \dots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \dots, e_n \in E_n\}$$

donde (e_1, e_2, \dots, e_n) es una relación.

Considérense las dos entidades *cliente* y *préstamo* de la Figura 2.1. Se define el conjunto de relaciones *prestatario* para denotar la asociación entre clientes y préstamos bancarios que los clientes tengan. Esta asociación se describe en la Figura 2.3.

**FIGURA 2.3.** Conjunto de relaciones *prestatario*.

Como otro ejemplo, considérense los dos conjuntos de entidades *préstamo* y *sucursal*. Se puede definir el conjunto de relaciones *sucursal-préstamo* para denotar la asociación entre un préstamo y la sucursal en que se mantiene ese préstamo.

La asociación entre conjuntos de entidades se conoce como *participación*; es decir, los conjuntos de entidades E_1, E_2, \dots, E_n **participan** en el conjunto de relaciones R. Un **ejemplar de relación** en un esquema E-R representa que existe una asociación entre las entidades denominadas en la empresa del mundo real que se modela. Como ilustración, el *cliente* individual López, que tiene D.N.I. 67.789.901, y la entidad *préstamo* P-15 participan en un ejemplar de relación de *prestatario*. Este ejemplar de relación representa que, en la empresa del mundo real, la persona llamada López cuyo número de D.N.I. es 67.789.901 ha tomado un préstamo que está numerado como P-15.

La función que desempeña una entidad en una relación se llama **papel** de la entidad.

Debido a que los conjuntos de entidades que participan en un conjunto de relaciones son generalmente distintos, los papeles están implícitos y no se especifican normalmente. Sin embargo, son útiles cuando el significado de una relación necesita aclaración. Tal es el caso cuando los conjuntos de entidades de una relación no son distintos; es decir, el mismo conjunto de entidades participa en una relación más de una vez con diferentes papeles. En este tipo de conjunto de relaciones, que se llama algunas veces conjunto de relaciones **recursivo**, es necesario hacer explícitos los papeles para especificar cómo participa una entidad en un ejemplar de relación. Por ejemplo, considérese una conjunto de entidades *empleado* que almacena información acerca de todos los empleados del banco. Se puede tener un conjunto de relaciones *trabaja-para* que se modela mediante pares ordenados de entidades *empleado*. El primer empleado de un par toma el papel de *trabajador*, mientras el segundo toma el papel de *jefe*. De esta manera, todas las relaciones *trabaja-para* son pares (*trabajador, jefe*); los pares (*jefe, trabajador*) están excluidos.

Una relación puede también tener **atributos descriptivos**. Considérese un conjunto de relaciones *impo-*

sitor con conjuntos de entidades *cliente* y *cuenta*. Se podría asociar el atributo *fecha-acceso* a esta relación para especificar la fecha más reciente en que un cliente accedió a una cuenta. La relación *impositor* entre las entidades correspondientes al cliente García y la cuenta C-217 se describen mediante $\{(fecha-acceso, 23\text{ mayo}\ 2002)\}$, lo que significa que la última vez que García accedió a la cuenta C-217 fue el 23 de mayo de 2002.

Como otro ejemplo de atributos descriptivos para relaciones, supóngase que se tienen los conjuntos de entidades *estudiante* y *asignatura* que participan en una relación *matriculado*. Se podría desechar almacenar un atributo descriptivo para *créditos* con la relación, para registrar si el estudiante se ha matriculado de la asignatura para obtener créditos o sólo como oyente.

Un ejemplar de relación en un conjunto de relaciones determinado debe ser identificado únicamente a partir de sus entidades participantes, sin usar los atributos descriptivos. Para comprender este punto supóngase que deseemos modelar todas las fechas en las que un cliente ha accedido a una cuenta. El atributo monovalorado *fecha-acceso* puede almacenar sólo una única fecha de acceso. No se pueden representar varias fechas de acceso por varios ejemplares de relación entre el mismo cliente y cuenta, ya que los ejemplares de relación no estarían identificados únicamente por las entidades participantes. La forma correcta de manejar este caso es crear un atributo multivalorado *fechas-acceso* que pueda almacenar todas las fechas de acceso.

Sin embargo, puede haber más de un conjunto de relaciones que involucren los mismos conjuntos de entidades. En nuestro ejemplo los conjuntos de entidades *cliente* y *préstamo* participan en el conjunto de relaciones *prestatario*. Además, supóngase que cada préstamo deba tener otro cliente que sirva como avalista para el préstamo. Entonces los conjuntos de entidades *cliente* y *préstamo* pueden participar en otro conjunto de relaciones: *avalista*.

Los conjuntos de relaciones *prestatario* y *sucursal-préstamo* proporcionan un ejemplo de un conjunto de relaciones **binario**, es decir, uno que implica dos conjuntos de entidades. La mayoría de los conjuntos de relaciones en un sistema de bases de datos son binarios.

Ocasionalmente, sin embargo, los conjuntos de relaciones implican más de dos conjuntos de entidades.

Por ejemplo, considérense los conjuntos de entidades *empleado*, *sucursal* y *trabajo*. Ejemplos de las entidades *trabajo* podrían ser director, cajero, auditor y otros. Las entidades *trabajo* pueden tener los atributos *puesto* y *nivel*. El conjunto de relaciones *trabaja-en* entre *empleado*, *sucursal* y *trabajo* es un ejemplo de una relación ternaria. Una relación ternaria entre Santos, Navacerrada y director indica que Santos actúa de

director de la sucursal Navacerrada. Santos también podría actuar como auditor de la sucursal Centro, que estaría representado por otra relación. Podría haber otra relación entre Gómez, Centro y cajero, indicando que Gómez actúa como cajero en la sucursal Centro.

El número de conjuntos de entidades que participan en un conjunto de relaciones es también el **grado** del conjunto de relaciones. Un conjunto de relaciones binario tiene grado 2; un conjunto de relaciones ternario tiene grado 3.

2.2. RESTRICCIONES

Un esquema de desarrollo E-R puede definir ciertas restricciones a las que los contenidos de la base de datos se deben adaptar. En este apartado se examina la correspondencia de cardinalidades y las restricciones de participación, que son dos de los tipos más importantes de restricciones.

2.2.1. Correspondencia de cardinalidades

La **correspondencia de cardinalidades**, o razón de cardinalidad, expresa el número de entidades a las que otra entidad puede estar asociada vía un conjunto de relaciones.

La correspondencia de cardinalidades es la más útil describiendo conjuntos de relaciones binarias, aunque ocasionalmente contribuye a la descripción de conjuntos de relaciones que implican más de dos conjuntos de entidades. Este apartado se centrará en conjuntos de relaciones binarias únicamente.

Para un conjunto de relaciones binarias *R* entre los conjuntos de entidades *A* y *B*, la correspondencia de cardinalidades debe ser una de las siguientes:

- **Uno a uno.** Una entidad en *A* se asocia con *a lo sumo* una entidad en *B*, y una entidad en *B* se asocia con *a lo sumo* una entidad en *A* (véase la Figura 2.4a).

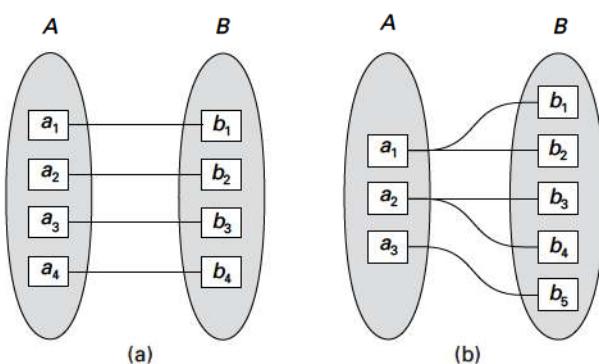


FIGURA 2.4. Correspondencia de cardinalidades. (a) Uno a uno. (b) Uno a varios.

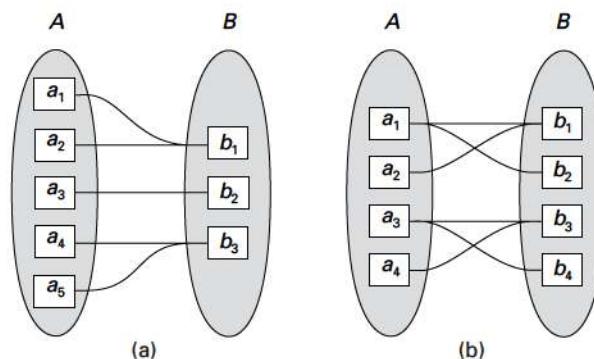


FIGURA 2.5. Correspondencia de cardinalidades. (a) Varios a uno. (b) Varios a varios.

- **Uno a varios.** Una entidad en *A* se asocia con cualquier número de entidades en *B* (ninguna o varias). Una entidad en *B*, sin embargo, se puede asociar con *a lo sumo* una entidad en *A* (véase la Figura 2.4b).
- **Varios a uno.** Una entidad en *A* se asocia con *a lo sumo* una entidad en *B*. Una entidad en *B*, sin embargo, se puede asociar con cualquier número de entidades (ninguna o varias) en *A* (véase la Figura 2.5a).
- **Varios a varios.** Una entidad en *A* se asocia con cualquier número de entidades (ninguna o varias) en *B*, y una entidad en *B* se asocia con cualquier número de entidades (ninguna o varias) en *A* (véase la Figura 2.5b).

La correspondencia de cardinalidades apropiada para un conjunto de relaciones particular depende obviamente de la situación del mundo real que el conjunto de relaciones modela.

Como ilustración considérese el conjunto de relaciones *prestatario*. Si en un banco particular un préstamo puede pertenecer únicamente a un cliente y un cliente puede tener varios préstamos, entonces el conjunto de relaciones de *cliente a préstamo* es uno a varios. Si un préstamo puede pertenecer a varios clientes (como préstamos que se toman en conjunto por varios socios de un negocio) el conjunto de relaciones es varios a varios. Este tipo de relación se describe en la Figura 2.3.

2.2.2. Restricciones de participación

La participación de un conjunto de entidades E en un conjunto de relaciones R se dice que es **total** si cada entidad en E participa al menos en una relación en R . Si sólo algunas entidades en E participan en relaciones en R , la participación del conjunto de entidades E en la relación R se llama **parcial**. Por ejemplo, se puede esperar que cada entidad *préstamo* esté relacionada con al

menos un cliente mediante la relación *prestatario*. Por lo tanto, la participación de *préstamo* en el conjunto de relaciones *prestatario* es total. En cambio, un individuo puede ser cliente de un banco tenga o no tenga un préstamo en el banco. Así, es posible que sólo algunas de las entidades *cliente* estén relacionadas con el conjunto de entidades *préstamo* mediante la relación *prestatario*, y la participación de *cliente* en el conjunto de relaciones *prestatario* es por lo tanto parcial.

2.3. CLAVES

Es necesario tener una forma de especificar cómo las entidades dentro de un conjunto de entidades dado y las relaciones dentro de un conjunto de relaciones dado son distinguibles. Conceptualmente las entidades y relaciones individuales son distintas; desde una perspectiva de bases de datos, sin embargo, la diferencia entre ellas se debe expresar en término de sus atributos.

Por lo tanto, los valores de los atributos de una entidad deben ser tales que permitan *identificar únicamente* a la entidad. En otras palabras, no se permite que ningún par de entidades tengan exactamente los mismos valores de sus atributos.

Una *clave* permite identificar un conjunto de atributos suficiente para distinguir las entidades entre sí. Las claves también ayudan a identificar únicamente a las relaciones y así a distinguir las relaciones entre sí.

2.3.1. Conjuntos de entidades

Una **superclave** es un conjunto de uno o más atributos que, tomados colectivamente, permiten identificar de forma única una entidad en el conjunto de entidades. Por ejemplo, el atributo *id-cliente* del conjunto de entidades *cliente* es suficiente para distinguir una entidad *cliente* de las otras. Así, *id-cliente* es una superclave. Análogamente, la combinación de *nombre-cliente* e *id-cliente* es una superclave del conjunto de entidades *cliente*. El atributo *nombre-cliente* de *cliente* no es una superclave, porque varias personas podrían tener el mismo nombre.

El concepto de una superclave no es suficiente para lo que aquí se propone, ya que, como se ha visto, una superclave puede contener atributos innecesarios. Si K es una superclave, entonces también lo es cualquier superconjunto de K . A menudo interesan las superclaves tales que los subconjuntos propios de ellas no son superclave. Tales superclaves mínimas se llaman **claves candidatas**.

Es posible que conjuntos distintos de atributos pudieran servir como clave candidata. Supóngase que una combinación de *nombre-cliente* y *calle-cliente* es suficiente para distinguir entre los miembros del conjunto de entidades *cliente*. Entonces, los conjuntos $\{id-cliente\}$ y $\{nombre-cliente, calle-cliente\}$ son claves candi-

tas. Aunque los atributos *id-cliente* y *nombre-cliente* juntos puedan distinguir entidades *cliente*, su combinación no forma una clave candidata, ya que el atributo *id-cliente* por sí solo es una clave candidata.

Se usará el término **clave primaria** para denotar una clave candidata que es elegida por el diseñador de la base de datos como elemento principal para identificar las entidades dentro de un conjunto de entidades. Una clave (primaria, candidata y superclave) es una propiedad del conjunto de entidades, más que de las entidades individuales. Cualesquier dos entidades individuales en el conjunto no pueden tener el mismo valor en sus atributos clave al mismo tiempo. La designación de una clave representa una restricción en el desarrollo del mundo real que se modela.

Las claves candidatas se deben designar con cuidado. Como se puede comprender, el nombre de una persona es obviamente insuficiente, ya que hay mucha gente con el mismo nombre. En España, el D.N.I. puede ser una clave candidata. Como los no residentes en España normalmente no tienen D.N.I., las empresas internacionales pueden generar sus propios identificadores únicos. Una alternativa es usar alguna combinación única de otros atributos como clave.

La clave primaria se debería elegir de manera que sus atributos nunca, o muy raramente, cambien. Por ejemplo, el campo dirección de una persona no debería formar parte de una clave primaria, porque probablemente cambiará. Los números de D.N.I., por otra parte, es seguro que no cambiarán. Los identificadores únicos generados por empresas generalmente no cambian, excepto si se fusionan dos empresas; en tal caso el mismo identificador puede haber sido emitido por ambas empresas y es necesario la reasignación de identificadores para asegurarse de que sean únicos.

2.3.2. Conjuntos de relaciones

La clave primaria de un conjunto de entidades permite distinguir entre las diferentes entidades del conjunto. Se necesita un mecanismo similar para distinguir entre las diferentes relaciones de un conjunto de relaciones.

Sea R un conjunto de relaciones que involucra los conjuntos de entidades E_1, E_2, \dots, E_n . Sea *clave-prima-*

$ria(E_i)$ el conjunto de atributos que forma la clave primaria para el conjunto de entidades E_i .

Asúmase por el momento que los nombres de los atributos de todas las claves primarias son únicos y que cada conjunto de entidades participa sólo una vez en la relación. La composición de la clave primaria para un conjunto de relaciones depende de la estructura de los atributos asociados al conjunto de relaciones R .

Si el conjunto de relaciones R no tiene atributos asociados, entonces el conjunto de atributos:

$$\begin{aligned} \text{clave-primaria}(E_1) \cup \text{clave-primaria}(E_2) \cup \dots \\ \cup \text{clave-primaria}(E_n) \end{aligned}$$

describe una relación individual en el conjunto R .

Si el conjunto de relaciones R tiene atributos a_1, a_2, \dots, a_m asociados a él, entonces el conjunto de atributos

$$\begin{aligned} \text{clave-primaria}(E_1) \cup \text{clave-primaria}(E_2) \cup \dots \\ \cup \text{clave-primaria}(E_n) \cup \{a_1, a_2, \dots, a_m\} \end{aligned}$$

describe una relación individual en el conjunto R .

En ambos casos, el conjunto de atributos

$$\begin{aligned} \text{clave-primaria}(E_1) \cup \text{clave-primaria}(E_2) \cup \dots \\ \cup \text{clave-primaria}(E_n) \end{aligned}$$

forma una superclave para el conjunto de relaciones.

En el caso de que los nombres de atributos de las claves primarias no sean únicos en todos los conjuntos de entidades, los atributos se renombran para distinguirlos; el nombre del conjunto de entidades combinado con el atributo formaría un nombre único. En el caso de que

un conjunto de entidades participe más de una vez en un conjunto de relaciones (como en la relación *trabaja-para* del Apartado 2.1.2) el nombre del papel se usa en lugar del nombre del conjunto de entidades para formar un nombre único de atributo.

La estructura de la clave primaria para el conjunto de relaciones depende de la correspondencia de cardinalidades asociada al conjunto de relaciones. Como ilustración, considérese el conjunto de entidades *cliente* y *cuenta*, y un conjunto de relaciones *impositor*, con el atributo *fecha-acceso* del Apartado 2.1.2. Supóngase que el conjunto de relaciones es varios a varios. Entonces la clave primaria de *impositor* consiste en la unión de las claves primarias de *cliente* y *cuenta*. Sin embargo, si un cliente puede tener sólo una cuenta —es decir, si la relación *impositor* es varios a uno de *cliente* a *cuenta*— entonces la clave primaria de *impositor* es simplemente la clave primaria de *cliente*. Análogamente, si la relación es varios a uno de *cuenta* a *cliente* —es decir, cada cuenta pertenece a lo sumo a un cliente— entonces la clave primaria de *impositor* es simplemente la clave primaria de *cuenta*. Para relaciones uno a uno se puede usar cualquier clave primaria.

Para las relaciones no binarias, si no hay restricciones de cardinalidad, entonces la superclave formada como se describió anteriormente en este apartado es la única clave candidata, y se elige como clave primaria. La elección de la clave primaria es más complicada si aparecen restricciones de cardinalidad. Ya que no se ha discutido cómo especificar restricciones de cardinalidad en relaciones no binarias, no se discutirá este aspecto en este capítulo. Se considerará este aspecto con más detalle en el apartado 7.3.

2.4. CUESTIONES DE DISEÑO

Las nociones de conjunto de entidades y conjunto de relaciones no son precisas, y es posible definir un conjunto de entidades y las relaciones entre ellas de diferentes formas. En este apartado se examinan cuestiones básicas de diseño de un esquema de bases de datos E-R. El proceso de diseño se trata con más detalle en el Apartado 2.7.4.

2.4.1. Uso de conjuntos de entidades o atributos

Considérese el conjunto de entidades *empleado* con los atributos *nombre-empleado* y *número-teléfono*. Se puede argumentar fácilmente que un *teléfono* es una entidad por sí misma con atributos *número-teléfono* y *ubicación* (la oficina donde está ubicado el teléfono). Si se toma este punto de vista, el conjunto de entidades *empleado* debe ser redefinido como sigue:

- El conjunto de entidades *empleado* con el atributo *nombre-empleado*
- El conjunto de entidades *teléfono* con atributos *número-teléfono* y *ubicación*
- La relación *empleado-teléfono*, que denota la asociación entre empleados y los teléfonos que tienen.

¿Cuál es, entonces, la diferencia principal entre esas dos definiciones de un empleado? Al tratar un teléfono como un atributo *número-teléfono* implica que cada empleado tiene precisamente un número de teléfono. Al tratar un teléfono como una entidad *teléfono* permite que los empleados puedan tener varios números de teléfono (incluido ninguno) asociados a ellos. Sin embargo, se podría definir fácilmente *número-teléfono* como un atributo multivalorado para permitir varios teléfonos por empleado.

La diferencia principal es que al tratar un teléfono como una entidad se modela mejor una situación en la que se puede querer almacenar información extra sobre un teléfono, como su ubicación, su tipo (móvil, video-teléfono o fijo) o quiénes comparten un teléfono. Así, al tratar un teléfono como una entidad es más general que tratarlo como un atributo y es apropiado cuando la generalidad pueda ser de utilidad.

En cambio, no sería adecuado tratar el atributo *nombre-empleado* como una entidad; es difícil argumentar que *nombre-empleado* sea una entidad por sí mismo (a diferencia del teléfono). Así, es apropiado tener *nombre-empleado* como un atributo del conjunto de entidades *empleado*.

Por tanto, aparecen dos cuestiones naturales: ¿qué constituye un atributo? y ¿qué constituye un conjunto de entidades? Por desgracia no hay respuestas simples. Las distinciones dependen principalmente de la estructura de la empresa del mundo real que se esté modelando y de la semántica asociada con el atributo en cuestión.

Un error común es usar la clave primaria de un conjunto de entidades como un atributo de otro conjunto de entidades, en lugar de usar una relación. Por ejemplo, es incorrecto modelar *id-cliente* como un atributo de *préstamo* incluso si cada préstamo tiene sólo un cliente. La relación *prestatario* es la forma correcta de representar la conexión entre préstamos y clientes, ya que hace su conexión explícita en lugar de implícita mediante un atributo.

Otro error relacionado que se comete es designar a los atributos de la clave primaria de los conjuntos de entidades relacionados como atributos del conjunto de relaciones. Esto no se debería hacer, ya que los atributos de la clave primaria son ya implícitos en la relación.

2.4.2. Uso de conjuntos de entidades o conjuntos de relaciones

No siempre está claro si es mejor expresar un objeto mediante un conjunto de entidades o mediante un conjunto de relaciones. En el Apartado 2.1.1 se asumió que un préstamo se modelaba como una entidad. Una alternativa es modelar un préstamo no como una entidad, sino como una relación entre clientes y sucursales, con *número-préstamo* e *importe* como atributos descriptivos. Cada préstamo se representa mediante una relación entre un cliente y una sucursal.

Si cada préstamo está asociado exactamente con un cliente y con una sucursal, se puede encontrar satisfactorio el diseño en el que un préstamo se representa como una relación. Sin embargo, con este diseño no se puede representar convenientemente una situación en que varios clientes comparten un préstamo. Habría que definir una relación separada para cada prestatario de ese préstamo común. Entonces habría que replicar los valores para los atributos descriptivos *número-préstamo* e *importe* en cada una de estas relaciones. Cada una de

estas relaciones debe, por supuesto, tener el mismo valor para los atributos descriptivos *número-préstamo* e *importe*.

Surgen dos problemas como resultado de esta réplica: 1) los datos se almacenan varias veces, desperdi-ciando espacio de almacenamiento; y 2) las actualizaciones dejan potencialmente los datos en un estado inconsistente, en el que los valores difieren en dos relaciones para atributos que se supone tienen el mismo valor. El asunto de cómo evitar esta réplica se trata formalmente mediante la *teoría de la normalización*, discutida en el Capítulo 7.

El problema de la réplica de los atributos *número-préstamo* e *importe* no aparece en el diseño original del Apartado 2.1.1. porque *préstamo* es un conjunto de entidades.

Una posible guía para determinar si usar un conjunto de entidades o un conjunto de relaciones es designar un conjunto de relaciones para describir una acción que ocurre entre entidades. Este enfoque puede también ser útil para decidir si ciertos atributos se pueden expresar más apropiadamente como relaciones.

2.4.3. Conjuntos de relaciones binarias o n-arias

Las relaciones en las bases de datos son generalmente binarias. Algunas relaciones que parecen no ser binarias podrían ser representadas mejor con varias relaciones binarias. Por ejemplo, uno podría crear una relación ternaria *padres*, que relaciona un hijo con su padre y su madre. Sin embargo, tal relación se podría representar por dos relaciones binarias *padre* y *madre*, relacionando un hijo con su padre y su madre por separado. Al usar las dos relaciones *padre* y *madre* se permite registrar la madre de un niño incluso si no se conoce la identidad del padre; en la relación ternaria *padres* se necesitaría usar un valor nulo. En este caso es preferible usar conjuntos de relaciones binarias.

De hecho, siempre es posible reemplazar un conjunto de relaciones no binarias (*n*-aria, para $n > 2$) por un número de diferentes conjuntos de relaciones binarias. Por simplicidad, considérese el conjunto de relaciones abstracto *R*, ternario ($n = 3$), y los conjuntos de entidades *A*, *B*, y *C*. Se sustituye el conjunto de relaciones *R* por un conjunto de entidades *E* y se crean tres conjuntos de relaciones:

- R_A , relacionando *E* y *A*
- R_B , relacionando *E* y *B*
- R_C , relacionando *E* y *C*

Si el conjunto de relaciones *R* tiene atributos, éstos se asignan al conjunto de entidades *E*; por otra parte se crea un atributo de identificación especial para *E* (debido a que cada conjunto de entidades debe tener al menos un atributo para distinguir los miembros del conjunto).

Para cada relación (a_i, b_i, c_i) del conjunto de relaciones R , se crea una nueva entidad e_i en el conjunto de entidades E . Entonces, en cada uno de los tres nuevos conjuntos de relaciones, se inserta un nuevo miembro como sigue:

- (e_i, a_i) en R_A
- (e_i, b_i) en R_B
- (e_i, c_i) en R_C

Se puede generalizar este proceso de una forma semejante a conjuntos de relaciones n -arias. Así, conceptualmente, se puede restringir el modelo E-R para incluir sólo conjuntos de relaciones binarias. Sin embargo, esta restricción no siempre es deseable.

- Un atributo de identificación puede haber sido creado para el conjunto de entidades para representar el conjunto de relaciones. Este atributo, con los conjuntos de relaciones extra necesarios, incrementa la complejidad del diseño y (como se verá en el Apartado 2.9) los requisitos de almacenamiento.
- Un conjunto de relaciones n -arias muestra más claramente que varias entidades participan en una relación simple.
- Podría no haber una forma de traducir restricciones en la relación ternaria en restricciones sobre relaciones binarias. Por ejemplo, considérese una restricción que dice que R es varios a uno de A, B a C ; es decir, cada par de entidades de A y B se asocia con a lo sumo una entidad C . Esta restricción no se puede expresar usando restricciones de cardinalidad sobre los conjuntos de relaciones R_A, R_B y R_C .

Considérese el conjunto de relaciones *trabaja-en* del Apartado 2.1.2 que relaciona *empleado*, *sucursal* y *trabajo*. No se puede dividir directamente *trabaja-en* en relaciones binarias entre *empleado* y *sucursal* y entre *empleado* y *trabajo*. Si se hiciese habría que registrar que Santos es director y auditor y que Santos trabaja en Navacerrada y Centro; sin embargo, no se podría registrar que Santos es director de Navacerrada y auditor de Centro, pero que no es auditor de Navacerrada y director de Centro.

El conjunto de relaciones *trabaja-en* se puede dividir en relaciones binarias creando nuevos conjuntos de entidades como se describió anteriormente. Sin embargo, no sería muy natural.

2.4.4. Ubicación de los atributos de las relaciones

La razón de cardinalidad de una relación puede afectar a la situación de los atributos de la relación. Los atributos de los conjuntos de relaciones uno a uno o uno a varios pueden estar asociados con uno de los conjuntos de entidades participantes, en lugar de con el conjunto

de relaciones. Por ejemplo, especificamos que *impositor* es un conjunto de relaciones uno a varios tal que un cliente puede tener varias cuentas, pero cada cuenta está asociada únicamente con un cliente. En este caso, el atributo *fecha-acceso*, que especifica cuándo accedió por última vez el cliente a la cuenta, podría estar asociado con el conjunto de entidades *cuenta*, como se describe en la Figura 2.6; para mantener la simplicidad de la figura sólo se muestran algunos de los atributos de los dos conjuntos de entidades. Como cada entidad *cuenta* participa en una relación con a lo sumo un ejemplar de *cliente*, hacer esta designación de atributos tendría el mismo significado que si se colocase *fecha-acceso* en el conjunto de relaciones *impositor*. Los atributos de un conjunto de relaciones uno a varios se pueden colocar sólo en el conjunto de entidades de la parte «varios» de la relación. Por otra parte, para los conjuntos de entidades uno a uno, los atributos de la relación se pueden asociar con cualquiera de las entidades participantes.

La decisión de diseño de dónde colocar los atributos descriptivos en tales casos —como un atributo de la relación o de la entidad— podría reflejar las características de la empresa que se modela. El diseñador puede elegir mantener *fecha-acceso* como un atributo de *impositor* para expresar explícitamente que ocurre un acceso en el punto de interacción entre los conjuntos de entidades *cliente* y *cuenta*.

La elección de la colocación del atributo es más clara para los conjuntos de relaciones varios a varios. Volviendo al ejemplo, especificamos el caso quizás más realista de *impositor* que es un conjunto de relaciones varios a varios, expresando que un cliente puede tener una o más cuentas, y que una cuenta puede ser mantenida por uno o más clientes. Si se expresa la fecha en que un cliente específico accedió por última vez a una cuenta específica, *fecha-acceso* debe ser un atributo del conjunto de relaciones *impositor*, en lugar de una de las entidades participantes. Si *fecha-acceso* fuese un atributo de *cuenta*, por ejemplo, no se podría determinar

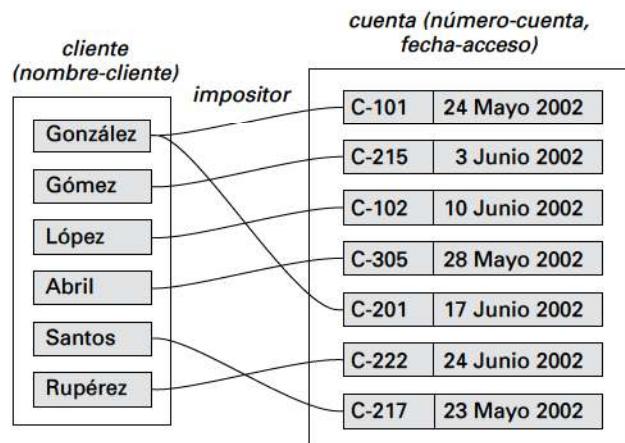


FIGURA 2.6. Fecha-acceso como atributo del conjunto de entidades *cuenta*.

qué cliente hizo el acceso más reciente a una cuenta conjunta. Cuando un atributo se determina mediante la combinación de los conjuntos de entidades participantes, en lugar de por cada entidad por separado, ese atributo debe estar asociado con el conjunto de relaciones varios a

varios. La colocación de *fecha-acceso* como un atributo de la relación se describe en la Figura 2.7; de nuevo, para mantener la simplicidad de la figura, sólo se muestran algunos de los atributos de los dos conjuntos de entidades.

2.5. DIAGRAMA ENTIDAD-RELACIÓN

Como se vio brevemente en el Apartado 1.4, la estructura lógica general de una base de datos se puede expresar gráficamente mediante un **diagrama E-R**. Los diagramas son simples y claros, cualidades que pueden ser responsables del amplio uso del modelo E-R. Tal diagrama consta de los siguientes componentes principales:

- **Rectángulos**, que representan conjuntos de entidades.
- **Elipses**, que representan atributos.
- **Rombos**, que representan relaciones.
- **Líneas**, que unen atributos a conjuntos de entidades y conjuntos de entidades a conjuntos de relaciones.
- **Elipses dobles**, que representan atributos multivariados.
- **Elipses discontinuas**, que denotan atributos derivados.
- **Líneas dobles**, que indican participación total de una entidad en un conjunto de relaciones.

- **Rectángulos dobles**, que representan conjuntos de entidades débiles (se describirán posteriormente en el Apartado 2.6).

Considérese el diagrama entidad-relación de la Figura 2.8, que consta de dos conjuntos de entidades, *cliente* y *préstamo*, relacionadas a través de un conjunto de relaciones binarias *prestatario*. Los atributos asociados con *cliente* son *id-cliente*, *nombre-cliente*, *calle-cliente*, y *ciudad-cliente*. Los atributos asociados con *préstamo* son *número-préstamo* e *importe*. Como se muestra en la Figura 2.8, los atributos de un conjunto de entidades que son miembros de la clave primaria están subrayados.

El conjunto de relaciones *prestatario* puede ser varios a varios, uno a varios, varios a uno o uno a uno. Para distinguir entre estos tipos, se dibuja o una línea dirigida (\rightarrow) o una línea no dirigida ($-$) entre el conjunto de relaciones y el conjunto de entidades en cuestión.

- Una línea dirigida desde el conjunto de relaciones *prestatario* al conjunto de entidades *préstamo* espe-

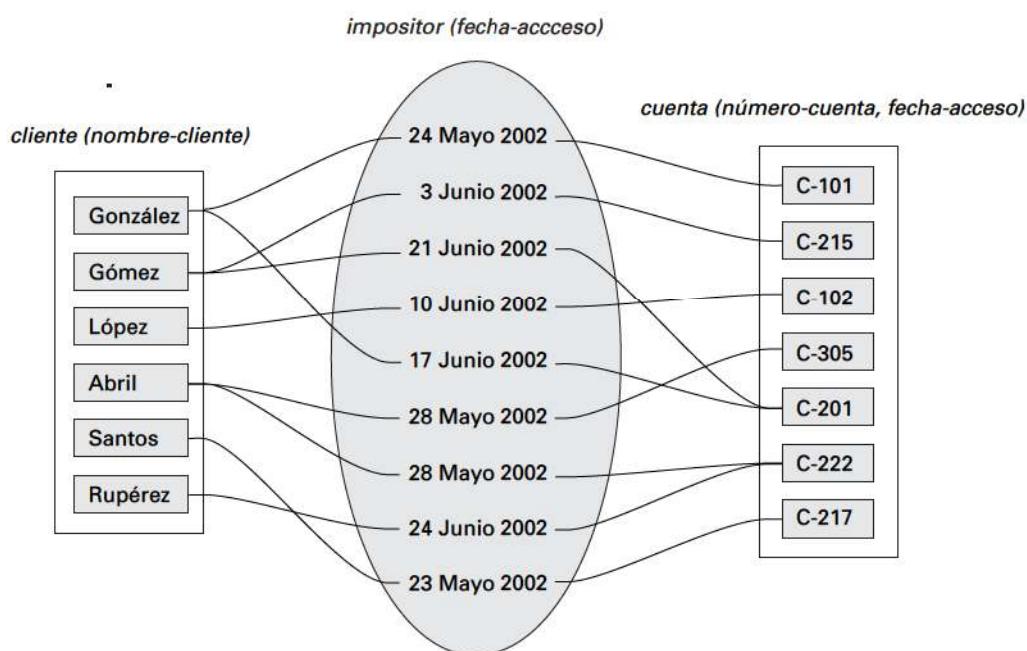


FIGURA 2.7. Fecha-acceso como atributo del conjunto de relaciones *impositor*.

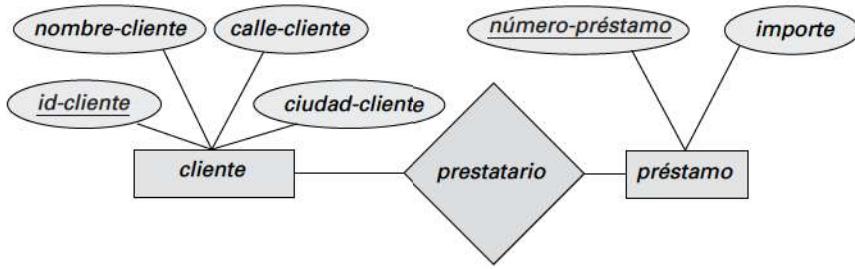


FIGURA 2.8. Diagrama E-R correspondiente a clientes y préstamos.

cifica que *prestatario* es un conjunto de relaciones uno a uno, o bien varios a uno, desde *cliente* a *préstamo*; *prestatario* no puede ser un conjunto de relaciones varios a varios ni uno a varios, desde *cliente* a *préstamo*.

- Una línea no dirigida desde el conjunto de relaciones *prestatario* al conjunto de relaciones *préstamo* especifica que *prestatario* es o bien un con-

junto de relaciones varios a varios, o bien uno a varios, desde *cliente* a *préstamo*.

Volviendo al diagrama E-R de la Figura 2.8, se ve que el conjunto de relaciones *prestatario* es varios a varios. Si el conjunto de relaciones *prestatario* fuera uno a varios, desde *cliente* a *préstamo*, entonces la línea desde *prestatario* a *cliente* sería dirigida, con una flecha apuntando al conjunto de entidades *cliente* (Figura 2.9a). Análogamente, si el conjunto de relaciones *pres-*

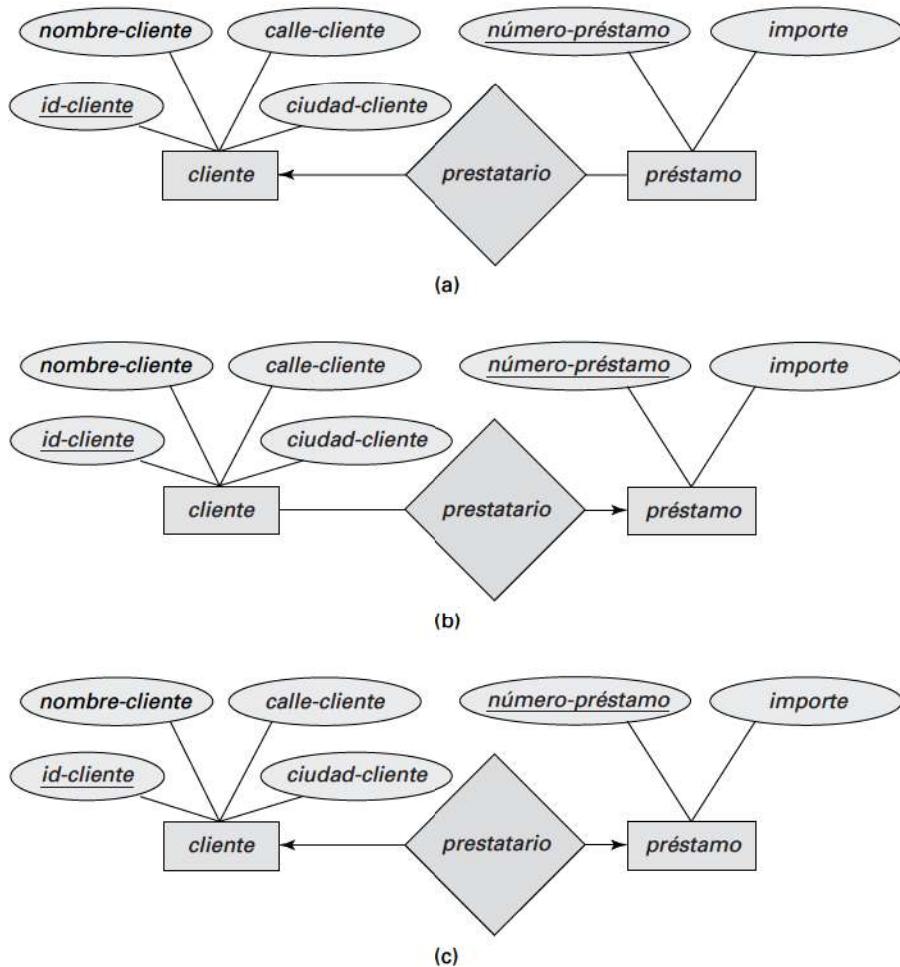


FIGURA 2.9. Relaciones. (a) Uno a varios. (b) Varios a uno. (c) Uno a uno.

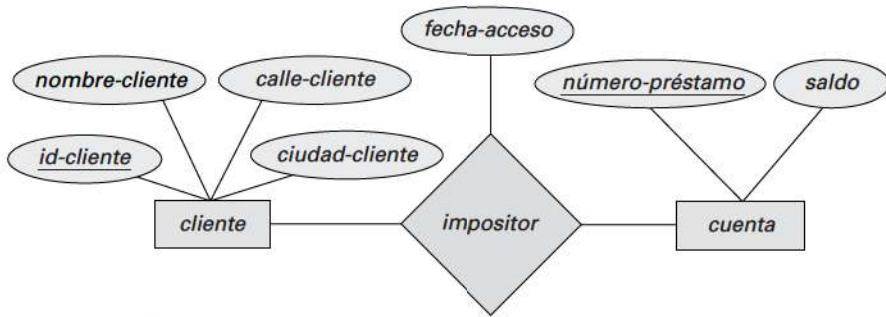


FIGURA 2.10. Diagrama E-R con un atributo unido a un conjunto de relaciones.

Si el atributo *prestatario* fuera varios a uno desde *cliente* a *préstamo*, entonces la línea desde *prestatario* a *préstamo* tendría una flecha apuntando al conjunto de entidades *préstamo* (Figura 2.9b). Finalmente, si el conjunto de relaciones *prestatario* fuera uno a uno, entonces ambas líneas desde *prestatario* tendrían flechas: una apuntando al conjunto de entidades *préstamo* y otra apuntando al conjunto de entidades *cliente* (Figura 2.9c).

Si un conjunto de relaciones tiene también algunos atributos asociados a él, entonces se unen esos atributos a ese conjunto de relaciones. Por ejemplo, en la Figura 2.10, se tiene el atributo descriptivo *fecha-acceso* unido al conjunto de relaciones *impositor* para especificar la fecha más reciente en la que un cliente accedió a esa cuenta.

La Figura 2.11 muestra cómo se pueden representar atributos compuestos en la notación E-R. Aquí, el atributo compuesto *nombre*, con atributos componentes *nombre-pila*, *primer-apellido* y *segundo-apellido* reemplaza al atributo simple *nombre-cliente* de *cliente*. También se muestra el atributo compuesto *dirección*, cuyos atributos componentes son *calle*, *ciudad*, *provincia* y *código-postal*, que reemplaza a los atributos *calle-cliente* y *ciudad-cliente* de *cliente*. El atributo *calle* es por si

mismo un atributo compuesto cuyos atributos componentes son *número-calle*, *nombre-calle* y *número-piso*.

La Figura 2.11 también muestra un atributo multivalorado, *número-teléfono*, indicado por una elipse doble, y un atributo derivado *edad*, indicado por una elipse discontinua.

En los diagramas E-R se indican papeles mediante etiquetas en las líneas que unen rombos con rectángulos. En la Figura 2.12 se muestran los indicadores de papeles *director* y *trabajador* entre el conjunto de entidades *empleado* y el conjunto de relaciones *trabaja-para*.

Los conjuntos de relaciones no binarias se pueden especificar fácilmente en un diagrama E-R. La Figura 2.13 consta de tres conjuntos de entidades *cliente*, *trabajo* y *sucursal*, relacionados a través del conjunto de relaciones *trabaja-en*.

Se pueden especificar algunos tipos de relaciones varios a uno en el caso de conjuntos de relaciones no binarias. Supóngase un empleado que tenga a lo sumo un trabajo en cada sucursal (por ejemplo, Santos no puede ser director y auditor en la misma sucursal). Esta restricción se puede especificar con una flecha apuntando a *trabajo* en el borde de *trabaja-en*.

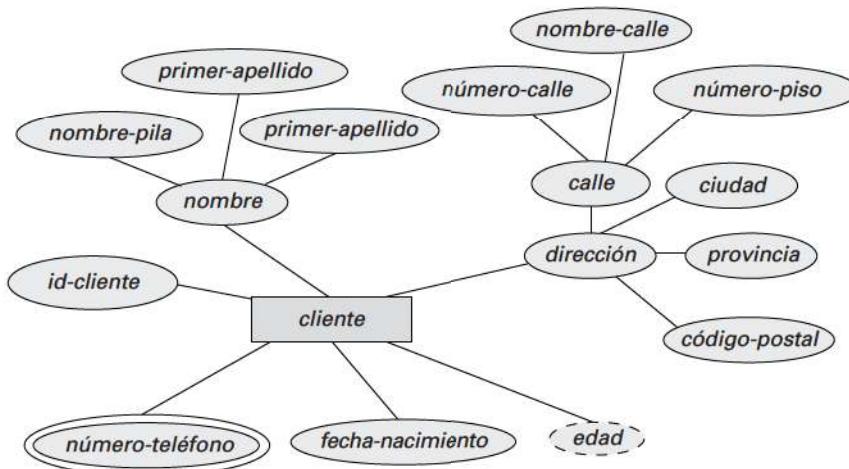


FIGURA 2.11. Diagrama E-R con atributos compuestos, multivalorados y derivados.

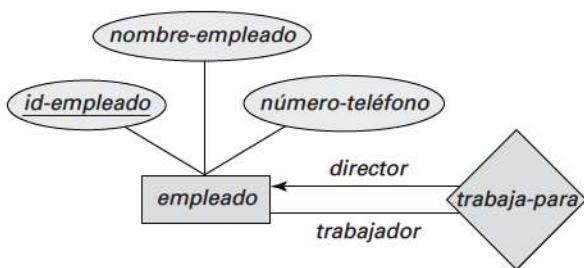


FIGURA 2.12. Diagrama E-R con indicadores de papeles.

Se permite a lo sumo una flecha desde un conjunto de relaciones, ya que un diagrama E-R con dos o más flechas salientes de un conjunto de relaciones no binarias se puede interpretar de dos formas. Supónganse que hay un conjunto de relaciones R entre conjuntos de entidades A_1, A_2, \dots, A_n , y las únicas flechas están en los bordes de los conjuntos de entidades $A_{i+1}, A_{i+2}, \dots, A_n$. Entonces, las dos posibles interpretaciones son:

1. Una combinación particular de entidades de A_1, A_2, \dots, A_i se puede asociar con a lo sumo una combinación de entidades de $A_{i+1}, A_{i+2}, \dots, A_n$. Así, la clave primaria de la relación R se puede construir por la unión de las claves primarias de A_1, A_2, \dots, A_i .
2. Para cada conjunto de entidades A_k , $i < k \leq n$, cada combinación de las entidades de los otros conjuntos de entidades se pueden asociar con a lo sumo una entidad de A_k . Cada conjunto $\{A_1, A_2, \dots, A_{k-1}, A_{k+1}, A_{i+2}, \dots, A_n\}$, para $i < k \leq n$, forma entonces una clave candidata.

Cada una de estas interpretaciones se han usado en diferentes libros y sistemas. Para evitar confusión se permite sólo una flecha que salga de un conjunto de relaciones, y así las representaciones son equivalentes. En el Capítulo 7 (Apartado 7.3) se estudia la noción de *dependencias funcionales*, que permiten especificar cualquiera de estas dos interpretaciones sin ambigüedad.

En el diagrama E-R se usan las líneas dobles para indicar que la participación de un conjunto de entidades en un conjunto de relaciones es total; es decir, cada entidad en el conjunto de entidades aparece al menos en una relación en ese conjunto de relaciones. Por ejemplo, considérese la relación *prestamista* entre clientes y préstamos. Una línea doble de *préstamo a prestamista*, como en la Figura 2.14, indica que cada préstamo debe tener al menos un cliente asociado.

Los diagramas E-R también proporcionan una forma de indicar restricciones más complejas sobre el número de veces en que cada entidad participa en las relaciones de un conjunto de relaciones. Un segmento entre un conjunto de entidades y un conjunto de relaciones binarias puede tener una cardinalidad mínima y máxima, mostrada de la forma $mín..máx$, donde $mín$ es la mínima cardinalidad y $máx$ es la máxima. Un valor mínimo de 1 indica una participación total del conjunto de entidades en el conjunto de relaciones. Un valor máximo de 1 indica que la entidad participa de a lo sumo una relación, mientras que un valor máximo de * indica que no hay límite. Nótese que una etiqueta $1..*$ en un segmento es equivalente a una línea doble.

Por ejemplo, considérese la Figura 2.15. El segmento entre *préstamo* y *prestamista* tiene una restricción de cardinalidad de $1..1$, significando que la cardinalidad mínima y máxima son ambas 1. Es decir, cada préstamo debe tener exactamente un cliente asociado. El límite $0..*$ en el segmento de *cliente* a *prestamista* indica que un cliente puede tener ninguno o varios préstamos. Así, la relación *prestamista* es uno a varios de *cliente* a *préstamo*, y además la participación de *préstamo* en *prestamista* es total.

Es fácil malinterpretar $0..*$ en el segmento entre *cliente* y *prestamista*, y pensar que la relación *prestamista* es de varios a uno de *cliente* a *préstamo* —esto es exactamente lo contrario a la interpretación correcta.

Si ambos segmentos de una relación binaria tienen un valor máximo de 1, la relación es uno a uno. Si se hubiese especificado un límite de cardinalidad de $1..*$ en el segmento entre *cliente* y *prestamista*, se estaría diciendo que cada cliente debe tener al menos un préstamo.



FIGURA 2.13. Diagrama E-R con una relación ternaria.